

Informatik

Geodatenverarbeitung mit Budget Instanzen

Semesterarbeit



Abbildung 1: Eine Tour mit Budget Instanzen [Web10].

| | |
|------------------|--------------------------------|
| Departement: | Informatik |
| Kurs: | CAS CLD FS20 – Cloud Computing |
| Autor: | Tobias Reber |
| Experte: | Jörg Thomann |
| Ansprechpartner: | Christoph Böcklin |
| Datum: | 28. 08. 2020 |

Perplexity
is the beginning of knowledge.

- Kahlil Gibran

[AD19, S. 33]

1 Management Summary

In der Kürze liegt die Würze

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Management Summary | ii |
| 2 Einführung | 1 |
| 2.1 GIS - Geografische Informationssysteme | 1 |
| 3 Vorgehen | 2 |
| 3.1 Arbeitsmethodik | 2 |
| 3.2 Projektplan | 2 |
| 4 Ausgangslage | 3 |
| 4.1 swisstopo bei AWS | 3 |
| 4.2 Publikation von Geodaten | 3 |
| 4.3 Die Web Services geben den Technologie Stack vor | 3 |
| 4.4 Exkurs 3D Daten | 4 |
| 5 Der Use Case | 5 |
| 5.1 Problemstellung | 5 |
| 5.1.1 Budget Instanzen | 5 |
| 5.2 Ist-Zustand der 3D Datenpublikation | 6 |
| 5.2.1 3D Datenpublikation | 6 |
| 5.2.2 Aufwand der Verarbeitung | 7 |
| 5.2.3 Technische Komponenten | 7 |
| 6 Architektur | 8 |
| 6.1 Analyse des Ist-Zustandes | 8 |
| 6.1.1 Bereitstellung der Rohdaten und Sicherstellung dessen Qualität | 8 |
| 6.1.2 Bereitstellung der Infrastruktur für die Verarbeitung | 8 |
| 6.2 Bewertungskriterien | 8 |
| 6.3 Architekturentscheid | 9 |
| 6.3.1 Verarbeitung auf Spot Instanzen | 9 |
| 7 Realisierung des Prototyps | 10 |
| 7.1 Die Datenverarbeitung als Code | 10 |
| 7.2 Spot Flottenanfrage | 11 |
| 7.3 Handling der Interrupts | 11 |
| 7.4 Testen der Datenstruktur | 11 |
| 7.5 Logging | 11 |
| 7.6 Inhaltliche Kontrolle der publizierten Daten | 11 |
| 7.7 Testen | 11 |
| 7.7.1 Unerwartete Interrupts | 11 |
| 7.7.2 Full-load test | 11 |
| 8 Evaluation | 12 |
| 8.1 Erfahrungen | 12 |
| 8.2 Wirtschaftlichkeit | 12 |
| 8.3 Kritische Punkte | 13 |
| 8.3.1 Security | 13 |
| 8.3.2 Datenverarbeitung als Blackbox | 13 |
| 9 Ausblick | 14 |
| Literaturverzeichnis | 15 |
| A Fachbegriffe und Abkürzungen | I |

| | |
|---|------------|
| B Kanban | II |
| C Projektplan | II |
| D Konfigurationen und Kommandos | III |
| D.1 EFS auf EC2-Instanz mounten | III |
| D.2 Von der SPOT Instanz aus abfragen, was ihr Status ist | V |
| D.3 XML Testen | VI |
| D.4 Publikationsschritte in Ansible | VII |

Abbildungsverzeichnis

| | | |
|---|--|----|
| 1 | Eine Tour mit Budget Instanzen [Web10]. | 1 |
| 2 | Internetkarte des Bundes <i>map.geo.admin.ch</i> . Hier ein Ausschnitt der Saane bei Kleinbödingen, das Luftaufnahmen von 1946 mit heute vergleicht. | 1 |
| 3 | Im Viewer werden zurzeit Gebäude, Bäume, Seilbahnen, Namen und das Terrain dargestellt. Um aktuell zu bleiben, müssen diese 3D Daten regelmässig nachgeführt werden. | 4 |
| 4 | So funktioniert das Ausleihen von Budget (SPOT) Instanzen | 5 |
| 5 | Arbeitsschritte, die es braucht, um die 3D Daten zu Publizieren. | 6 |
| 6 | Geodatenverarbeitung mit SPOT Instanzen. | 10 |
| 7 | Klassisches Kanban auf <i>github.com</i> | II |
| 8 | Projektplan. Die orangen Meilensteine wurden von der BFH vorgegeben. | II |

Tabellenverzeichnis

| | | |
|---|---|----|
| 1 | Relativ betrachtet ist das Sparpotential enorm: 76%. | 12 |
| 2 | In der Arbeit verwendete Fachbegriffe und Abkürzungen | I |

2 Einführung

Ich arbeite als GIS-Spezialist bei der swisstopo¹, dem Bundesamt für Landestopografie, in Wabern. Wir machen Karten. Unser Team macht Internetkarten - wie Google Maps², jedoch von der Schweiz für die Schweiz; und für alle anderen auch. Unsere Internetkarte, der Viewer, erfreut sich relativ grosser Beliebtheit und beinhaltet ca. 800 Themen wie Wanderwege, Solarkataster und Luftfahrthindernisse. Lieber Leser³, falls dir map.geo.admin.ch noch kein Begriff sein sollte, kann ich dir wärmstens empfehlen, darin zu schmökern. Es gibt viel zu entdecken und es ist gratis - ein Service Public.



Abbildung 2: Internetkarte des Bundes *map.geo.admin.ch*. Hier ein Ausschnitt der Saane bei Kleinbödingen, das Luftaufnahmen von 1946 mit heute vergleicht.

2.1 GIS - Geografische Informationssysteme

Wie erwähnt, arbeite ich als *GIS-Spezialist*. Wobei mir der Titel *Geoinformatiker* besser gefällt: weil er die Begriffe *Geografie* und *Informatik* vereint. *Geografie* kommt aus dem Griechischen und bedeutet Erdbeschreibung [SE04, S. 14]. *Informatik* ist die Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen [10c].

GIS ist ein Akronym für Geografische Informationssysteme. Es bedeutet im engsten Sinn eine Ansammlung von Computerprogrammen, die zur Bearbeitung von Karten und Geodaten verwendet werden. Geodaten sind nichts weiter als Daten mit einem räumlichen Bezug⁴. In einem weiteren Sinn deckt der Begriff GIS ein ganzes Fachgebiet ab, das sich mit Karten und Geodaten auskennt. Es ist also nicht nur ein Werkzeug, sondern ein Fachgebiet, das Kenntnisse über Datensammlung, Speicherung, Analyse und Darstellung innerhalb von vielen verschiedenen Themen mit einem räumlichen Bezug abdeckt. Typische Geodaten sind digitale Karten, Inventare und Register von Parzellen, Umweltfaktoren, Grenzen, Entwicklung, Planung etc., die einen räumlichen Bezug haben und dadurch in einem geografischen Zusammenhang analysiert und dargestellt werden können [BJB06, S. 15].

Es ist zwar weit von der klassischen Geografie zu Zeiten Alexanders von Humboldt⁵ entfernt, aber es liegt auf der Hand, dass auch die Geografie Cloud Computing nutzt.

¹www.swisstopo.ch

²maps.google.com

³Im vorliegenden Dokument wird durchwegs der männliche Singular (Leser, Benutzer) als Ansprache verwendet. Diese Ansprache bezieht sich auf beide Geschlechter sowie gegebenenfalls mehrere Personen. Sie dient lediglich der leichteren Lesbarkeit der Semesterarbeit.

⁴Mit Koordinaten (Nord/Ost, x/y).

⁵Ein Forschungsreisender des 19. Jh. mit einem weit über Europa hinausreichenden Wirkungsfeld. Mehrjährige Forschungsreisen führten ihn nach Lateinamerika, USA und nach Zentralasien [Keh05].

3 Vorgehen

In diesem Kapitel wird das Vorgehen beschrieben, wie die Arbeit geplant und realisiert wurde.

3.1 Arbeitsmethodik

Da die Arbeit von einem einzigen Autor umgesetzt wurde, musste bezüglich Arbeitsmethodik nur wenig definiert und koordiniert werden.

Um zu starten, hat dem Autor geholfen, dass der Experte Jörg Thomann bereits anfangs Sommer (16. Juli) einen ersten Besprechungstermin angesetzt hat. Um eine Diskussionsbasis für diesen Termin zu haben, musste sich der Autor ernsthaft mit der Semesterarbeit auseinandersetzen. Dazu hat er aus einer BFH Vorlage ⁶ ein Gerüst der Arbeit erstellt und stichwortartig abgefüllt. Nach gründlicher Diskussion mit dem Experten wurde dieses Gerüst etwas angepasst und diente von da an als Basis für die Arbeit; wo sich die Kapitel zunehmend von losen Notizen zum finalen Zustand festigten.

