

Fachhochschule Frankfurt am Main

Fachbereich 2 Informatik und Ingenieurwissenschaften

Bachelorarbeit

im Studiengang Informatik

zur Erlangung des akademischen Grades

Bachelor of Science

von:	Daniel Roth
geb. am:	31.05.1990
Studiengang:	Informatik
Matrikelnummer:	979869
Prüfer:	Prof. Dr. Christian Baun
Korreferent:	Prof. Dr. Thomas Gabel
Thema:	Installation und Evaluation der Cloud-Speicherdienst-Emulation S3 ninja auf Einplatinencomputern
Eingereicht am:	13.09.2016

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter der Angabe der Quellen kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Frankfurt, 12. September 2016

.....

Unterschrift

Danksagung

Hiermit möchte ich mich bei allen bedanken die mich beim Absolvieren meines Studiums unterstützt haben. Insbesondere bei meiner Familie sowie meiner Freundin. Außerdem danke ich Marius Schmidt für die Chancen, die er mir mit diversen Software-Projekten geboten hat, sowie für sein Mentoring in den letzten 5 Jahren.

Ich danke Prof. Dr. Christian Baun und Prof. Dr. Thomas Gabel dafür, dass sie mir als Betreuer sowie Prüfer zur Verfügung standen.

Kurzfassung

Diese Bachelorthesis beschreibt die Verwendung von S3 bzw. der Simple Storage Service (S3) Schnittstelle sowohl programmatisch als auch mit Kommandozeilen-Programmen. Hierzu wird eine Emulation des Amazons S3 Dienstes auf einem Raspberry Pi installiert und zur Verfügung gestellt. In weiteren Schritten wird die Verwendung von S3 in Java und Python Programmen gezeigt und erläutert. Das beschriebene System sowie seine Steuerung wird bewertet sowie seine Verwendbarkeit, Anwendung und Alternativen analysiert.

Abstract

This bachelor thesis describes the usage of the web interface of S3 or more specifically of the S3 emulation S3 ninja. It describes the usage of this interface as well programmatically with Python and Java as with the command line tool s3cmd. To achieve this goal, the s3 emulation S3 ninja will be installed and configured on a Raspberry Pi. The installed system is then used to show the programmatically usage of S3 in Java and Python. The described system will be rated and its usability, application spectrum and alternatives will be analyzed.

Inhaltsverzeichnis

Abbildungsverzeichnis	VIII
Tabellenverzeichnis	IX
Abkürzungen	X
1. Einleitung	1
1.1. Zielsetzung	1
1.2. Aufbau der Arbeit	2
2. Stand der Technik und verwendete Technologien	3
2.1. Raspberry Pi	3
2.1.1. Raspberry Pi Modell 3	5
2.2. Raspbian	6
2.2.1. Secure Shell (SSH)	6
2.3. Hypertext Transfer Protocol (HTTP) und REpresentational State Transfer (RESTful)	7
2.3.1. eXtensible Markup Language (XML)	8
2.3.2. Verstehen einer HTTP-Antwort	9
2.4. Cloud Computing	11
2.4.1. Amazon Web Services	11
2.4.2. Simple Storage Service (S3)	11
2.5. S3 Ninja	13
3. Installation der Cloud-Speicherdienst-Emulation S3 ninja auf einem Raspberry Pi	14
3.1. Vorbereitung des Raspberry Pis	14
3.1.1. Vorbereitung des Images	16
3.1.2. Konfiguration des Raspian Betriebssystems	17
3.2. Installation von S3 ninja	21
4. Bedienung von S3 ninja	25
4.1. Einfache Operationen mit curl	26
4.2. Bedienung mit S3cmd	28
4.2.1. Anlegen einer Konfigurationsdatei und eines Proxy Servers	28
4.2.2. Bedienung	30
4.3. Verwendung der Java Bibliothek	31
4.4. Verwendung der Python Bibliothek (Boto3)	35
4.4.1. Boto3 auf der Kommandozeile	37

5. Bewertung	39
5.0.1. Performance	39
5.1. Nachteile der Umsetzung	40
5.2. Vorteile der Umsetzung	42
5.3. Alternativen	44
5.3.1. Amazon Web Services Developer Stack	44
5.3.2. Docker	44
5.4. Fazit und Ausblick	45
A. Beispiel Anwendung mit Spring - Rückgabe eines XML Dokuments (Autowerkstatt)	49
A.1. Car Model	49
A.2. Car List Wrapper	51
A.3. Car Controller	51
A.4. Spring Main Klasse	53
B. IntelliJ und Maven	54
B.1. IntelliJ Maven Assistent Schritt 1	54
B.2. IntelliJ Maven Assistent Schritt 2	55
B.3. IntelliJ Projekt Struktur	56
C. Java SDK Beispiel	57
C.1. Interface Java Service	57
C.2. Implementierung der S3Service Schnittstelle	58
C.3. AWSCredential Beispiel	60
C.4. S3Service Ende-Zu-Ende Test	60
D. Beispiel Konfiguration für S3cmd	64
E. S3 ninja Bucket Klasse	66

Abbildungsverzeichnis

2.1. Das Raspberry Pi 3 Modell B V1.2	3
2.2. GPIO Übersicht aller Pins	5
2.3. XML im Browser	9
2.4. Übersicht aller AWS Angebote	12
3.1. Netzteil zur Stromversorgung eines Raspberry Pis	15
3.2. Netzteil zur Stromversorgung eines Raspberry Pis	16
3.3. Menüübersicht des Programms raspi-config	18
3.4. Ausgabe der Netzwerkkonfiguration unter Mac OS X	19
3.5. Fritzbox 6490 Address-Konfiguration	20
3.6. S3 ninjas grafische Oberfläche mit Fehler	23
4.1. S3 ninja Log	26
4.2. Aufrufen eines S3 ninja API-Endpunkts im Browser	27
5.1. Raspberry Pi Auslastung mit htop	39
5.2. JMeter Response Time Graph	40
B.1. IntelliJ Maven Assistent Schritt 1	54
B.2. IntelliJ Maven Assistent Schritt 2	55
B.3. IntelliJ Projekt Struktur	56

Tabellenverzeichnis

3.1. Übersicht der SD-Karten Klassen	15
4.1. Übersicht der s3cmd Funktionen	31
5.1. Preisübersicht S3 Speicher (Frankfurt)	41
5.2. Preisübersicht S3 Anforderungen (Frankfurt)	41

Abkürzungen

AWS	Amazon Web Services
S3	Simple Storage Service
HTTP	Hypertext Transfer Protocol
etc.	et cetera
dpkg	Debian Package
APT	Advanced Packaging Tool
i.d.R	in der Regel
API	Application Programming Interface - Programmierschnittstelle
GB	Gigabyte
MB	Megabyte
TB	Terabyte
SQL	Structured Query Language
Bash	Bourne-again shell
JDK	Java Development Kit
SDK	Software Development Kit
z.B.	zum Beispiel
EC2	Elastic Compute Cloud
HDMI	High Definition Multimedia Interface
GPIO	General Purpose Input/Output - Allzeckeingabe/-ausgabe
LED	Licht-emittierende Diode
MHz	Megahertz
JSON	JavaScript Object Notation
RFC	Request For Comments
HTTP	Hypertext Transfer Protocol
WWW	World Wide Web
HTML	Hypertext Markup Language
RESTful	REpresentational State Transfer
SGML	Standard General Markup Language
XML	eXtensible Markup Language
CLI	Command Line Interface - Kommandozeilen

cURL	Client for URLs
URI	Uniform Resource Identifier
NIC	Network Interface Card - Netzwerkkarte
SSH	Secure Shell
Vim	Vi IMproved
SOA	Service Orientierte Architekturen
W3C	World Wide Web Consortium
POM	Project Object Model

1. Einleitung

Seit einigen Jahren werden immer mehr Einplatinencomputer auf dem Markt angeboten. Ihre Anwendungsgebiete reichen von einfachen Experimentierplattformen bis hin zur Steuerung von Systemen in der Industrie. Alle neueren Einplatinen PC Boards sind extrem sparsam mit der Energieaufnahme und lüfterlos zu betreiben. Sogar ein Einsatz im erweiterten Temperaturbereich ist meistens problemlos möglich. Die Ansprüche an die Leistungsfähigkeit und Robustheit von IT-Hardware (Einplatinenrechner) sind in der Industrie besonders hoch. Gleichzeitig werden Bauteile benötigt, die flexibel einsetzbar sind, um die unterschiedlichsten Anwendungsfelder abdecken zu können.¹ Das Raspberry Pi ist der erfolgreichste Einplatinencomputer auf dem Markt.

Ursprünglich wurde das Raspberry Pi entwickelt um Kindern mit wenigen finanziellen Mitteln, das Erlernen des Programmierens sowie den Umgang mit Computern zu ermöglichen. Aus diesem Grund stand bei der Entwicklung ein kostengünstiger Preis im Vordergrund.² Der Preis der neusten Generation liegt zwischen 35€ und 45€. Weitere Kosten können für Tastatur, Maus, Netzteil, HDMI-Kabel, Bildschirm, Netzkabel und Gehäuse hinzu kommen. (Vergleich: Amazon.com 30.08.16)

Auch zur Umsetzung der vorliegenden Arbeit kommt das Raspberry Pi als Experimentierplattform zum Einsatz.