Um die Übersicht der anstehenden Aufgaben nicht zu verlieren, wurde beschlossen, nach Kanban zu arbeiten. Weil für die Arbeit sowieso Repository auf *github.com*⁷ angelegt wurde, konnte dort auch gleich ein Kanban Board erstellt werden. Das Kanban Board wurde vor allem auch dafür genutzt, fortlaufend neue kleine Aufgaben aufzulisten und darauf zu achten, die Menge der angefangenen Arbeit zu begrenzen.

Dadurch konnte im Alltag (wenn gerade etwas Zeit zur Verfügung stand) eine solche Aufgabe genommen und erledigt werden. Es wurde nicht strikte nach Kanban gearbeitet, aber einige Prinzipien daraus wurden verwendet. Zum Beispiel wurde das erste Grundprinzip angewendet *"Beginne mit dem, was du gerade tust: In diesem Grundprinzip stecken zwei Dinge. Indem man mit dem beginnt, was man gerade tut, beendet man die aktuelle Arbeit, bevor etwas Neues begonnen wird. Genauso ist hier aber auch die Aussage enthalten, dass Kanban einfach eingeführt werden kann."*[10d]. Eine Momentaufnahme des Kanbans befindet sich im Anhang B.

3.2 Projektplan

Um den übergeordneten Rahmen der Arbeit nicht aus dem Fokus zu verlieren und um sich bewusst zu machen, wieviel Zeit generell bis zur Abgabe zur Verfügung steht, war es hilfreich, einen Projektplan mit Meilensteinen konsultieren zu können. Durch die Berner Fachhochschule waren die grundlegenden Meilensteine bereits vorgegeben. Der Projektplan befindet sich im Anhang C.

⁶Herzlichen Dank an Andreas Habegger und Lukas Studer fürs Bereitstellen der LaTeX Vorlage.

⁷Projekt auf *github.com*.

4 Ausgangslage

Hier werden das Arbeitsumfeld und die Aufgaben beschrieben, aus denen sich die Problemstellung dieser Semesterarbeit ergeben hat.

4.1 swisstopo bei AWS

Es liegt auf der Hand, dass die swisstopo in ihrer Rolle als *Geoinformationszentrum* auf Cloud Computing setzt. Die swisstopo nutzt Cloud Computing mit AWS⁸ seit mehr als 10 Jahren für den Betrieb des Geoportal des Bundes.

"Mit der BGD⁹ unter AWS können wir derzeit ca. eine Million Internetbenutzer pro Monat bedienen. Dank AWS können wir die zur Zuordnung neuer Server benötigte Zeit erheblich verkürzen und unseren Fokus auf echte Kundenanforderungen verstärken." [Chr20].

4.2 Publikation von Geodaten

Wie bereits erwähnt, können auf dem Viewer ca. 800 Themen wie Wanderwege, Solarkataster und Luftfahrthindernisse angesehen werden. Unser Team publiziert diese Daten. Der Nachführungszyklus wie auch der Aufwand zur Aufbereitung der Daten fürs Web sind unterschiedlich. Einige Daten werden manuell aufwändig aufbereitet, andere stündlich automatisch nachgeführt.

4.3 Die Web Services geben den Technologie Stack vor

Nebst der Publikation der Daten ist unser Team für den Betrieb und der Weiterentwicklung der Web Services und des Viewers verantwortlich. Der ganze Technologie Stack wurde schon länger nicht mehr grundlegend erneuert. Zurzeit wird die gesamte Architektur analysiert und überarbeitet, um eine gestaffelte Migration auf eine neue Lösung zu ermöglichen. Einige Rahmenbedingungen dieser zukünftigen Architektur sind bereits klar: Freie Software auf Linux¹⁰. Das Geoportal des Bundes wird weiterhin in der AWS Cloud betrieben werden, die Migration wird vor allem über Microservices gestaffelt erfolgen, diese Services werden als Docker Container laufen, Amazon ElastiKubernetes Service wird die Orchestrierung der Container übernehmen; und für Continuous Integration wird AWS Codebuild/Pipeline zum Einsatz kommen.

⁸Amazon Web Services

⁹Bundesgeodateninfrastruktur: Viewer und andere Services.

¹⁰Wann immer möglich, freie Software [Sta10]: OpenLayers, PostGIS, Debian, Mapserver, Python Frameworks, Kubernetes etc.

4.4 Exkurs 3D Daten

Die swisstopo erfasst und aktualisiert Daten mit einem räumlichen Bezug. Diese Geodaten sind die Basis für die Ableitung in eine Vielzahl von Produkten, wie die Landeskarten 1:25'000. Nebst Karten gibt es unter anderem die Produktpalette der Landschaftsmodelle. Diese geben die Objekte der Landschaft im flexiblen Vektorformat wieder. Sie bestehen aus thematischen Ebenen (Bsp. Gebäude). Jede Ebene umfasst georeferenzierte Punkt-, Linien-, Flächen- oder 3D Objekte. Jedes Objekt enthält Attribute und Beziehungen [10e].

Zu den Landschaftsmodellen gehören Produkte wie swissTLM3D und swissBuildings3D. Im Viewer wird eine Auswahl von Themen aus eben diesen Landschaftsmodellen dargestellt: Zurzeit Gebäude, Bäume, Seilbahnen, Namen und das Terrain. Vor wenigen Jahren wurden diese 3D Daten mit einem grossen Effort medienwirksam publiziert.

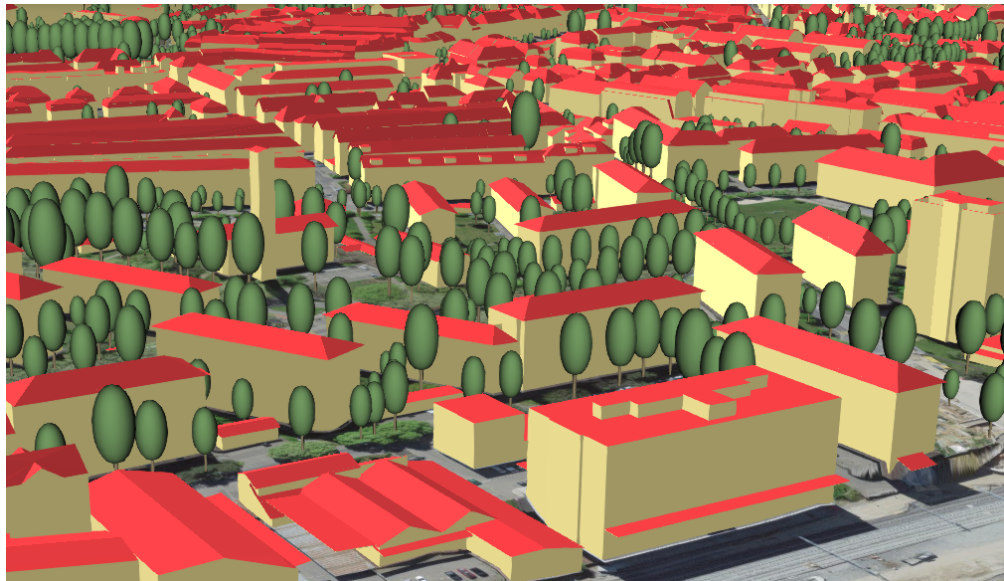


Abbildung 3: Im Viewer werden zurzeit Gebäude, Bäume, Seilbahnen, Namen und das Terrain dargestellt. Um aktuell zu bleiben, müssen diese 3D Daten regelmässig nachgeführt werden.

Seit der Erstpublikation ist inzwischen etwas Zeit vergangen. Als die ersten Aktualisierungen der Daten anstanden, wurde den Beteiligten bewusst, dass sich diese nicht einfach so *auf Knopfdruck* realisieren lässt: Seit der Erstpublikation hat es personelle Wechsel gegeben und punkto Dokumentation und Automatisierungsgrad wurden Lücken identifiziert.

Es gibt immer gute Gründe für *technische Schulden*, wie in diesem Fall für positive medienwirksame Reaktionen¹¹. Früher oder später müssen diese abgebaut werden, weil es einen direkten Einfluss auf die Wartbarkeit des Produktes hat [10f].

¹¹Wie Bsp. auf watson.ch oder Twitter [18].

5 Der Use Case

5.1 Problemstellung

Es ist der swisstopo schon lange ein Anliegen, dass die Publikationvorgänge von Geodaten optimiert werden sollen. Wann immer möglich, soll der Automatisierungsgrad erhöht werden.

Besonders aufwändig erweist sich zurzeit die Publikation von 3D Daten. Die manuelle Publikation der 3D Daten benötigt eigene Tools, die auf einem performanten und somit teuren Rechner laufen müssen. Ausserdem erfordert die Bereitstellung einen hohen Koordinationsaufwand zwischen der Infrastruktur und der Entwicklung. Dabei passiert es, dass Mängel in den 3D Daten erst nach beendeter Webpublikation bemerkt werden; und der ganze Publikationsvorgang muss wieder von vorne gestartet werden. Auch dem Hersteller der 3D Daten (dem Bereich Topografie) wäre es ein Anliegen, wenn er diese Daten selbst automatisch publizieren und prüfen könnte.