1.1. Zielsetzung

Ziel der vorliegenden Arbeit ist das Installieren sowie erfolgreiches Steuern der Emulation des öffentlichen Cloud-Speicherdienstes Simple Storage Service (S3) von Amazon, S3 Ninja auf einem Raspberry Pi. Dazu sollen die Werkzeuge s3cmd sowie das Software Development Kit (SDK) von Amazon Web Services (AWS) für die Programmiersprachen Java und Python verwendet werden. Es sollen unter anderem alle Installations- sowie Konfigurationsschritte für die Verwirklichung dieses Ziels aufgezeigt werden. Außerdem sollen die verwendeten Technologien sowie deren Anwendung erläutert werden. Abschließend

¹ *Einplatinenrechner (SBC Computer) - zuverlässige Arbeitstiere der Industrie*. Aufgerufen am 30.08.2016.
URL: <https://www.distronek.de/computer/>.

² Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 12.

sollen Alternativen bzw. Anwendungsbereiche dieser Emulation beschrieben werden.

1.2. Aufbau der Arbeit

Das erste Kapitel behandelt die zugrunde liegenden Technologien. Es beschreibt wie diese zu verwenden sind sowie was deren Anwendungsgebiete sind. Ein grundlegendes Verständnis des Zielsystems sowie der Programmierung wird hierbei vermittelt. Das zweite Kapitel beschreibt wie das System zu installieren bzw. konfigurieren ist. Darauf folgt eine ausführliche Beschreibung zur Verwendung von S3 ninja in Python, Java und mit dem Kommandozeilen-Werkzeug s3cmd. Ein Rückblick auf die implementierten Systeme bezüglich Performance, Anwendbarkeit, Umsetzung und Alternativen bewertet die vorher beschriebenen Prozesse und beendet die vorliegende Arbeit mit einem Fazit.

2. Stand der Technik und verwendete Technologien

2.1. Raspberry Pi

Das Raspberry Pi ist ideal geeignet für die Realisierung kleiner und größerer Projekte, wie z.B. eine Hausautomatisierung, ein Mediacenter; oder einfach nur als Experimentierplattform für Experimente und erste Programmiersuche.¹

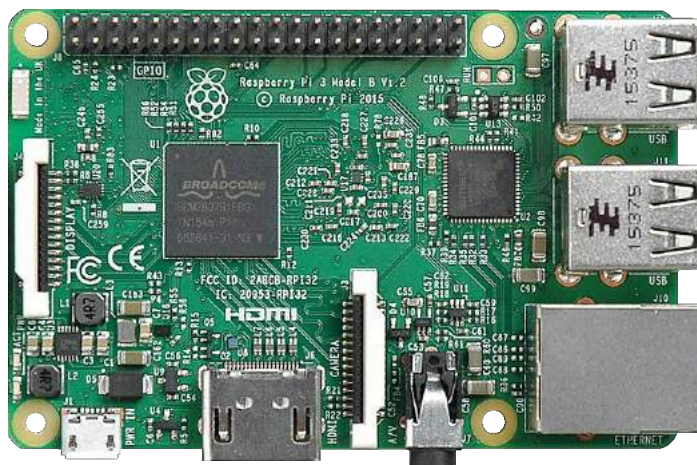


Bild 2.1.: Das Raspberry Pi 3 Modell B V1.2

Das Raspberry Pi wurde von der britischen Raspberry Pi Foundation aus den Komponenten von Android-Smartphones entwickelt. Der Erfolg des seit Anfang 2012 ausgelieferten Raspberry Pi übertraf alle Erwartungen.² Zur Zeit gibt es 3 Generationen des Raspberry Pis:

¹Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 12.

²Michael Kofler, Charly Kühnast und Christoph Scherbeck. *Raspberry Pi: Das umfassende Handbuch. Komplett in Farbe - inkl. Schnittstellen, Schaltungsaufbau, Steuerung mit Python u.v.m.* Rheinwerk, 2015, S. 15.

1. Generation
 - Model A
 - Model B
 - Model A+
 - Model B+
2. Generation
 - Model A
 - Model B
3. Generation
 - Model A
 - Model B

3

Außerdem gibt es das, absolut minimierte, im Jahr 2015 veröffentlichte, Raspberry Pi Zero, für einen Preis von 4.99€. ⁴ Dieses Modell ist aber weitestgehend nicht mehr verfügbar (Stand 13.08.2016).

Natürlich hat auch das Raspberry Pi diverse Konkurrenten auf dem Markt. Welches Gerät ideal geeignet ist, ergibt sich anhand der Art des Projektes. Das Raspberry Pi beispielsweise ist ideal geeignet für Einsteiger, welche das Programmieren von Software sowie Digitaltechnik erlernen wollen aber auch für diverse Projekte für Profis. ⁵ Das Raspberry Pi verfügt seit dem Model A+ der ersten Generation über eine 40-Pin-Steckerleiste mit sogenannten General Purpose Input/Output - Allzweckeingabe/-ausgabe (GPIO) (Abbildung 2.2). ⁶ Die älteren Modelle der ersten Generation verfügten lediglich über 26. Als GPIO werden die Ein- und Ausgänge eines Microcontrollers bzw. eines Prozessors bezeichnet. Mit Hilfe dieser Pins ist es auf einfache Weise möglich Periferie-Geräte zu steuern. Solche Geräte können einfache Schaltkreise z.B. zum Ein- und Ausschalten von LEDs oder komplexerer Geräte wie Funksender oder kleine Touch-Bildschirme.

³Vgl. *Welcher Raspberry Pi? Alle Modelle im Vergleich*. Aufgerufen am 30.08.2016. URL: http://praxistipps.chip.de/welcher-raspberry-pi-alle-modelle-im-vergleich_41923.

⁴Vgl. Michael Kofler, Charly Kühnast und Christoph Scherbeck. *Raspberry Pi: Das umfassende Handbuch. Komplett in Farbe - inkl. Schnittstellen, Schaltungsaufbau, Steuerung mit Python u.v.m.* Rheinwerk, 2015, S. 23.

⁵Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 13.

⁶Vgl. Michael Kofler, Charly Kühnast und Christoph Scherbeck. *Raspberry Pi: Das umfassende Handbuch. Komplett in Farbe - inkl. Schnittstellen, Schaltungsaufbau, Steuerung mit Python u.v.m.* Rheinwerk, 2015, S. 22.

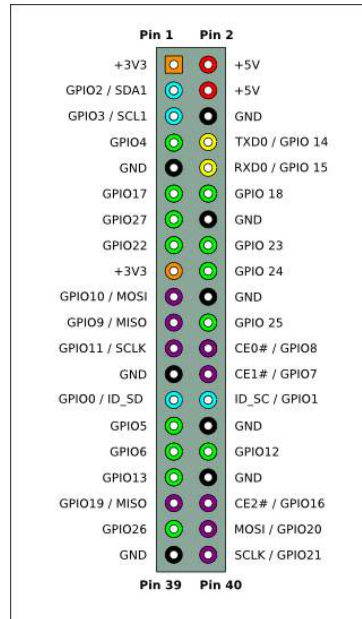


Bild 2.2.: GPIO Übersicht aller Pins

Quelle: http://elinux.org/RPi_Low-level_peripherals Aufgerufen: 13.08.2016

Einige häufig verwendete Module sind unter anderem:

433MHz Funk-Sender: Der Funk-Sender wird häufig, z.B. für die Realisierung von Hausautomatisierungen genutzt.

Kamera: Das Kamera wurde speziell für das Raspberry Pi entwickelt. Im Gegensatz zu den meisten anderen Komponenten für das Raspberry Pi wird die Kamera über eine extra für dieses Modul vorgesehene Schnittstelle mit dem Raspberry Pi verbunden. Meistens wird die Kamera für Projekte im Bereich Gebäudesicherheit als Überwachungskamera verwendet.

Diverse Touch-Bildschirme: Das Angebot an Touch-Bildschirmen ist sehr groß. Daher gibt es auch verschiedenste Arten dieser Bildschirme und unzählige Anwendungsmöglichkeiten. (z.B. als Raspberry Pi Smartphone)

2.1.1. Raspberry Pi Modell 3

Für die Umsetzung des beschriebenen Systems wurde das Raspberry Pi Model B der 3. Generation verwendet. Das Raspberry Pi 3 ist im Februar 2016 erschienen und soll das Raspberry Pi 2 ablösen. Es ist ausgestattet mit:

- 1.2GHz 64-bit quad-core ARMv8 CPU
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core
- 802.11n Wireless LAN

7

Die wesentlichsten Unterschiede zu den Vorgänger-Modellen sind die vorab eingebauten WLAN und Bluetooth Module.

2.2. Raspbian

Raspbian ist ein, kostenloses, auf Debian basierendes Betriebssystem, welches für die Hardware von Raspberry Pis optimiert ist.⁸ Zudem eignet sich Raspbian für die Umsetzung kleinerer Projekte, da viele Bibliotheken und Programme bereits vorinstalliert sind. Weil dieses Betriebssystem auf Debian basiert, verfügt es auch über das Paketverwaltungssystem, Advanced Packaging Tool (APT).

Das Paketverwaltungssystem, APT bzw. Debian Package, auf welchem APT basiert, dient dem zuverlässigen Installieren und Deinstallieren, sowie Warten von Software auf Debian ähnlichen Betriebssystemen (Ubuntu, Knoppix, Raspbian etc.)

2.2.1. Secure Shell (SSH)

Für die Fernsteuerung von Linux Distributionen ist auf allen nennenswerten Betriebssystemen Secure Shell (SSH) vorab installiert. Da Raspbian eines dieser Betriebssysteme ist,

⁷Vgl. *Raspberry Pi 3 Model B*. Aufgerufen am 30.08.2016. URL: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.

⁸*Raspbian Website*. Aufgerufen am 10.08.2016. URL: <https://www.raspbian.org>.

kann mit einer SSH-Verbindung auch auf ein Raspberry Pi zugegriffen werden. Sofern die IP-Adresse(Port) sowie Benutzername und Passwort, des für eine SSH-Verbindung gewünschten Gerätes bekannt ist und ein SSH server darauf ausgeführt wird, kann eine SSH-Verbindung mit dem folgenden Befehl initialisiert werden.⁹

```
1 ssh Benutzername@Ip-Adresse # (z.B: ssh root@192.168.178.10)
```

Sobald die Verbindung zu dem Gerät besteht, kann es über die Kommandozeile ferngesteuert werden.

2.3. Hypertext Transfer Protocol (HTTP) und REpresentational State Transfer (RESTful)

Das Hypertext Transfer Protocol (HTTP) ist ein Protokoll der Anwendungsschicht, für verteilte, kollaborative und multimediale Informationssysteme.¹⁰ Hauptsächlich wird HTTP im World Wide Web (WWW) für das Aufrufen von Webseiten verwendet. Es kann jedoch auch für den Datenzugriff entfernter Ressourcen über REST Services dienen. Diese Art der Verwendung ist für die folgenden Kapitel relevant. Besagte Ressourcen können jeglicher Art sein. (z.B HTML Dokumente, JavaScript Object Notation (JSON) Dokumente, Film- oder Audiodateien etc.)

REpresentational State Transfer (RESTful) ist einer von zwei populären Ansätze zur Realisierung von Dienste-Schnittstellen. Ein Anderer ist SOAP, welcher aber für die Nutzung des hier Verwendeten S3 Dienstes nicht unterstützt wird. REST beschreibt einen Architekturstil, der auf HTTP aufbaut.

Das gängigste Datenaustausch-Format für REST ist XML. Eine Alternative, und ein ebenfalls häufig genutztes Format ist JSON. JSON eignet sich besonders dann, wenn nach der Konsumierung eines RESTful Services im Browser, die Datenverarbeitung durch JavaScript vorgenommen wird.¹¹

Die Konsumierung einer REST API bzw. das Verwalten von Ressourcen wird durch die von HTTP bereitgestellten Methoden ermöglicht.¹² Die Methoden sind POST, PATCH, PUT, GET, DELETE, OPTIONS und HEAD. Den Zugriffspunkt einer Resource bezeichnet man als Uniform Resource Identifier (URI). Auf eine GET-Anfrage an eine solche URI

⁹Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 120.

¹⁰RFC: *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Aufgerufen am 14.08.2016. URL: <https://tools.ietf.org/html/rfc7540>.

¹¹Vgl. Christian Baun u. a. *Cloud Computing: Web-basierte dynamische IT-Services (Informatik im Fokus)*. Springer, 2011, S. 24.

¹²Vgl. Thomas Barton. *E-Business mit Cloud Computing*. Springer, 2014, S. 38.

sollte üblicherweise eine Antwort mit einer Representation der angeforderten Resource in JSON oder XML folgen.

Für die Konsumierung einer REST API relevante Methoden:

- Für das Anfordern von Ressourcen dient die GET Methode
- Das Anlegen von Ressourcen dient die Methode POST
- PATCH wird verwendet für das Ändern bereits vorhandener Ressourcen
- Mit der DELETE-Methode können vorhandene Ressourcen gelöscht werden
- Wird PUT auf eine URI mit vorhandener Resource ausgeführt, wird diese ersetzt. Sollte keine Resource vorhanden sein, kann eine Neue angelegt werden

¹³ ¹⁴

Mit der OPTIONS-Methode lassen sich Informationen zu einem Endpunkt abrufen. Diese Informationen enthalten unter anderem, welche HTTP-Methoden verfügbar sind. Leider ist diese Funktion bei vielen RESTful Services nicht implementiert. HEAD ist ähnlich wie GET, abgesehen davon, dass die Antwort einer HEAD Anfrage keiner Antwort bedarf. Die Kopf-Informationen reichen bei HEAD als Antwort aus. Die Antwort auf eine TRACE Anfrage sollte immer identisch mit der gesendeten Anfrage sein.¹⁵

2.3.1. eXtensible Markup Language (XML)

HTML eignet sich dazu, Informationen im World Wide Web zu präsentieren. Dagegen ist eXtensible Markup Language (XML) eine Sprache zur Beschreibung des Inhalts und der logischen Struktur, die unabhängig von einer Programmiersprache und von einem Betriebssystem ist. XML ist ebenso wie HTML eine Untermenge der universellen Auszeichnungssprache Standard General Markup Language (SGML).¹⁶

Wer RESTful APIs verwenden möchte, muss XML verstehen bzw. lesen können.

Folgendes Beispiel zeigt ein XML-Dokument, welches ein Fahrzeug für die Registrierung in einer Werkstatt beschreibt. Dafür werden Kennzeichen, Modell, Erstzulassung und Kilometer stand in dem Dokument angegeben. Die Namen der XML-Elemente werden üblicherweise in englisch schreibweise verwendet.

¹³Vgl. Thomas Barton. *E-Business mit Cloud Computing*. Springer, 2014, S. 38.

¹⁴Vgl. *RFC: Hypertext Transfer Protocol Version 2 (HTTP/2)*. Aufgerufen am 14.08.2016. URL: <https://tools.ietf.org/html/rfc7540>, S. 52 - 55.

¹⁵Vgl. *RFC: Hypertext Transfer Protocol Version 2 (HTTP/2)*. Aufgerufen am 14.08.2016. URL: <https://tools.ietf.org/html/rfc7540>, S. 54 - 56.

¹⁶Thomas Barton. *E-Business mit Cloud Computing*. Springer, 2014, S. 28.

```
1 <Car>
2     <Licenseplate>F-AB-123</Licenseplate>
3     <Model>VW T4</Model>
4     <Registrationdate>1995</Registrationdate>
5     <Mileage>200000</Mileage>
6 </Car>
```

Sofern das XML Dokument keine Fehler enthält wird es im Browser als Baum angezeigt. (ABB. 2.3)¹⁷



Bild 2.3.: XML im Browser

2.3.2. Verstehen einer HTTP-Antwort

Ein sehr verbreitetes Werkzeug zum einfachen Senden von HTTP Nachrichten ist Client for URLs (cURL). cURL ist ein Kommandozeilen-Programm zum versenden von Daten unter der Verwendung verschiedener Protokolle. Zu den unterstützten gehören HTTP, HTTPS, FTP, FTPS, DICT, LDAP, RTMP und Gopher.¹⁸ ¹⁹ Eine GET-Anfrage an einen beliebigen Server ist mit cURL sehr einfach. Dieses Beispiel zeigt eine GET Anfrage an eine Beispiel Applikation nach dem oben genannten Beispiel einer Autowerkstatt. (Quellcode: Anhang A.1 - A.4, zum Downloaden: <https://github.com/Darot/cardemo#cardemo>)

```
1 curl -H "Accept: application/xml" -H "Content-Type: application/xml"
2 -X GET http://localhost:8080/cars -i
```

¹⁷Vgl. Thomas Barton. *E-Business mit Cloud Computing*. Springer, 2014, S. 28.

¹⁸Vgl. Michael Kofler, Charly Kühnast und Christoph Scherbeck. *Raspberry Pi: Das umfassende Handbuch. Komplett in Farbe - inkl. Schnittstellen, Schaltungsaufbau, Steuerung mit Python u.v.m.* Rheinwerk, 2015, S. 128.

¹⁹Vgl. Daniel Stenberg. *Curl Manual*. Curl 7.40.0. 2014.

Die Anfrage mit cURL wird an den Endpunkt mit der URL: `http://localhost:8080/cars` gestellt. Im Header der Anfrage werden "Content-Type: application/xml" bzw. "Accept: application/xml" angegeben und geben Information über das Format der erwarteten Antwort. In diesem Fall ein XML-Dokument. Sollte diese Anfrage an einen Endpunkt gestellt werden, welcher statt eines XML-Dokumentes z.B. JSON zurück gibt, sollte dieser eine Fehlermeldung zurück geben.

```
1 HTTP/1.1 200
2 X-Application-Context: application
3 Content-Type: application/xml
4 Transfer-Encoding: chunked
5 Date: Mon, 15 Aug 2016 13:25:25 GMT
6
7 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
8 <cars>
9     <car>
10         <initialRegistration>1990</initialRegistration>
11         <licensePlate>F-A-123</licensePlate>
12         <mileage>200000</mileage>
13         <model>Volkswagen T4</model>
14     </car>
15     <car>
16         <initialRegistration>1991</initialRegistration>
17         <licensePlate>F-B-456</licensePlate>
18         <mileage>300000</mileage>
19         <model>Porsche 911</model>
20     </car>
21 </cars>
```

Der Nachrichtenkopf ist in den Zeilen 1-5 zu sehen. Er enthält Informationen zum Format der enthaltenen Daten sowie einen Zeitstempel und den Code 200. In diesem Fall enthält der Kopf den Code 200 (Success). Das bedeutet, dass keine Fehler bei der Anfrage aufgetreten sind. Außerdem sieht man in Zeile 3, dass das Format der enthaltenen Daten XML ist. In Zeile 4 wird der, von HTTP verwendete Übertragungsmechanismus beschrieben. In Zeile 7 ist die XML Auszeichnung zu sehen. Jedes XML-Dokument sollte mit einer solchen Auszeichnung beginnen. Sie enthält Information zur Version, Zeichenkodierung und Verarbeitung.²⁰ Von Zeile 8 - 21 sind die angeforderten Ressourcen zu sehen, in diesem Fall eine Liste von Fahrzeugen.