Einerseits soll in dieser Arbeit mittels Prototyp aufgezeigt werden, wie der Automatisierungsgrad erhöht werden könnte. Andererseits soll untersucht werden, ob für die Verarbeitung Budget Instanzen¹² anstelle von On-Demand Instanzen¹³ verwendet werden können und wie deren Einsatz aussehen könnte.

5.1.1 Budget Instanzen

Amazon preist Budget Instanzen folgendermassen an: *"Mit Amazon EC2 Spot-Instances können Sie die Vorteile nicht genutzter EC2-Kapazitäten in der AWS Cloud nutzen. Spot-Instances sind mit einem Rabatt von bis zu 90% im Vergleich zum On-Demand-Preis verfügbar. Sie können Spot-Instances für diverse statuslose, fehlertolerante und flexible Anwendungen verwenden. Dazu zählen unter anderem Big-Data-Anwendungen, auf Containern ausgeführte Workloads, CI/CD, Webserver-Anwendungen, HPC-Anwendungen (High-Performance Computing) sowie Test- und Entwicklungs-Workloads."* [10a].

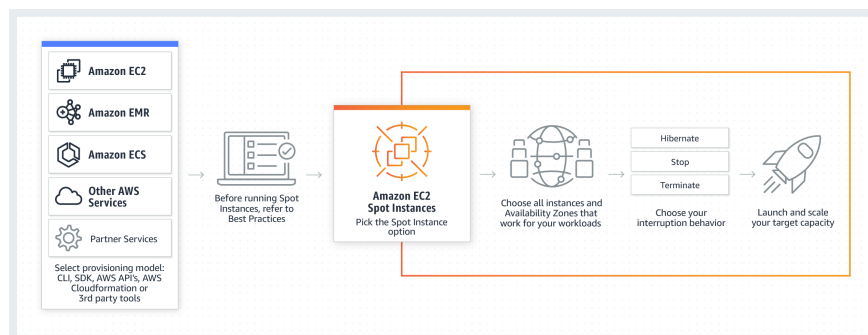


Abbildung 4: So funktioniert das Ausleihen von Budget (SPOT) Instanzen

Das Verkaufsargument 90% Rabatt ist eine Ansage: Eine Preis-Aktion, ein Budget Produkt, damit AWS nicht genutzte EC2-Kapazitäten doch noch verkaufen kann. Der Konsument gibt viel weniger aus, mit dem Hacken, dass einem die Instanz innerhalb von 2 minütiger Vorankündigung weggenommen werden kann.

Möchte man die EC2 Spot Instanzen für die Geodatenverarbeitung einsetzen, muss also ein Weg gefunden werden, um mit diesen Unterbrüchen umgehen zu können.

Wie auf der Abbildung 4 dargestellt, kann das Verhalten der Instanz bei einem Interrupt¹⁴ definiert werden. Es besteht dabei sogar die Möglichkeit eines *Hibernate*, dass der Zustand des RAMs auf der Festplatte persistiert

¹²Amazon Spot Instanzen

¹³Herkömmliche EC2 Instanzen.

¹⁴Wenn Amazon die Spot Instanz für einen anderen Zweck beanspruchen möchte, die wegnimmt.

wird.

Diese 2 minütige Vorankündigung eines Interrupts kann auch via RESTful Abfrage¹⁵ von der Instanz aus abgefragt werden.

Diese Vorankündigung lässt sich auch via AWS CLI abfragen. In einem grösseren Setup könnte das Signal der Vorankündigung auch mit dem AWS Überwachungstool *CloudWatch* verarbeitet werden.

5.2 Ist-Zustand der 3D Datenpublikation

5.2.1 3D Datenpublikation

Das Aufzeigen des Ist-Zustandes der 3D Datenpublikation soll helfen sich einen Überblick, eine Ausgangslage, zu verschaffen. Es bildet die Grundlage, um die Frage zu beantworten, welche Arbeiten erledigt werden müssen, um die 3D Daten im Web zu publizieren? Welche Arbeitsschritte könnten automatisiert werden?

Eine Datenpublikation läuft folgendermassen ab: Sobald der Auftrag für eine 3D Datenpublikation erteilt wurde¹⁶, müssen zurzeit folgende Schritte, die in der Abbildung 6 referenziert sind, erledigt werden:

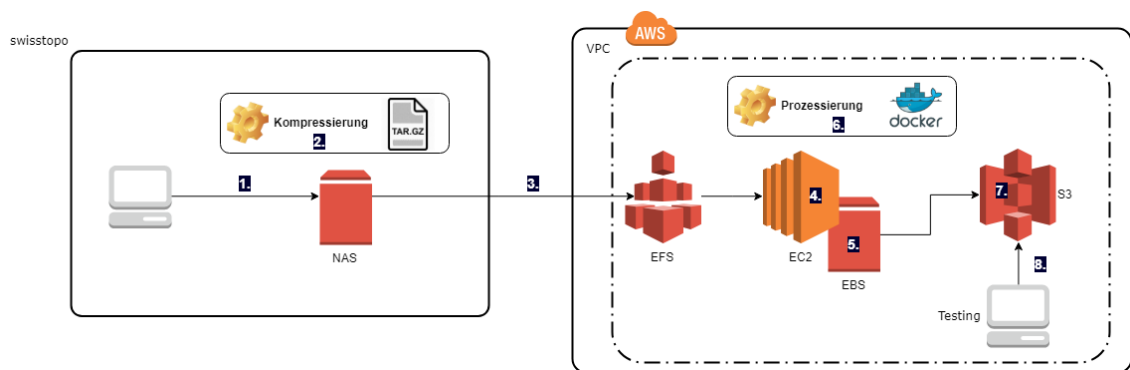


Abbildung 5: Arbeitsschritte, die es braucht, um die 3D Daten zu Publizieren.

1. Die Rohdaten¹⁷ werden vom Auftraggeber (Bereich Topografie) auf einem NAS bereitgestellt.
2. Da es sich um Millionen Dateien handelt, werden diese je Kartenblatt¹⁸ erst einmal gezippt, um so (weil bedeutend weniger Dateien) schneller kopiert werden zu können.
3. Kopieren der gezippten Dateien vom swisstopo Netzwerk in die Amazon Cloud (AWS VPC¹⁹) kopiert²⁰.
4. Parallel dazu wird die IT via Ticket gebeten einen Server mit dem entsprechenden Image²¹ bereitzustellen.
5. Kopieren und Entzippen der Rohdaten auf die gemountete Festplatte²² des Servers.
6. Die Daten auf dem Server via Docker verarbeiten²³.
7. Kopieren der Web-optimierten Daten auf S3.

¹⁵EC2 Spot Instance Interruption Notice, eine HTTP Abfrage: Im Anhang D.2 hat es ein Beispiel dazu.

¹⁶Vom Bereich Topografie.

¹⁷Das Format der Rohdaten ist KML/COLLADA (beides XML).

¹⁸Blatteinteilung: Die swisstopo arbeitet viel nach Blatteinteilung nach Kartenblättern, hier ein Beispiel: map.geo.admin.ch.

¹⁹AWS: Amazon Web Services, VPC: Virtual Private Network. Ein Kopieren.

²⁰Via *rsync*.

²¹Eine EC2 Instanz eine *m4.10xlarge* aus einem bereits vorhandenes AMI.

²²Ein EBS Volumen.

²³Umwandeln in das Web-Format *Cesium3DTiles*.

8. Die Daten visualisieren, um inhaltlich testen zu können. Ein Codepen Projekt²⁴, dass auf die 3D Tiles zugreift.

5.2.2 Aufwand der Verarbeitung

Folgende Schritte sind besonders aufwendig:

- ▶ **Abb. 6, Schritt 2. und 5.:** Das Kopieren / Zippen (Packen und Entpacken) der Rohdaten dauert lange.
- ▶ Es kommt immer mal wieder vor, dass Daten korrupt sind, was zu einer Nachlieferung führt, mit der Gefahr, dass es mit Versionen der Lieferung zu einem Durcheinander kommen könnte.
- ▶ **Abb. 6, Schritt 4.:** Unsere IT muss für die Verarbeitung eine EC2 Instanz mit EFS bereitzustellen. Um laufende Kosten zu verringern, wird diese Instanz nach getaner Arbeit²⁵ wieder gestoppt. Falls mit den Daten etwas nicht in Ordnung ist, muss dieser Schritt von der IT wiederholt werden. Nebst der Bemühung der IT, muss auf der Instanz selber anschliessend das eine und andere manuell installiert und konfiguriert werden.

5.2.3 Technische Komponenten

Auflistung der technischen Komponenten:

- ▶ **Abb. 6, Schritt 2., 3. und 5.:** Das Komprimieren und Kopieren der Rohdaten erfolgt mit Linux Bordmitteln (*cp, rsync, tar*)
- ▶ **Abb. 6, Schritt 4. und 7.:** Erfolgen via AWS CLI
- ▶ **Abb. 6, Schritt 6.:** Via Docker. Der Container wurde von der Firma Analytical Graphics Inc. [10b] bereitgestellt. Das Tool, das die Rohdaten in ein Web-Format umwandelt, wird mit Node.js ausgeführt.
- ▶ **Abb. 6, Schritt 7.:** Ein Projekt, um die Daten im Browser betrachten und inhaltlich Testen zu können, erfolgt über die Webseite codepen.io.

²⁴SaaS: Eine Webseite, um Front-End Code zu schreiben, zu testen, und bereitzustellen (codepen.io).

²⁵Der Verarbeitung.

6 Architektur

6.1 Analyse des Ist-Zustandes

6.1.1 Bereitstellung der Rohdaten und Sicherstellung dessen Qualität

Basierend auf der Aufwandeinschätzung des Ist-Zustandes des Kapitels 5.2.2 geht hervor, dass vor allem das Bereitstellen der Rohdaten und das Sicherstellen dessen Qualität aufwändig ist.

Das Bereitstellen der Rohdaten nimmt aufgrund der Datenmenge²⁶ viel Zeit in Anspruch. Hier gäbe es folgende zwei Lösungsansätze:

1. Der Datenlieferant könnte direkt aufs EFS schreiben (Abb. 6, Nr. 5)
2. Automatischer Prozess via Cronjob oder oder Trigger, der die Kopier-Schritte (Abb. 6, Schritte Abb. 6) regelmässig bereitstellt.

Wobei der Lösungsansatz simpler und somit weniger fehleranfällig zu sein scheint. Dieser Ansatz kann, sobald die swisstopo eine Hybrid Cloud hat, weiterverfolgt werden.

Sicherung der Qualität: Weil es dem Lieferanten an Tools fehlt, um die bereitgestellten Daten inhaltlich zu Prüfen, werden Fehler häufig erst nach der Publikation entdeckt. Dies sicherlich auch, weil die Daten im Web an ein breites Publikum gelangen. Da keine inhaltliche Prüfung der Rohdaten gemacht werden kann, wäre es sicher hilfreich, wenn der Datenlieferant die Iterationen der Publizierung gleich selber machen könnte und dann nur noch Bescheid gibt, wenn sie seines Erachtens in Ordnung sind.

6.1.2 Bereitstellung der Infrastruktur für die Verarbeitung

Die Rohdaten werden via Docker Image verarbeitet. Jedes Mal wenn das Image für die Verarbeitung ausgeführt werden soll, muss die IT gebeten werden die nötige Infrastruktur bereitzustellen. Idealerweise findet sich eine Lösung, die diesen Schritt überflüssig macht.

6.2 Bewertungskriterien

| | |
|-------------------------------|---|
| Kosten einsparen | Änderungen des Prozesses sollen nicht teurer sein, als die bisherige Lösung |
| Automatisierbar | Die Lösung soll den Grad der Automatisierung möglichst weit vorantreiben und dadurch so wenig Personalaufwand wie möglich beanspruchen. |
| Einfach | Ein neuer Mitarbeiter soll die Lösung rasch verstehen und warten können. |
| Bestehende Technologie | Obwohl der Anwendungsfall anders ist, soll der Technologie Stack möglichst demjenigen der Microservices entsprechen. |

²⁶Weil es sich um mehrere Millionen Dateien handelt.

6.3 Architekturentscheid

Für den Architekturentscheid wurde auf eine Entscheidungsmatrix verzichtet. Dies Aufgrund der grossen Auswahl und der vielen Entscheidungen, die getroffen werden mussten. Die oben erwähnten Kriterien wurden in der Entscheidungsfindung mit einbezogen und weitere Argumente werden ausgeführt:

6.3.1 Verarbeitung auf Spot Instanzen

Der Autor ist auf drei Möglichkeiten gestossen, um Geodaten mit AWS Spot Instanzen zu verarbeiten: *Direkt auf der Spot Instanz*, *Kubernetes (EKS)* und *AWS Batch*. Gerne hätte der Autor alle drei Lösungen als POC weiterverfolgt, aber die Zeit dazu reichte leider nicht. Aus diesem Grund beschränkt sich der Autor lediglich auf eine Beschreibung der drei Möglichkeiten. Wobei die erste Möglichkeit als POC umgesetzt wurde.

Direkt auf einer Spot Instanz: Diese Möglichkeit bedingt nur geringe Anpassungen der bisherigen Verarbeitung, weil die Daten im Prinzip wie bisher auf EC2 Instanz verarbeitet werden; mit dem Nachteil, dass diese jederzeit durch eine andere Instanz ersetzt werden könnte.

Der Autor hat sich bei der Auswahl der Images²⁷ für *Ubuntu Server 18.04 LTS* entschieden. Dies weil die Verarbeitung bisher auf Ubuntu gelaufen ist und weil er keinen Grund sieht, etwas Funktionierendes zu ändern²⁸. Dasselbe Argument gilt für die Wahl der Dimensionierung des Servers: Ein Server mit 200 GB SSD Festplatte, 16 CPUs und 60 GByte Arbeitsspeicher konnte die Verarbeitung der 3D Daten²⁹ gut stemmen.

Bei der Provisionierung³⁰ des Servers wurde der *"Cloud-init"* Ansatz³¹ gewählt: Dieser Ansatz provisioniert den Server der Startup Phase. Dies hat gegenüber einem eigenen AMI den Vorteil, dass die Spezialisierung³² in einer Textdatei festgehalten wird und somit jederzeit nachvollziehbar ist. Ausserdem muss bei einem Update des Basis Images kein neues Image gebaut werden. Die Initialisierungsskript wurde auf ein Minimum beschränkt und die eigentliche Initialisierung wurde an *Ansible* übergeben.

Kubernetes (EKS) auf Spot Instanzen erweitern TODO: Weil die Microservices in Zukunft auf EKS laufen werden, wäre es sicher ein Möglichkeit, abzuklären, inwiefern sich eine Datenverarbeitung in dieser Umgebung einrichten lässt. In der Literatur hat der Autor folgende Elemente gefunden.

- ▶ Job
- ▶ CronJob
- ▶ Eigenes ReplicaSet?

EBook auf meinem Reader: Job und CronJob aber auch Kubernetes on AWS S. 88

AWS Batch auf Spot Instanzen Diese Möglichkeit wäre wohl die Naheliegendste, weil AWS Batch für genau diese Art von Aufgaben gebaut wurde. Im Internet hat der Autor einen schönes Beispiel gefunden, wie sich AWS Batch einsetzen lässt. <https://aws.amazon.com/de/blogs/compute/creating-a-simple-fetch-and-run-aws-batch-job/>

²⁷AMI: Amazon Machine Image.

²⁸Security Maintenance bis 2028 [20d].

²⁹Gebäude, Bäume und Namen.

³⁰Bereitstellung

³¹Eigentlich ein Bash Skript. Unter AWS als *User Data* bezeichnet.

³²Das Einrichten.

7 Realisierung des Prototyps

Als Umgebung, um die Infrastruktur für den Prototypen bereitzustellen, hat der Autor das Kennenlernangebot von AWS, ein einjähriges kostenloses Kontingent an Services und Produkten [20a], gewählt.

Nachdem der erwähnte *AWS Account* erstellt war, mussten erste Schritte, wie grundlegende Konfigurationen des Identity Access Management (IAM) gemacht und das Steuern von Spot Anfragen, das Einrichten eines Container Repositories (ECR), das Aufsetzen eines öffentlich zugänglichen S3 Buckets und anderes aus dem AWS Produktkatalog, erlernt werden.

Weitere Schritte folgten. Die eigentliche Realisierung des Prototypen wird in den folgenden Kapiteln beschrieben werden. Das Endresultat wird jedoch der Nachvollziehbarkeit halber schon einmal hier abgebildet:

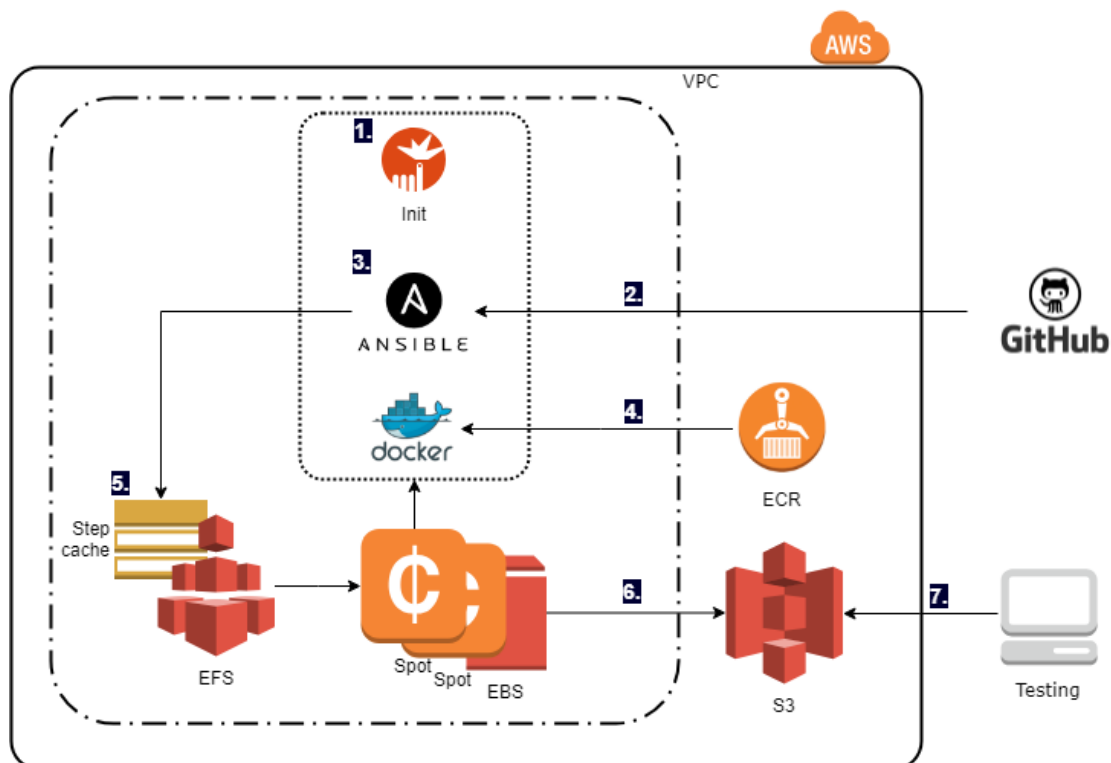


Abbildung 6: Geodatenverarbeitung mit SPOT Instanzen.

7.1 Die Datenverarbeitung als Code

Mit der Entscheidung, die Spot Instanz via *Cloud-Init*-Ansatz (Abb. 6, Nr. 1) mit Ansible zu Provisionieren³³, lag es auf der Hand, auch gleich dieselbe Technologie für die Realisierung der Publizierung zu verwenden (Nr. 3). Viele Annehmlichkeiten, wie die Nähe zur Shell und Python, das Logging und deklarativer Code zeigten sich in der Realisierung als hilfreich. Das Init-Skript wie auch das Ansible Playbook können auf github.com/bfh-semesterarbeit eingesehen werden.

³³Und nicht mit einem neuen AMI.

7.2 Spot Flottenanfrage

Als Setup für die Spot Flottenanfrage wurde ein sogenanntes Launch Template mit den wichtigsten Konfigurationen wie Sicherheitsgruppe, Wahl des AMI, die 200 GB grosse SSD Festplatte und User Data angelegt. Anschliessend wurde eine Spot Flottenanfrage basierend auf diesem Template erstellt. Wobei als Mindestanforderung an den Rechner 16 CPUs und 60 GB Memory waren. Als Zielkapazität wurde eine Instanz gewählt, mit der Option, dass die Zielkapazität aufrechterhalten bleiben soll: Somit wird nach jedem Interrupt automatisch eine neue Instanz mit der definierten Konfiguration zur Verfügung gestellt.

7.3 Handling der Interrupts

Wie im Kapitel 5.1.1 beschrieben, sind Spot Instanzen so günstig, weil sie jederzeit einem besser zahlenden Kunden zur Verfügung gestellt werden können. Also muss der Publikationsvorgang mit dem Handling von Interrupts umgehen können. Der Einfachheit halber hat der Autor gänzlich auf das *Horchen eines bevorstehenden Interrupts* via RESTful Abfrage (In Anhang D.2 beschrieben) oder *CloudWatch* verzichtet und hat stattdessen die Publizierung als Serie betrachtet und in Schritte unterteilt. Sobald ein Schritt erledigt wird, wird dies auf dem EFS Volumen in einer Textdatei festgehalten. Falls es zu einem Interrupt kommen sollte, wird von der Spot Flotte die nächste Instanz bereitgestellt und die Publizierung macht bei dem Schritt weiter, der zuletzt in der Textdatei festgehalten wurde (Abbildung 6, Nr. 5).

7.4 Testen der Datenstruktur

7.5 Logging

7.6 Inhaltliche Kontrolle der publizierten Daten

Das Resultat kann vorläufig³⁴ hier betrachtet werden: <https://codepen.io/rebert/pen/ExKZmmE> Wobei nur die Gebäude vom S3 Testbucket kommen (Abb. 6, Nr. 7)

7.7 Testen

7.7.1 Unerwartete Interrupts

Mit einem Test

7.7.2 Full-load test

Der komplette Rohdatensatz, alle (erfassten) Gebäude der Schweiz, wurde verarbeitet.

³⁴Bis am 1. November 2020

8 Evaluation

8.1 Erfahrungen

Die Anregungen aus dem Studium³⁵ und das günstige Angebot von AWS haben den Autor motiviert, die komplette Infrastruktur für den Prototypen selber in einem eigenen AWS Testaccount aufzubauen. Da die Infrastruktur und die Verarbeitung deklarativ festgehalten wurden, kann ein Grossteil dieses Codes auch im Account des Betriebes ohne grosse Anpassungen wiederverwendet werden.

Im Betrieb hätte das Team und die IT Abteilung das nötige Know-How gehabt, um mit dieser Arbeit weiter zu kommen, als es jetzt der Fall ist. Aber der Autor hat sich bewusst dazu entschieden, die Arbeit möglichst selbstständig zu realisieren und diese Ressource zu schonen, indem er sie so wenig wie möglich genutzt hat. Es war für den Autor schon nur eine Hilfe, zu Wissen, dass die Möglichkeit einer Unterstützung da war. Dies führte dazu, dass der Autor Momente der Ratlosigkeit erleben musste. Was nicht weiter schlimm war, da *"Perplexity is the beginning of knowledge"* [AD19, S. 33].

Bei der Auswahl der Spot-Instanz Aufgabe hat der Autor *Flexible Workloads* gewählt. Der Autor war darüber erstaunt, dass ihm während der ganzen Entwicklungs- und Testphase nicht ein einziges Mal eine Instanz weggenommen wurde. Während der Entwicklung wurde die Spot Flotte aufgrund eines EBS-Volumens auf eine bestimmte Availability-Zone beschränkt. Und nicht einmal mit dieser Einschränkung ist es zu einem Interrupt gekommen. Um das Verhalten bei Interrupts testen zu können, mussten die Spot-Instanzen somit mutmasslich terminiert werden.

8.2 Wirtschaftlichkeit

Personalstunden Der Grad der Automatisierung konnte erheblich erhöht werden. Dadurch fallen Personalstunden³⁶ weg. Vor allem diese Kosten rechnen sich. Die Verarbeitungsschritte sind im Code abgebildet, was die Fehleranfälligkeit kleiner macht, als wenn Kommandos aus der Prozessdokumentation kopiert werden müssen. Um die IT Abteilung gänzlich zu Entlasten, müsste noch eine Rolle eingerichtet werden, die eine vordefinierte Spot Flottenanfrage steuern kann.

Einsparungen Spot im Vergleich zu On-Demand Um die Kosten von On-Demand mit Spot Instanzen zu vergleichen, werden hier die Ergebnisse des Prototyps aufgelistet. Verarbeitung der Gebäude mit der Mindestanforderung: 16 CPUs und 60 GByte Memory. Die Gesamtrechnzeit war ca. 18 Stunden und es konnten 76% der Kosten eingespart werden.

| Rechenzeit | On-Demand | Spot |
|-------------|-----------|---------|
| Pro Stunde | 1.19 \$ | 0.29 \$ |
| Total: 18 h | 21.42 \$ | 5.22 \$ |

Tabelle 1: Relativ betrachtet ist das Sparpotential enorm: 76%.

Schaut man die Kosten relativ an, ist das Sparpotential enorm: 76%! In absoluten Zahlen erscheint das Sparpotential für einen einzelnen Verarbeitungsauftrag nicht riesig: ca. 15 \$. Dazu muss allerdings ergänzt werden, dass die Verarbeitungszeit durch die Automatisierung verkürzt wurde. Ausserdem werden jährlich mehrere 3D Updates in Auftrag gegeben. Der aufwändigste Auftrag davon ist das Update des 3D Terrains, das eine wesentlich grössere Instanz über eine längere Zeitspanne³⁷ braucht.

³⁵CAS in Cloud Computing an der BFH Bern: Bsp. die Vorlesungen Architektur, IaaS, PaaS, Docker und Kubernetes.

³⁶In unserem Team.

³⁷Ca. 1 Woche.

8.3 Kritische Punkte

8.3.1 Security

Bezüglich IAM hat der Prototyp noch technische Schulden. Dem Autor ist bewusst, dass folgende zwei Sicherheitslücken bezüglich Zugängen vorhanden sind:

github.com: Der Einfachheit halber wurde das Github Repository, auf welchem sich das Init-Skript und das Ansible Playbook befinden, öffentlich zugänglich gemacht (Abb. 6, Nr. 2). Dies ist nicht weiter problematisch, da der Code auf Github keine Passwörter etc. bekannt gibt. Aber es sind darin Informationen über die AWS Infrastruktur enthalten. Sauber wäre eine Implementation, die Github Credentials eines privaten Repositories verwaltet. Geeignet dazu wäre z.B. der AWS Secrets Manager [20b].