²⁰ XML/Regeln/XML-Deklaration. Aufgerufen am 15.08.2016. URL: <https://wiki.selfhtml.org/wiki/XML/Regeln/XML-Deklaration>.

2.4. Cloud Computing

In Cloud Computing Systemen wird die Komplexität der Informationstechnologie vor Nutzern und Entwicklern verborgen. Man muss nicht im Einzelnen wissen, wie ein Dienst generiert wird und es ist die Aufgabe des Dienstleisters, eine entsprechende Abstraktionsschicht bereitzustellen. Cloud Computing basiert auf 3 Technologien: Virtualisierung, Service Orientierte Architekturen (SOA), und Web Services.²¹

Virtualisierung abstrahiert Software Komponenten, Datenspeicher (z.B. S3), Server und Netzwerke von ihren Abhängigkeiten und der darunter liegenden Hardware. SOA ist ein Architekturstil, welcher das Anbieten und Nutzen von Diensten definiert. Die Dienste wiederum können sowohl von einem Endnutzer genutzt werden, als auch von weiteren Diensten. Bei Webservices handelt es sich um eine asynchrone, nachrichtenbasierte Kommunikation.²²

2.4.1. Amazon Web Services

Einer der größten Dienstleister im Bereich Cloud Computing ist Amazon. Mit Amazon Web Services(AWS) stellt Amazon eine große Palette Cloud-Dienstleistungen bereit. AWS wurde am 13. März 2006 veröffentlicht. Damals gab es nur den Simple Storage Service(S3). Etwas später im Jahr 2006 erweiterte Amazon das Angebot mit Elastic Compute Cloud (EC2), ein Cloud Computing Service, welcher Rechenleistung für Virtuelle Server nach bedarf liefert.²³ Ein großer Vorteil der AWS Dienstleistungen ist die Bezahlung nach Nutzung. Das bedeutet, es muss lediglich die benötigte Rechenleistung bzw. der verwendete Speicher bezahlt werden. In den kommenden 6 Jahren erweiterte Amazon das Angebot mit mehr als 25 weiteren Cloud-Dienstleistungen. (siehe Abbildung 2.4)

2.4.2. Simple Storage Service (S3)

S3 ist der skalierbare Cloud-Speicher von AWS. Es bietet eine umfangreiche und gut dokumentierte Schnittstelle für Websevicees. Entwicklungspackete stehen für alle häufig verwendeten Programmiersprachen zur Verfügung. Unter anderem Python, Java, PHP, .NET, Ruby und für die mobilen Plattformen Android und iOS.²⁴

²¹Vgl. Christian Baun u. a. *Cloud Computing: Web-basierte dynamische IT-Services (Informatik im Fokus)*. Springer, 2011, S. 9.

²²Vgl. Christian Baun u. a. *Cloud Computing: Web-basierte dynamische IT-Services (Informatik im Fokus)*. Springer, 2011, S. 9, 19, 22.

²³Vgl. Bernard Golden. *Amazon Web Services FOR DUMMIES*. Wiley, 2013, S. 10, 11.

²⁴Vgl. *Getting Started with Amazon Simple Storage Service*. Aufgerufen am 10.08.2016. URL: <http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>.

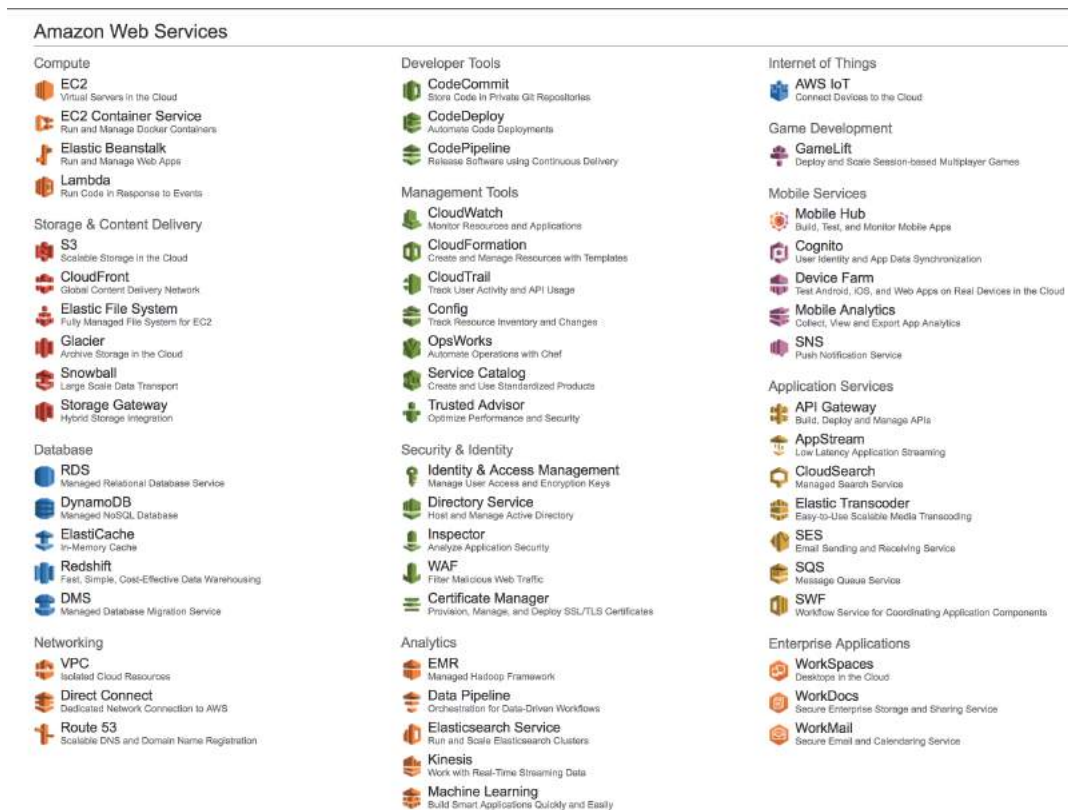


Bild 2.4.: Übersicht aller AWS Angebote

Quelle: AWS Management Console

Amazon S3 bietet eine Vielzahl von Speicherklassen für diverse Anwendungsfälle an. Hierzu gehören Amazon S3 Standard für die Speicherung allgemein genutzter Daten mit häufigem Zugriff, Amazon S3 Standard – Infrequent Access (Standard-IA) für Langzeitdaten mit selteneren Zugriffen und Amazon Glacier zur Langzeitarchivierung.²⁵ S3 muss nicht zwangsläufig als einzelner Service bereit gestellt werden. Andere Services, wie EC2, Lambda oder ElastiCache, können S3 Verzeichnisse (Buckets) verwenden um z.B. Produktiv-Daten oder Backups zu sichern. Diese Buckets können öffentlich, nur für einen oder mehrere Services oder für User einer bestimmten Rolle freigegeben sein.

S3 wird von sehr vielen Unternehmen verwendet:

Dropbox: Dieser Cloud-Speicher und Synchronisations-Service verwendet S3 um alle Daten und Dokumente seiner Benutzer zu speichern.²⁶

Netflix: Dieser Video Service verwendet S3 um alle Videos zu speichern bevor sie für die Benutzer ausgegeben werden. Tatsächlich wird Netflix zu 100% über AWS bereit gestellt.²⁷

Medcommons: Dieses Unternehmen speichert medizinische Dokumente wie Röntgenbilder mittels S3.²⁸

Brickscout: Dieser Marktplatz für den Handel mit Lego Produkten verwendet S3 für die Speicherung von über 40.000 Bildern angebotener Lego Produkte.

2.5. S3 Ninja

S3 Ninja ist eine kostenlose Emulation des S3 Dienstes und steht unter MIT Lizenz. S3 Ninja ist nicht für den produktiv Betrieb geeignet sondern zu Test- und Entwicklungszwecken. Derzeit sind noch nicht alle Funktionen, welche die originale API zur Verfügung stellt, unterstützt. Aber die wichtigsten Funktionen, welche zur Entwicklung und zu Testen benötigt werden, sind bereits implementiert. Die HTTP Methoden GET, PUT, HEAD und DELETE werden derzeit unterstützt.²⁹

²⁵ *Getting Started with Amazon Simple Storage Service*. Aufgerufen am 10.08.2016. URL: <http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>.

²⁶ Vgl. Bernard Golden. *Amazon Web Services FOR DUMMIES*. Wiley, 2013, S. 60.

²⁷ Vgl. Bernard Golden. *Amazon Web Services FOR DUMMIES*. Wiley, 2013, S. 60.

²⁸ Vgl. Bernard Golden. *Amazon Web Services FOR DUMMIES*. Wiley, 2013, S. 60.

²⁹ Vgl. *S3 Ninja Webseite*. Aufgerufen am 14.08.2016. URL: <http://s3ninja.net>.