AWS Key für S3: Eigens für das Kopieren auf S3 wurde eine IAM Rolle angelegt, die nicht mehr darf, als auf S3 zu kopieren (Abb. 6, Nr. 6). Die Keys dazu wurden auf dem EFS-Volumen abgelegt. Dies ist sicher nicht ideal. Zwar ist die Sicherheitsgruppe des EFS ist zwar so konfiguriert, dass nur die Sicherheitsgruppe der EC2 Instanzen darauf Zugriff haben sollten, aber dennoch sollten keine unverschlüsselten Keys auf dem Filesystem herumliegen. Um diese Sicherheitslücke zu schliessen, könnte eigens dafür eine IAM Rolle für die EC2 Instanz angelegt werden [20c].

Sollte der Prototyp in den Betrieb überführt werden, würde sowieso das IAM der swisstopo übernommen werden müssen; und dieses Kapitel hätte sich erübrigt.

8.3.2 Datenverarbeitung als Blackbox

Idealerweise wird die Datenverarbeitung in kleine parallelisierbare und in sich selber geschlossene Schritte unterteilt. Nach einem Interrupt der Instanz können dann die noch nicht abgeschlossenen Schritte fortgeführt werden.

Beim gewählten Use Case der 3D Datenverarbeitung handelt es sich um eine Blackbox in einem Container, die entweder Alles oder Nichts verarbeitet. Was in diesem Fall alle (erfassten) Gebäude der Schweiz bedeutet. Damit der Prototyp funktioniert, müssen die Spot Instanzen mindestens 10 Stunden ohne Interrupt laufen. Zum Glück verhalten sich die Spot Instanzen in der Regel stabil genug, um die einzelnen langwierigen Schritte seriell verarbeiten zu können.

Der Test, der die Rohdaten auf well-formed XML testet, hilft Fehler, vor dem langwierigen Verarbeiten in der Blackbox, abzufangen. Jedoch leider nicht alle Fehler. Wenn man nun davon ausgeht, dass es in den ca. 8 Millionen Gebäuden, die geliefert wurden, immer mal wieder fehlerhafte Daten hat, welche die Blackbox ins Straucheln bringen, dann kann die 3D Datenverarbeitung trotz aller Automaistation zu einem mühseligen Unterfangen werden.

9 Ausblick

Bald steht das nächste 3D Update vor der Tür. Bei dieser Gelegenheit könnte der Automatisierungsteil dieser Semesterarbeit übernommen werden. Da es sich dabei um ein Ansible Playbook in einer deklarativen Sprache handelt, sind die einzelnen Verarbeitungsschritte auch gleich dokumentiert. Anpassungen sind einfach zu machen. Falls diese Nachführung auf einer herkömmlichen On-Demand EC2 Instanz gemacht werden müsste, könnte das Skript dennoch verwendet werden.

Mittelfristig wäre es schon nur aus Gründen des Kostensparens interessant, wenn rechenintensive Geodatenverarbeitungen auf Spot Instanzen laufen könnten. Hier nochmals: Relative Einsparungen von mehr als 70%, was sich mit der Zeit auch in absoluten Kosteneinsparungen (\$) sehen lassen würde.

Um den Koordinationsaufwand mit der IT zu verringern, könnte die Person, die eine Spot Instanz braucht, mit den nötigen Rechten versehen werden: Bsp. könnte sie, wie bei einem Load-Balancer, die Anzahl Instanzen der Spot Flotte von 0 auf 1 ändern dürfen, um nach getaner Arbeit die Anzahl Instanzen wieder auf 0 zu setzen.

Sobald die swisstopo über eine Hybrid Cloud verfügt, könnte die Topografie neue Daten direkt auf das EFS schreiben. Dieser Event könnte getriggert werden, um eine Spot Instanz für die 3D Geodatenverarbeitung zu starten. Jeweils ca. 18 Stunden später wären die Daten im Web und die Topografie könnte diese auf der Testumgebung einsehen (Abb. 6, Nr. 7). Diesen Vorgang könnte die Topografie solange wiederholen, bis sie mit dem Resultat zufrieden ist.

Ohne Hybrid Cloud müssten die Daten immer noch vom Intranet in die AWS VPC kopiert werden. Auch dieser Vorgang liesse sich einfach automatisieren.

Persönlich hat dem Autor die Tour mit Spot Instanzen Spass gemacht. Die bei der Tour gemachten Erfahrungen waren ein guter Einstieg, um Services der AWS Cloud besser verstehen und nutzen zu können. Es wird sicher nicht bei dieser Tour bleiben.

Literaturverzeichnis

- [10a] (2010). Amazon EC2-Spot-Instances, Adresse: <https://aws.amazon.com/de/ec2/spot/> (besucht am 08.07.2010).
- [10b] (2010). Analytical Graphics Inc., Adresse: <https://www.agi.com> (besucht am 09.08.2020).
- [10c] (2010). Informatik, Adresse: <https://de.wikipedia.org/wiki/Informatik> (besucht am 16.07.2010).
- [10d] (2010). Kanban (Softwareentwicklung), Adresse: [https://de.wikipedia.org/wiki/Kanban_\(Softwareentwicklung\)](https://de.wikipedia.org/wiki/Kanban_(Softwareentwicklung)) (besucht am 19.07.2010).
- [10e] (2010). swisstopo Onlineshop, Adresse: <https://shop.swisstopo.admin.ch> (besucht am 17.07.2020).
- [10f] (2010). Technische Schulden, Adresse: https://de.wikipedia.org/wiki/Technische_Schulden (besucht am 18.07.2010).
- [18] (2018). Diese 3D-Landkarte gibt dir einen völlig neuen Blick auf die Schweiz, Adresse: <https://www.watson.ch/digital/schweiz/674619561-diese-3d-landkarte-gibt-dir-einen-voellig-neuen-blick-auf-die-schweiz> (besucht am 18.07.2010).
- [20a] (2020). AWS Free Tier, Adresse: <https://aws.amazon.com/de/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc> (besucht am 21.08.2020).
- [20b] (2020). AWS Secrets Manager, Adresse: <https://aws.amazon.com/secrets-manager/> (besucht am 28.08.2020).
- [20c] (2020). IAM roles for Amazon EC2, Adresse: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html?icmpid=docs_ec2_console (besucht am 28.08.2020).
- [20d] (2020). The Ubuntu lifecycle and release cadence, Adresse: <https://ubuntu.com/about/release-cycle> (besucht am 28.08.2020).
- [AD19] J. Arundel und J. Domingus, *Cloud Native DevOps with Kubernetes*. O'RELLY, 2019.
- [BJB06] T. Balstrøm, O. Jacobi und L. Bodum, *GIS og geodata*. Forlaget GIS og Geodata, 2006.
- [Chr20] H. Christ. (9. Juli 2020). swisstopo Fallstudie, Adresse: <https://aws.amazon.com/de/solutions/case-studies/swisstopo-hpc/>.
- [Keh05] D. Kehlmann, *Die Vermessung der Welt*. Rowohlt Verlag GmbH, 23. Sep. 2005, 304 S., ISBN: 3498035282. Adresse: https://www.ebook.de/de/product/3563778/daniel_kehlmann_die_vermessung_der_welt.html.
- [SE04] M.-H. Schertenleib und H. Egli-Broz, *Grundlagen Geografie: Aufgaben des Fachs, Erde als Himmelskörper und Kartografie : Lerntext, Aufgaben mit Lösungen und Kurztheorie*. Zürich: Compendio Bildungsmedien, 2004, ISBN: 3715591714.
- [Sta10] R. Stallman. (2010). What is Free Software?, Adresse: <http://www.gnu.org/philosophy/free-sw.html> (besucht am 28.08.2020).
- [Web10] Webpage. (2010). Keep calm and make your choice..., Adresse: <https://www.grenke-40-one.de/excursions-detail/> (besucht am 11.07.2010).