3. Installation der Cloud-Speicherdienst-Emulation S3 ninja auf einem Raspberry Pi

Für die Installation, Konfiguration und Verwendung von Raspberry Pis sowie S3 ninja sind Grundlegende Kenntnisse von Unix Betriebssystemen Voraussetzung. Zudem sind Kenntnisse in einer Programmiersprache wie Java, Python oder C++ empfehlenswert. Beispiele zur Verwendung der S3 Application Programming Interface - Programmierschnittstelle (API) stehen in den Sprachen Java und Python zur Verfügung. Die Unix Befehle sowie der Quellcode zur Realisierung und Verwendung dieses Systems sind im Laufe der nächsten Kapitel vorgeführt und ausführlich beschrieben. Bildschirmfotos, sowie die Unix Befehle aus den Folgenden Kapiteln zeigen die Nutzung der verwendeten Technologien unter Mac OS X El Capitan Version 10.11.

3.1. Vorbereitung des Raspberry Pis

Neben dem Raspberry Pi werden weitere Komponenten benötigt. Zur Stromversorgung des Raspberry Pi's wird ein Netzteil(Abbildung 3.1) mit 1,5 bis 2,5 A benötigt.¹ Die Leistung eines Netzteils ist i.d.R auf der Unterseite des Netzteils dokumentiert(Abbildung 3.2). Außerdem wird ein Micro-SD-Karte benötigt. Empfehlenswert sind mindestens 8 GB und einer Schreibgeschwindigkeit von mindestens 2 MB pro Sekunde.² Abhängig von dem Volumen der Daten, welche in der Cloud gespeichert werden sollen, könnte die Micro-SD-Karte über zu wenig Speicher verfügen. Da das Raspberry Pi über 4 USB Anschlüsse verfügt, kann der Speicher durch externe Laufwerke erweitert werden. Die Schreibgeschwindigkeiten von SD-Karten ist anhand der Klasse zu erkennen, der die SD Karte angehört(Siehe Tabelle 3.1). Optional aber empfehlenswert ist eine Gehäuse um das Raspberry Pi vor Beschädigungen zu schützen.

Für die erste Inbetriebnahme wird ein HDMI-Kabel benötigt um das Raspberry Pi mit einem Bildschirm zu verbinden und die ersten Konfigurationen vor zu nehmen. Danach

¹Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 18.

²Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 17.

wird das HDMI-Kabel nicht mehr benötigt.

Das Raspberry Pi 3 verfügt, gegenüber seiner Vorgängermodelle, über ein WLAN-Modul. Ebenso verfügt es über einen Ethernet (RJ45) Port. Somit ist es möglich, bei der ersten Inbetriebnahme über die grafische Oberfläche, mit einem kabellosen Netzwerk zu verbinden oder mit einem RJ45 Kabel und dem Ethernet Port das Raspberry Pi mit einem Netzwerk zu verbinden.

Tab. 3.1.: Übersicht der SD-Karten Klassen

Klasse	Schreibgeschwindigkeit
Klasse 2	mind. 2 MB/s max. 12,5 MB/s
Klasse 4	mind. 4 MB/s max. 12,5 MB/s
Klasse 6	mind. 6 MB/s max. 12,5 MB/s
Klasse 10	mind. 10 MB/s max. 25 MB/s
UHS-I(SDHC)	max. 50 MB/s
UHS-I(SDXC)	max. 104 MB/s
UHS-II(SDHC)	max. 156 MB/s
UHS-II(SDXC)	max. 312 MB/s

3

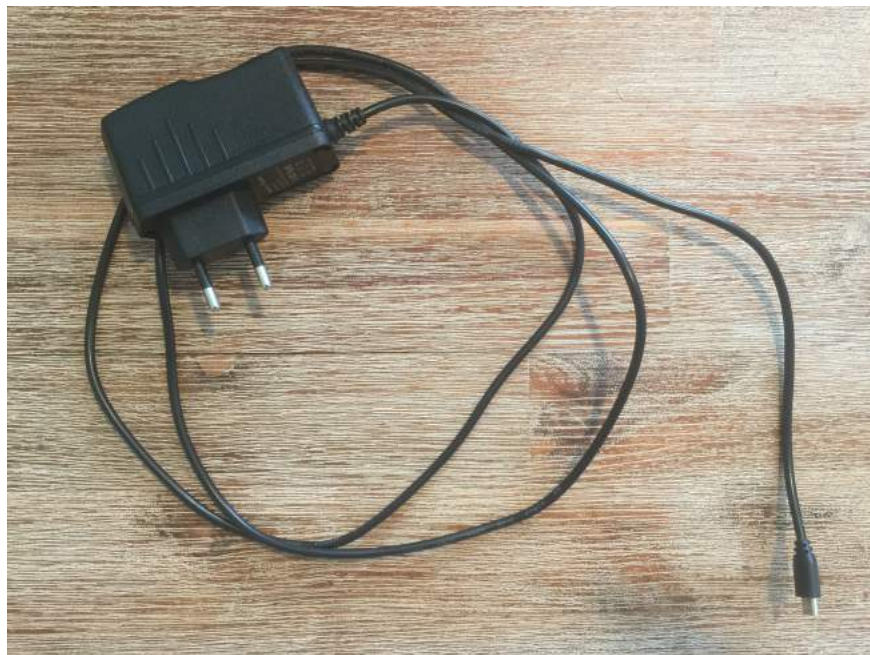


Bild 3.1.: Netzteil zur Stromversorgung eines Raspberry Pis

³Vgl. *SD Card Speed Classes, Grades, Bus Modes, and File Systems Explained*. Aufgerufen am 30.08.2016.
URL: <http://www.pcpaper.com/reviews/Editorial/SD-Card-Speed-Classes-Grades-Modes-and-File-Systems-Explained>.



Bild 3.2.: Netzteil zur Stromversorgung eines Raspberry Pis

3.1.1. Vorbereitung des Images

Als erstes sollte die SD-Karte formatiert werden. Das bedeutet, dass alle darauf gespeicherten Daten verloren gehen.

Dafür benötigt man den BSD-Namen (wie z.B /dev/disk1s1). Im Apple-Menu ist dieser unter 'USB' nachzulesen.⁴ Oder mit dem Programm diskutil:

```
1 diskutil list
```

Die Ausgabe sollte wie folgt aussehen, wobei hier der BSD-Name der SD-Karte disk1s1 ist:

```
1 /dev/disk0 (internal, physical):
2   #:                                TYPE NAME                                SIZE
3   IDENTIFIER
4   0:                                GUID_partition_scheme                *500.3 GB   disk0
5   1:                                EFI EFI                               209.7 MB   disk0s1
6   2:                                Apple_CoreStorage Macintosh HD       402.4 GB   disk0s2
7   3:                                Apple_Boot Recovery HD               650.0 MB   disk0s3
   4:                                SD-Karte                              3,96 GB   disk1s1
```

Das Programm diskutil ermöglicht ebenfalls das Formatieren von Massenspeichergeräten:

```
1 sudo diskutil eraseDisk FAT32 NAME MBRFormat /dev/disk1s1
```

⁴Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 25.

'NAME' sollte hierbei durch den gewünschten Namen für das Medium ersetzt werden. (z.B Raspbian)

Die Abbild-Datei der Raspbian Distribution steht kostenlos als Download, auf der offiziellen Webseite (<https://www.raspbian.org>) zur Verfügung. Die im Beispiel verwendete Version von Raspian ist folgende:

- RASPBIAN JESSIE
- Veröffentlichung :2016-05-27
- Kernel Version:4.4

Sobald die Abbild-Datei heruntergeladen und mit einer Software wie UnRarX entpackt wurde, kann das Abbild mit den folgenden Befehlen auf der SD-Karte installiert werden. Damit die SD-Karte überschrieben werden darf, muss sie zuvor ausgeworfen werden. Das geschieht ebenfalls mit dem Programm diskutil.

```
1 diskutil unmount /dev/{BSD-Name}
```

Das Abbild kann mit dem folgenden Befehl auf die SD-Karte geschrieben werden:

```
1 sudo dd if=Pfad zum Image.img of=/dev/disk1s1 bs=1m
```

'Pfad zum Image.img' muss ersetzt werden mit dem absoluten oder relativen Pfad des Raspbian Abbilds. (z.B ~/Downloads/2016-05-27-raspbian-jessie.img)

Sobald sich das Abbild auf der SD-Karte befindet, kann sie im Raspberry Pi verwendet werden. Dafür muss nur die SD-Karte in das Raspberry Pi gesteckt und die Stromversorgung angeschlossen werden. Sobald das Raspberry Pi mit Strom versorgt wird, beginnt der Bootvorgang des Raspbian Betriebssystems.

3.1.2. Konfiguration des Raspian Betriebssystems

Nachdem das Raspberry Pi gestartet wurde, erscheint zunächst das Programm Raspberry Pi Software Configuration Tool (raspi-config). Hierbei handelt es sich um ein Programm, für die Unterstützung bei der Konfiguration des Raspian Betriebssystems. Die Abbildung 3.3 zeigt das Hauptmenü des Programms.

Als erstes sollte 'Expand Filesystem' ausgewählt werden damit das Dateisystem sowie das Betriebssystem auf der gesamten SD-Karte ausgebreitet wird.⁵ Außerdem ist es empfehlenswert das Passwort zu ändern. Da ein SSH-Server standardmäßig aktiviert ist, kann über das Menü 'Enable Boot to Desktop/Scratch' die grafische Oberfläche deaktiviert werden. Soll WLAN für die Verbindung zum Netzwerk verwendet werden ist es empfehlenswert die grafische Oberfläche erst nach dem Anmelden im Netzwerk zu deaktivieren. Das Programm kann jederzeit wieder über die Kommandozeile mit dem Befehl

⁵Vgl. Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014, S. 39.