A Fachbegriffe und Abkürzungen

| Abkürzung | Definition |
|----------------------------|---|
| Ansible | Ein Open-Source Automatisierungs-Werkzeug zur Orchestrierung und allgemeinen Konfiguration und Administration von Computern. |
| AMI | Amazon Machine Images: Images für virtuelle Server. |
| Availability Zone | Jeder Amazon Rechenzentrumstandort (Region) besteht aus mehreren isolierten Zonen, den Availability Zones. |
| AWS | Amazon Web Services. |
| AWS CLI | Das Command Line Interface, um AWS Ressourcen zu verwalten. |
| BGDI | Bundesgeodateninfrastruktur. |
| Budget Instanzen | Im Kontext dieser Arbeit ein Synonym für AWS Spot Instanzen. |
| EBS | Block Storage: Speicher (für eine Instanz). |
| EC2 | Amazon Elastic Compute Cloud: Rechenkapazität, Speicher (und mehr). |
| EFS | Amazon Elastic File System: Cloud NFS-Dateisystem. |
| Hybrid Cloud | Eine Computerinfrastruktur, die Public Cloud und Private Cloud kombiniert nutzt. |
| IaaS | Cloud Infrastructure as a Service: Infrastruktur <i>"Pay as you go"</i> beziehen. |
| IAM | Identity and Access Management: Verwaltung der Zugänge und der dazugehörenden Rechte. |
| On-Demand Instanzen | Herkömmliche EC2 Instanzen. Der Begriff wird in dieser Arbeit manchmal verwendet, um von EC2 Spot Instanzen unterscheiden zu können. |
| Kartenblatt | In der swisstopo wird viel in der Einheit von Kartenblättern gearbeitet, welche die ganze Schweiz abdecken. Hier ein Beispiel: map.geo.admin.ch . |
| PaaS | Plattform as a Service. |
| POC | Proof of Concept: Die Machbarkeit eines Produktes oder einer Idee aufzeigen. |
| S3 | Amazon S3 (Simple Storage Service): Ein Filehosting-Dienst dessen Zugriff über HTTP/HTTPS erfolgt. |
| SaaS | Software as a Service. |
| SSD | Solid-State-Disk - Halbleiterlaufwerk: Festplatte ohne bewegliche Teile, dafür mit kurzen Zugriffszeiten. |
| SSH | Secure Shell: Netzwerkprotokoll, um auf einen entfernten Rechner zuzugreifen. |
| S3 | Simple Storage Service: Ein Objektspeicherservice von Amazon. |

Tabelle 2: In der Arbeit verwendete Fachbegriffe und Abkürzungen

B Kanban

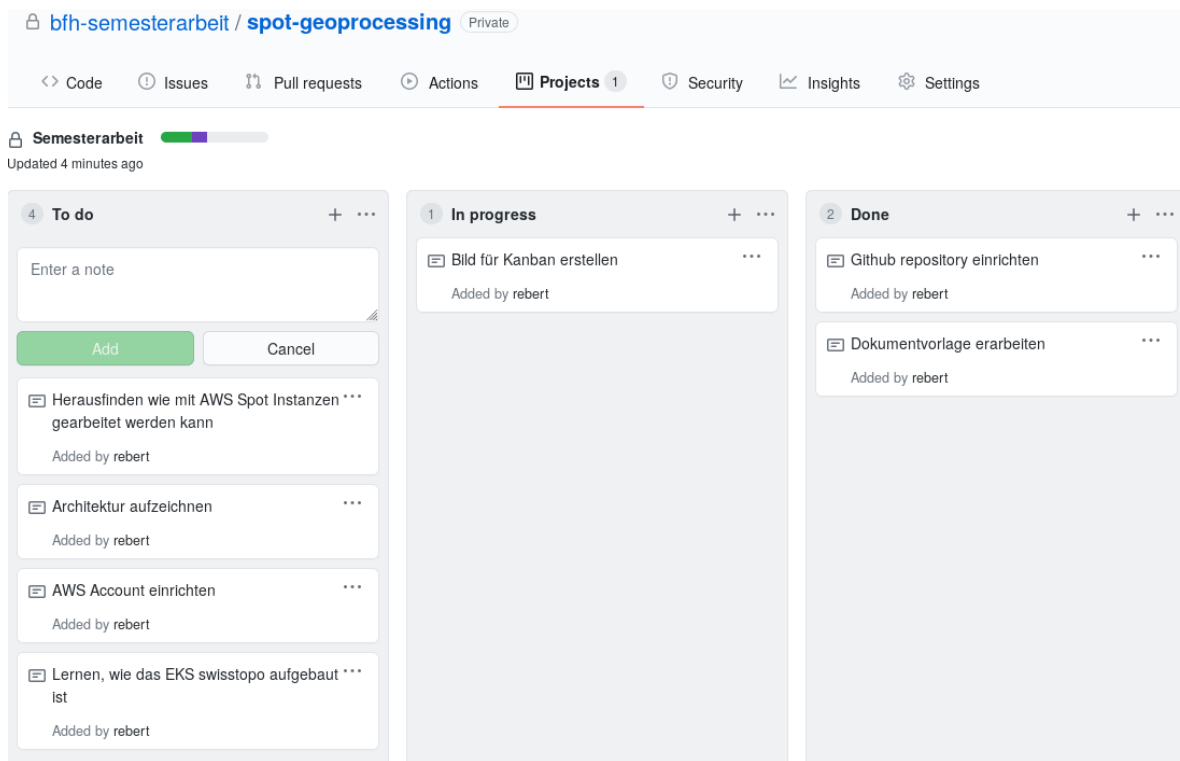


Abbildung 7: Klassisches Kanban auf *github.com*

C Projektplan

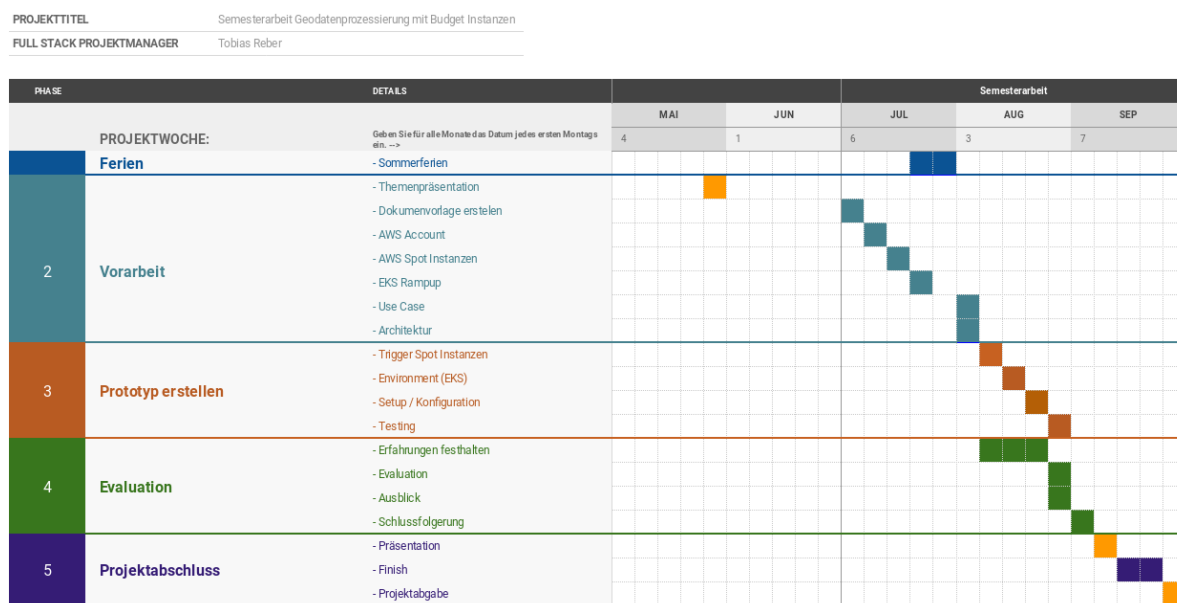


Abbildung 8: Projektplan. Die orangenen Meilensteine wurden von der BFH vorgegeben.

D Konfigurationen und Kommandos

D.1 EFS auf EC2-Instanz mounten

Anhand einer Anleitung, einem sogenannten Walkthrough, wurde via AWS CLI³⁸ ein EFS an eine EC2-Instanz gemountet und die Rohdaten wurden schon einmal dorthin kopiert. In der Verarbeitung bildet dieses EFS den Ausgang der Datenverarbeitung.

Listing 1: EFS auf EC2-Instanz mounten.

```
#!/bin/bash
# from https://docs.aws.amazon.com/efs/latest/ug/wt1-create-ec2-resources.html

# security group 4 ec2
aws ec2 create-security-group \
--region eu-west-1 \
--group-name efs-dataprocessing1-ec2-sg \
--description "Amazon EFS dataprocessing 1, SG for EC2 instance" \
--vpc-id vpc-87ad55fe

# security group 4 efs
aws ec2 create-security-group \
--region eu-west-1 \
--group-name efs-dataprocessing1-mt-sg \
--description "Amazon EFS dataprocessing 1, SG for mount target" \
--vpc-id vpc-87ad55fe

# access to ec2 instance group from everywhere
aws ec2 authorize-security-group-ingress \
--group-id sg-098669727548dcedd \
--protocol tcp \
--port 22 \
--cidr 0.0.0.0/0 \
--region eu-west-1

# describe security group
aws ec2 describe-security-groups \
--region eu-west-1 \
--group-id sg-098669727548dcedd

# access to efs storage from ec2 group
aws ec2 authorize-security-group-ingress \
--group-id sg-02778494bc39601d4 \
--protocol tcp \
--port 2049 \
--source-group sg-098669727548dcedd \
--region eu-west-1

# get subnet id
aws ec2 describe-subnets \
--region eu-west-1 \
--filters "Name=vpc-id,Values=vpc-87ad55fe"

# run ec2 instance
aws ec2 run-instances \
--image-id ami-047bb4163c506cd98 \
--count 1 \
--instance-type t2.micro \
--associate-public-ip-address \
--key-name bfh_root.pem \
--security-group-ids sg-098669727548dcedd \
--subnet-id subnet-f66512ac \
--region eu-west-1
```

³⁸AWS Command Line Interface: Ein kommandozeilenorientiertes Werkzeug.


```
aws ec2 describe-instances \
--instance-ids i-09cb26ed64cdde683 \
--region eu-west-1

# ec2-54-75-53-151.eu-west-1.compute.amazonaws.com
# EFS =====
aws efs create-file-system \
--creation-token FileSystemForDataprocessing1 \
--tags Key=Name,Value=Dataprocessing1 \
--region eu-west-1

# create mount target
aws efs create-mount-target \
--file-system-id fs-5aceld90 \
--subnet-id subnet-f66512ac \
--security-group sg-02778494bc39601d4 \
--region eu-west-1

# On instance amazon linux (ec2-user)
sudo yum -y update
sudo reboot # dont know why... like windows
sudo yum -y install nfs-utils

# On instance ubuntu (ubuntu)
sudo apt-get update
sudo apt-get install nfs-common

mkdir ~/efs-mount-point
cd ~/efs-mount-point
sudo chmod go+rw .

sudo mount -t nfs \
-o nfsvers=4.1,rsz=1048576,wsz=1048576,hard,timeo=600,retr=2,noresvport
fs-5aceld90.efs.eu-west-1.amazonaws.com:/ \
~/efs-mount-point

# CP from home to ec2 instance
scp -i bfh_root.pem \
/media/saibot/vortrag/bfh/buildings/*.tar \
ec2-user@54.75.53.151:/home/ec2-user/efs-mount-point/buildings/
```

D.2 Von der SPOT Instanz aus abfragen, was ihr Status ist

Von der Instanz aus können via RESTful API Metadaten der Instanz abgefragt werden. Bezüglich Determinierung einer Spot-Instanz kann der Zustand *none*, *hibernate*, *stop* oder *terminate* sein. *none*, wenn nichts ansteht. Von da an, wo klar ist, dass die Instanz abgestellt werden wird, kann der Zeitpunkt ausgelesen werden.

Listing 2: Status der Instanz abfragen

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H  
"X-aws-ec2-metadata-token-ttl-seconds: 21600") && curl -H "X-aws-ec2-metadata-token: $TOKEN"  
http://169.254.169.254/latest/meta-data/instance-action
```

D.3 XML Testen

Einfaches Skript, um zu testen, ob alle XML-Dateien well-formed sind.

Listing 3: XML testen

```
import os
from lxml import etree
import logging
import threading

BASEPATH = '/home/ubuntu/processing/input/'
LOGGING_FILE = '/home/ubuntu/data/log/bad_xml.log'

logger = logging.getLogger('not_good')
logger.setLevel(logging.DEBUG)
ch = logging.StreamHandler()
fh = logging.FileHandler(LOGGING_FILE)
logger.addHandler(fh)
logger.addHandler(ch)

def try_xml(my_f):
    try:
        tree = etree.parse(my_f)
    except Exception as e:
        logger.info(my_f)

# r=root, d=directories, f=files
for r, d, f in os.walk(BASEPATH):
    for file in f:
        if '.kml' in file or '.dae' in file:
            my_f = os.path.join(r, file)
            x = threading.Thread(target=try_xml, args=(my_f,))
            x.daemon = True
            x.start()
```

D.4 Publikationsschritte in Ansible

Der ganze Code zum hier aufgelisteten Ausschnitt kann auf github.com eingesehen, resp. verwendet, werden:

Listing 4: Nebst dem Setup werden die drei Publikationsschritte mit Ansible gesteuert.

```
---
- name: set processing file
  vars:
    processing_step: 1
  template:
    src: ../templates/processing_step.txt.tpl
    dest: '{{ processing_step_file }}'
    mode: '0660'
    force: no

# =====
# First step
# =====
- name: first step
  debug:
    msg: "Step 1: Uncompressing and validating data"
  when: lookup('file', processing_step_file)|int <= 1

- name: create input folder
  file:
    path: /home/{{ my_user }}/processing/input
    state: directory
    mode: '0700'
  when: lookup('file', processing_step_file)|int <= 1

# TODO rm subset
- name: unzip files
  shell: |
    for i in /home/{{ my_user }}/data/subset_buildings/*.tar;
    do echo extracting ${i};tar -xzf ${i} -C /home/{{ my_user }}/processing/input/;
    done
  when: lookup('file', processing_step_file)|int <= 1

- name: test if xml are wellformed
  command:
    cmd: python ../tests/test_xml_wellformed.py
  when: lookup('file', processing_step_file)|int <= 1

- name: check how many xml are false
  shell: wc -l /home/{{ my_user }}/data/log/bad_xml.log | cut -d " " -f1
  register: not_wellformed

- debug:
  var: not_wellformed

- name: end play when xml not valid
  fail:
    msg: there is a xml, that is not wellformed
  when: not_wellformed.stdout|int > 0 and lookup('file', processing_step_file)|int <= 1

- name: first step accomplished
  vars:
    processing_step: 2
  template:
    src: ../templates/processing_step.txt.tpl
    dest: '{{ processing_step_file }}'
    mode: '0660'
    force: yes
  when: lookup('file', processing_step_file)|int <= 1
# =====
```

```
# =====
# Second step
# =====
- name: Second step
  debug:
    msg: "Step 2: Processing raw data to web format (Cesium tiles)"
  when: lookup('file', processing_step_file)|int == 2

- name: create output folder
  file:
    path: /home/{{ my_user }}/processing/output
    state: directory
    mode: '0700'
    force: no
  when: lookup('file', processing_step_file)|int == 2

- name: create db folder
  file:
    path: /home/{{ my_user }}/processing/db
    state: directory
    mode: '0700'
    force: no
  when: lookup('file', processing_step_file)|int == 2

- name: create agi log folder
  file:
    path: /home/{{ my_user }}/data/log
    state: directory
    mode: '0700'
    force: no
  when: lookup('file', processing_step_file)|int == 2

- name: get ecr pw
  shell: |
    aws ecr get-login-password
  register: my_pw
  when: lookup('file', processing_step_file)|int == 2

- name: Log into private registry and force re-authorization
  docker_login:
    registry: https://483277333869.dkr.ecr.eu-west-1.amazonaws.com
    username: AWS
    password: "{{ my_pw.stdout }}"
    reauthorize: yes
  when: lookup('file', processing_step_file)|int == 2

- name: process to web format
  docker_container:
    name: analyticalgraphicsinc-swayze
    image:
      483277333869.dkr.ecr.eu-west-1.amazonaws.com/semesterarbeit:analyticalgraphicsinc-swayze
    volumes:
      - /home/{{ my_user }}/processing/input:/var/app/input/
      - /home/{{ my_user }}/processing/output:/var/app/output/
      - /home/{{ my_user }}/processing/db:/var/app/db/
      - /home/{{ my_user }}/data/log:/var/app/log/
    command: ["/root/.nvm/versions/node/v8.11.2/bin/node", "--max-old-space-size=20000",
      "/var/app/node_modules/.bin/roadhouse", "-i", "/var/app/input", "-o", "/var/app/output",
      "--db", "/var/app/db/database", "--clear-normals", "--face-normals", "--max-tiles",
      "1000", "-r", "UUID", "-r", "DATUM_AENDERUNG", "-r", "DATUM_ERSTELLUNG", "-r",
      "ERSTELLUNG_JAHR", "-r", "ERSTELLUNG_MONAT", "-r", "REVISION_JAHR", "-r",
      "REVISION_MONAT", "-r", "GRUND_AENDERUNG", "-r", "HERKUNFT", "-r", "HERKUNFT_JAHR", "-r",
      "HERKUNFT_MONAT", "-r", "OBJEKTART", "-r", "ORIGINAL_HERKUNFT", "-r", "GEBAEUDE_NUTZUNG",
      "-r", "Longitude", "-r", "Latitude", "-r", "Height"]
    detach: false
  register: docker_output
  when: lookup('file', processing_step_file)|int == 2

- name: log dockeroutput
  shell: echo {{ docker_output }} > /home/{{ my_user }}/data/log/agi_log.log
```

```

when: lookup('file', processing_step_file)|int == 2

- name: second step accomplished
  vars:
    processing_step: 3
  template:
    src: ../templates/processing_step.txt.tpl
    dest: '{{ processing_step_file }}'
    mode: '0660'
    force: yes
  when: lookup('file', processing_step_file)|int == 2
# =====

# =====
# Third step
# =====

- name: Third step
  debug:
    msg: "Step 3: Publishing data on S3"
  when: lookup('file', processing_step_file)|int == 3

- name: copy to s3
  command:
    cmd: aws s3 cp --recursive --content-encoding gzip /home/{{ my_user }}/processing/output
      s3://3d-tiles/preview/ch.swisstopo.swisstim3d.3d/current/
  when: lookup('file', processing_step_file)|int == 3

- name: third step accomplished
  vars:
    processing_step: 4
  template:
    src: ../templates/processing_step.txt.tpl
    dest: '{{ processing_step_file }}'
    mode: '0660'
    force: yes
  when: lookup('file', processing_step_file)|int == 3
# =====

```