'raspi-config' aufgerufen werden. Nach dem Einstellen dieser drei Optionen kann das Programm beendet werden. Wurde raspi-conf geschlossen, öffnet sich die Kommandozeile bzw. die grafische Oberfläche des Raspbian Betriebssystems.

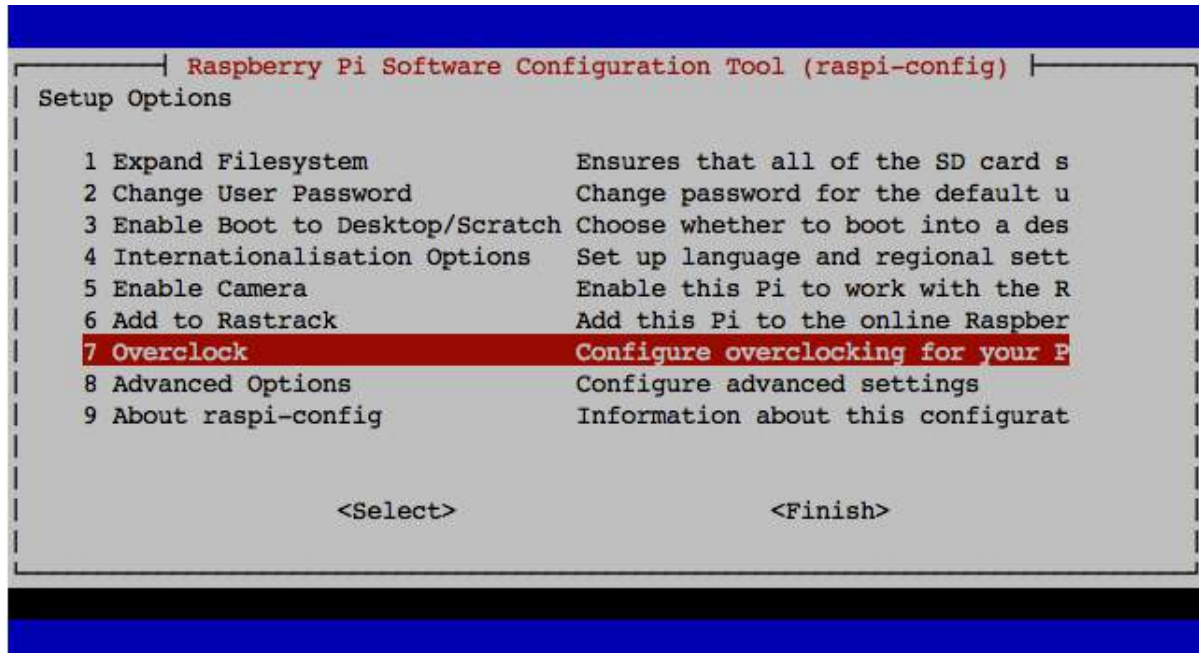


Bild 3.3.: Menüübersicht des Programms raspi-config

Quelle: <https://upload.wikimedia.org/wikipedia/commons/e/ed/Raspi-config.png>

Vergabe einer statischen IP-Adresse

Damit nach jedem Neustart des Raspberry Pi's kein unnötiger Konfigurations-Aufwand entsteht, ist es empfehlenswert, eine statische IP-Adresse zu vergeben. Dafür muss zunächst die IP-Adresse und der Netz- bzw. Hostbereich der IP-Adresse, analysiert und diese ggf. nach Wunsch konfiguriert werden. Die Standardeinstellung des Raspberry Pi ist die Adresse 192.168.1.1. Diese Adresse liegt damit erwartungsgemäß nach RFC1918⁶ in einem für private Netzwerke vorgesehen IP Bereich.

```
1 ifconfig
```

Wie auf der Abbildung 3.4 zu sehen ist, hat das Netz die Adresse 192.168.1.1. Also ist der bereich von 192.168.1.2 - 192.168.178.254 an Geräte im Netzwerk zu vergeben. Die Subnetzmaske ist 0xfffff00. Das bedeutet, dass das Netzwerk in keine weiteren Subnetze aufgeteilt ist.

Mit dem Programm ifconfig, kann auch eine statische IP-Adresse vergeben werden.

⁶Vgl. RFC1918. Aufgerufen am 30.08.2016. URL: <https://tools.ietf.org/html/rfc1918>.

3.1 Vorbereitung des Raspberry Pis

```
Daniels-MacBook-Pro:Bachelorarbeit daniel$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    inet 127.94.0.1 netmask 0xff000000
    inet 127.94.0.2 netmask 0xff000000
    nd6 options=1<PERFORMNUD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether ac:bc:32:b6:5e:45
    inet6 fe80::aebc:32ff:feb6:5e45%en0 prefixlen 64 scopeid 0x4
    inet 192.168.1.2 netmask 0xfffff00 broadcast 192.168.1.255
    nd6 options=1<PERFORMNUD>
    media: autoselect
    status: active
en1: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
    options=60<TS04,TS06>
    ether 6a:00:01:40:77:e0
    media: autoselect <full-duplex>
    status: inactive
en2: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
    options=60<TS04,TS06>
    ether 6a:00:01:40:77:e1
    media: autoselect <full-duplex>
    status: inactive
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
    ether 0e:bc:32:b6:5e:45
    media: autoselect
    status: inactive
awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1484
    ether fa:b6:97:eb:4b:fb
    inet6 fe80::f8b6:97ff:feeb:4bfb%awdl0 prefixlen 64 scopeid 0x8
    nd6 options=1<PERFORMNUD>
    media: autoselect
    status: active
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=63<RXCSUM,TXCSUM,TS04,TS06>
    ether ae:bc:32:6b:a0:00
    Configuration:
        id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
        maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
        root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
        ipfilter disabled flags 0x2
    member: en1 flags=3<LEARNING,DISCOVER>
        ifmaxaddr 0 port 5 priority 0 path cost 0
    member: en2 flags=3<LEARNING,DISCOVER>
        ifmaxaddr 0 port 6 priority 0 path cost 0
    nd6 options=1<PERFORMNUD>
    media: <unknown type>
    status: inactive
Daniels-MacBook-Pro:Bachelorarbeit daniel$
```

Bild 3.4.: Ausgabe der Netzwerkkonfiguration unter Mac OS X

```
1 ifconfig en0 192.168.178.70 netmask 255.255.255.0 up
```

In diesem Beispiel wurde die IP-Adresse 192.168.178.70 an die Network Interface Card - Netzwerkkarte (NIC) mit der Bezeichnung en0 vergeben. Dieses Beispiel bezieht sich auf ein Klasse C Netzwerk, welches nicht in weitere Subnetze unterteilt wurde, was anhand der Subnetzmaske 255.255.255.0 zu erkennen ist. Diese Netzwerkkonfiguration kann in jedem Netzwerk unterschiedlich sein. Die Bezeichnung des NIC kann ebenfalls auf anderen Rechnern unterschiedlich sein. Außerdem erhält das Raspberry Pi, sobald es mit einem Netzwerk verbunden wurde, üblicher Weise eine IP-Adresse von einem DHCP-Server. Die meisten Router bieten ebenfalls die Möglichkeit, angeschlossenen Geräten feste IP-Adressen zuzuweisen. (Siehe Abbildung 3.5)

Auf dieser Seite werden Detailinformationen zum Netzwerkgerät bzw. Benutzer angezeigt.


Name	<input type="text" value="raspberrypi"/>	<input type="button" value="Zurücksetzen"/>
IPv4-Adresse	<input type="text" value="192.168.178.70"/>	
	<input type="checkbox"/> Diesem Netzwerkgerät immer die gleiche IPv4-Adresse zuweisen.	
Geräteinformation	<input type="text" value="B8:27:EB:62:2B:6C, dhcpcd-6.7.1:Linux-4.4.11-v7+:armv7l:BCM2709"/>	
Heimnetzanbindung		
	<input type="text" value="raspberrypi"/>  <input type="text" value="fritz.box"/>	

Bild 3.5.: Fritzbox 6490 Address-Konfiguration

Installation benötigter Software

Mit APT können die benötigten Werkzeuge und die Java Laufzeitumgebung installiert werden. Es ist empfehlenswert die Informationen zu verfügbaren Debian Paketen zu aktualisieren.

```
1 sudo apt-get update
```

Nach dem ausführen dieses Befehls sind alle Informationen verfügbarer Software aktualisiert. Desweiteren können die aktualisierten Informationen genutzt werden, um die bereits installierten Debian Softwarepakete zu aktualisieren.

```
1 sudo apt-get upgrade
```

Alle verfügbaren Aktualisierungen, der auf dem System installierten Debian Pakete werden, auf diese Weise installiert.

Für das in den folgenden Kapiteln erklärte, Anpassen von Konfigurationsdateien eignet sich der Kommandozeilen-Editor Vi IMproved (Vim). Da S3 Ninja in Java entwickelt wurde,

wird eine JAVA Laufzeitumgebung benötigt. Im Gegensatz zu dem Python Interpreter, ist die JAVA Laufzeitumgebung nicht vorinstalliert. Diese Programme können ebenfalls, mit Hilfe von APT, installiert werden:

```
1 sudo apt-get install vim
2 sudo apt-get install openjdk-8-jre
```

3.2. Installation von S3 ninja

S3 Ninja ist nicht, wie viele andere Programme, in den Debian Paketen enthalten und muss daher herunter geladen werden. Für diesen Zweck, ist das Programm wget geeignet.

Vor dem Herunterladen sollte zunächst ein neues Verzeichnis im Heimatverzeichnis angelegt werden und in das angelegte Verzeichnis gewechselt werden:

```
1 mkdir ~/s3ninja
2 cd ~/s3ninja
```

S3 ninja kann mit folgendem Befehl in das aktuelle Verzeichnis herunter geladen werden:

```
1 wget https://oss.sonatype.org/content/groups/public/com/scireum/s3ninja/
2 2.7/s3ninja-2.7-sources.jar
```

Das Programm unzip ermöglicht das Auspacken der heruntergeladenen Datei:

```
1 unzip s3ninja-2.7-sources.jar
```

Nach dem Ausführen der zuvor beschriebenen Befehle, sollte das Verzeichnis nach dem Auflisten mit dem Programm ls wie folgt aussehen:

```
1 pi@raspberrypi:~/s3ninja \ $ ls -lh
2
3 drwxr-xr-x  5 pi pi 4.0K Mar 14 15:07 app
4 -rw-r--r--  1 pi pi  57 Aug 19  2015 config.sh.sdsignore
5 drwxr-xr-x  3 pi pi 4.0K Jun  6 14:32 data
6 -rw-r--r--  1 pi pi  61 Aug 19  2015 instance.conf.sdsignore
7 -rw-r--r--  1 pi pi 6.3K Aug 18  2015 IPL.class
8 drwxrwxrwx  2 pi pi 4.0K Mar 14 15:07 lib
9 drwxr-xr-x  2 pi pi 4.0K Aug 16 07:17 logs
10 drwxr-xr-x  3 pi pi 4.0K Mar 14 15:07 META-INF
11 -rwxr-xr-x  1 pi pi 102K Aug 19  2015 prunmgr.exe
12 -rwxr-xr-x  1 pi pi 204K Aug 19  2015 prunsrv-ia64.exe
```


3.2 Installation von S3 ninja

```
13 -rwxr-xr-x  1 pi pi 102K Aug 19  2015 prunsrv-x64.exe
14 -rwxr-xr-x  1 pi pi  79K Aug 19  2015 prunsrv-x86.exe
15 -rw-r--r--  1 pi pi  2.3K Aug 19  2015 README
16 -rw-r--r--  1 pi pi   24M Jun  6 14:28 s3ninja-2.7-zip.zip
17 -rwxr-xr-x  1 pi pi  3.8K Aug 19  2015 sirius.sh
```

In Zeile 17 ist das Skript zu sehen, welches zum steuern der Software dient. Wird dieses Skript ohne Angabe weiterer Parameter verwendet, gibt das Programm Informationen zur richtigen Verwendung aus.

```
1 pi@raspberrypi:~/s3ninja \ $ ./sirius.sh
2
3 SIRIUS Launch Utility
4 =====
5
6 Use a custom config.sh to override the settings listed below
7
8 SERVICE:
9 SIRIUS_HOME:  /home/pi/s3ninja
10 JAVA_CMD:    java
11 JAVA_OPTS:   -server -Xmx1024m -Djava.net.preferIPv4Stack=true
12 SHUTDOWN_PORT: 9191
13 STDOUT:     logs/stdout.txt
14 USER_ID:
15
16 Usage: sirius.sh start|stop|restart|show|logs|install|uninstall|patch
```

Möchte man S3 Ninja starten, ist das mit dem folgenden Befehl möglich:

```
1 ./sirius start
```

Da der Server im Hintergrund ausgeführt wird erhält man nur eine kurze Information darüber, dass der Server gestartet wurde.

```
1 SIRIUS Launch Utility
2 =====
3
4 Use a custom config.sh to override the settings listed below
5
6 SERVICE:
7 SIRIUS_HOME:  /home/pi/s3ninja
8 JAVA_CMD:    java
9 JAVA_OPTS:   -server -Xmx1024m -Djava.net.preferIPv4Stack=true
```

3.2 Installation von S3 ninja

```
10 SHUTDOWN_PORT: 9191
11 STDOUT:          logs/stdout.txt
12 USER_ID:
13
14 Starting Application...
```

Außerdem werden einige Informationen über die Konfiguration von S3 ninja ausgegeben. In Zeile 7 sieht man eine wichtige Information. Hierbei handelt es sich um das Verzeichnis, welches S3 ninja nutzen soll, um hochgeladenen Daten zu persistieren.

S3 Ninja bietet eine grafische Oberfläche welche über den Browser aufrufbar ist, sobald es gestartet wurde. Die grafische Oberfläche ist unter der URL `HTTP://raspberrypi-adresse:9444` aufrufbar. Wie auf Abbildung 3.6 zu sehen ist, wird eine Fehlermeldung angezeigt. Das liegt daran, dass das Verzeichnis, welches in der standard Konfiguration angegeben ist, nicht existiert. Dieses Problem ist einfach zu lösen, indem man das Verzeichnis nachträglich angelegt.

```
1 mkdir /home/pi/s3ninja/data/s3
```

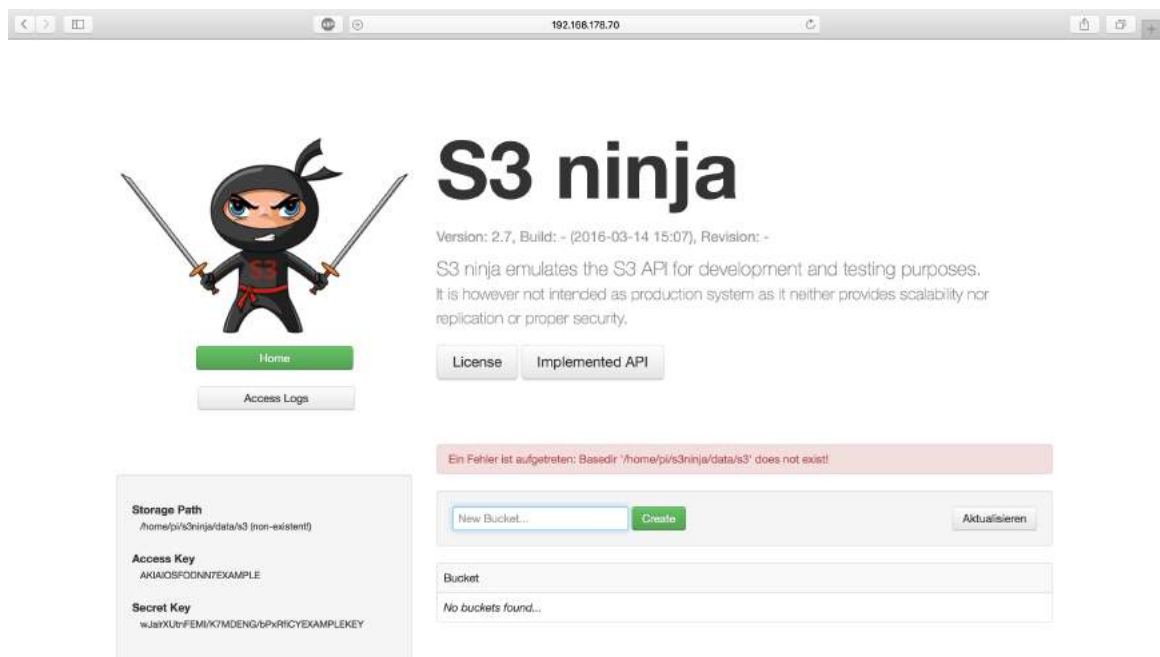


Bild 3.6.: S3 ninjas grafische Oberfläche mit Fehler

Damit nach einem Neustart des Systems S3 Ninja automatisch startet, muss man folgende Zeile in die Datei `/etc/rc.local` einfügen:

```
1 su pi -c "home/pi/s3ninja/sirius.sh start"
```

Beispiel einer rc.local Datei:

```
1 #!/bin/sh -e
2 #
3 # rc.local
4 #
5 # This script is executed at the end of each multiuser runlevel.
6 # Make sure that the script will "exit 0" on success or any other
7 # value on error.
8 #
9 # In order to enable or disable this script just change the execution
10 # bits.
11 #
12 # By default this script does nothing.
13
14 # Print the IP address
15 _IP=$(hostname -I) || true
16 if [ "$_IP" ]; then
17     printf "My IP address is %s\n" "$_IP"
18 fi
19
20 su pi -c "home/pi/s3ninja/sirius.sh start"    #run s3ninja on startup
21
22 exit 0
```

4. Bedienung von S3 ninja

Die einfachste Möglichkeit S3 ninja zu verwenden, bietet die grafische Oberfläche für den Browser. Das Anlegen von Buckets kann sofort über die Startseite der Oberfläche vorgenommen werden. Dafür reicht es, einen Namen des zu erstellenden Buckets in das Textfeld einzugeben und auf create zu klicken.

Jedes neu erstellte Bucket ist ein privates Bucket. Das bedeutet, dass bei POST-Anfragen, also zum Anlegen einer neuen Resource in das Bucket, ein valider Hash-Code zur Verifizierung im Kopf der Nachricht angegeben sein muss.

Ebenfalls auf der Startseite angezeigt ist eine Liste aller bereits erstellten Buckets. Klickt man auf eines dieser Buckets erhält man eine Übersicht aller darin enthaltenen Dateien sowie die Möglichkeit, das ausgewählte Bucket zu veröffentlichen bzw. zu privatisieren. Zudem ist es möglich, per Drag-and-Drop, Dateien in das Bucket zu laden. Ist ein Bucket öffentlich, so bedeutet das, dass jeder ohne Weiteres vollen Zugriff auf das Bucket hat.

Auf der Startseite der Oberfläche ist ein grauer Bereich zu sehen, welcher Informationen zur S3 ninja Instanz enthält. Zu sehen ist das Schlüsselpaar, bestehend aus Access Key und Secret Key. AWS verwendet solche Schlüsselpaare zur Autorisierung bei einem Zugriff auf bereitgestellte Services oder Ressource. S3 ninja Emuliert die Verwendung dieser Schlüssel lediglich. Eine Zugriffssicherung ist nicht implementiert. Hash-Codes werden jedoch bei schreibenden Anfragen validiert.

Klickt man auf der Startseite auf Access Logs erhält man eine Übersicht aller Zugriffe über die API.(Siehe Abbildung 4.1)

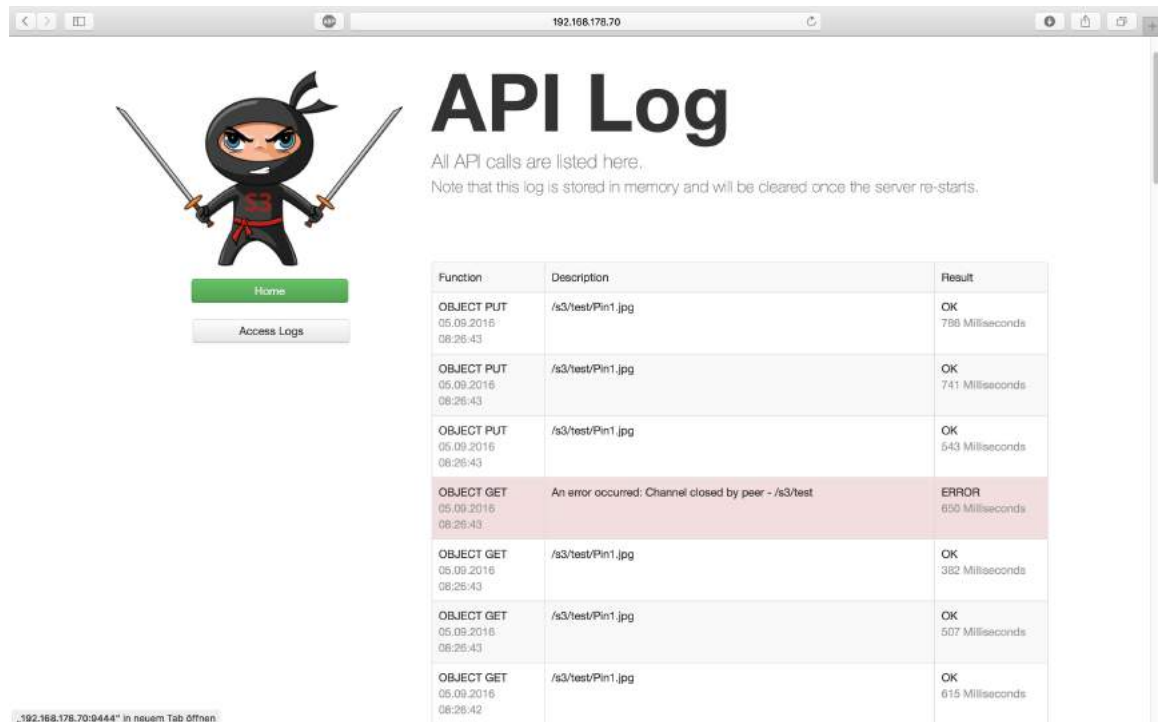


Bild 4.1.: S3 ninja Log

4.1. Einfache Operationen mit curl

Wie bereits erwähnt, ist curl ein nützliches Werkzeug für den Zugriff auf Web-Ressourcen. Mit curl kann ebenfalls S3 ninja gesteuert werden. So ist es z.B. möglich eine Liste aller bestehender Buckets über die Kommandozeile abzufragen. Der folgende Befehl sendet eine GET-Anfrage an den Endpunkt 'http://192.168.178.70:9444/s3':

```
1 curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
   http://192.168.178.70:9444/s3
```

Wie erwartet antwortet der Server mit einer XML-Representation der angeforderten Ressource. In diesem Fall eine Übersicht aller Buckets. Durch das Ergänzen von '-i' würden auch die Kopf-Informationen angezeigt werden. Die Antwort sieht wie folgt aus:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ListAllMyBucketsResult
3     xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
4     <Owner>
5         <ID>initiatorId</ID>
6         <DisplayName>initiatorName</DisplayName>
7     </Owner>
8     <Buckets>
```

```

9         <Bucket>
10             <Name>test</Name>
11
12         <CreationDate>2016-09-11T12:46:27.000Z</CreationDate>
13     </Bucket>
14     <Bucket>
15         <Name>mynewbucket</Name>
16
17     <CreationDate>2016-09-11T12:46:34.000Z</CreationDate>
18 </Bucket>
19 </Buckets>
20 </ListAllMyBucketsResult>

```

Natürlich sind auch weitere Operationen möglich. Dafür müssen die HTTP-Anfragen mit curl genau so beschrieben werden, wie es die RESTful-Schnittstelle von S3 ninja erwartet. So ist das Auflisten aller Objekte eines Buckets mit dem Namen 'Test' wie folgt möglich:

```

1 curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
    http://192.168.178.70:9444/s3/test

```

Wie zu sehen ist, unterscheiden die beiden Befehle sich lediglich anhand der angegebenen URI. Da S3 ninja keinerlei weiterer Informationen benötigt bei GET-Anfragen können diese URIs auch identisch im Browser aufgerufen werden. Der Browser stellt die angefragte Ressource direkt als Baum dar (Siehe Abbildung 4.2).

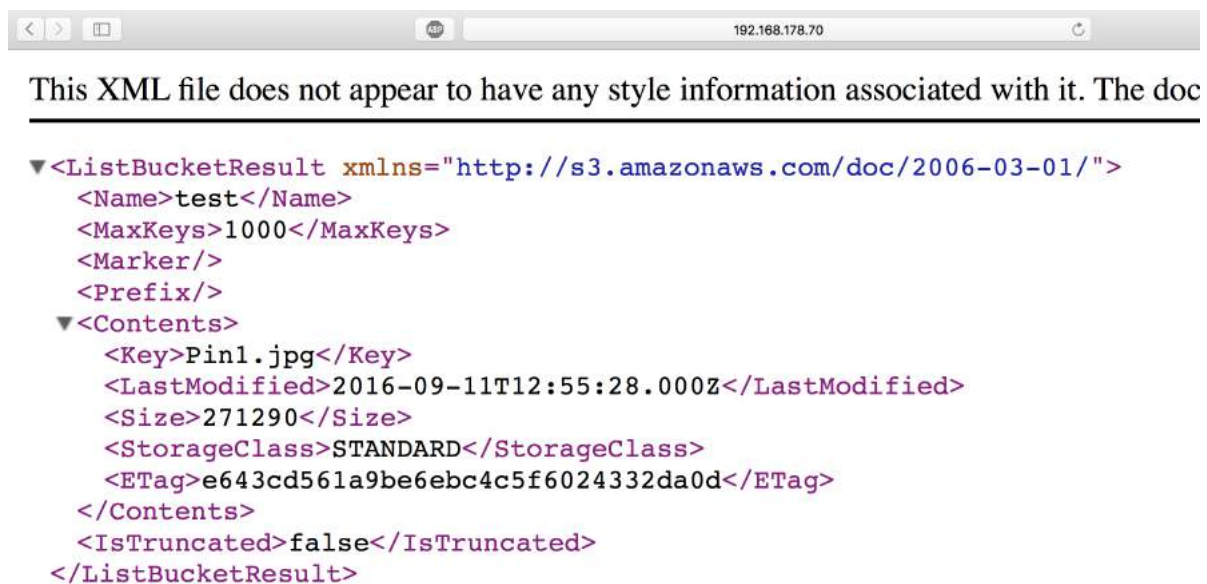


Bild 4.2.: Aufrufen eines S3 ninja API-Endpunkts im Browser

Sofern Buckets als öffentlich markiert sind können auch POST-, PUT- und PATCH-Anfragen ans S3 ninja gesendet werden. Sollte ein Bucket als privat markiert sein, ist es sehr aufwendig mit curl solche Anfragen zu senden, da für einen validen Hash zur Autorisation eine Signatur berechnet werden muss. Ansonsten ist auch das Anlegen eines neuen Objektes mit dem folgenden Befehl möglich:

```
1 curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X PUT  
   -d ~/Desktop/Pin1.jpg http://192.168.178.70:9444/s3/test/pin1.jpg
```

Das Steuern von S3 ninja mit Curl ist vergleichsweise aufwendig aber dennoch hilfreich für erste Schritte mit S3 ninja.

4.2. Bedienung mit S3cmd

Für die Steuerung von S3 über die Kommandozeile ist S3cmd ein hilfreiches Werkzeug. Der Zugriff mit S3cmd ist viel komfortabler, als mit simplen HTTP Werkzeugen wie curl. Zudem werden Signaturen automatisch berechnet. S3Cmd ist eine in Python programmierte kostenlose Software. Mit Hilfe des Python-Paket-Verwaltungs-Programm pip kann s3cmd installiert werden. Das Programm pip ist auf dem Raspbian-Betriebssystem bereits vorinstalliert. Mit pip kann s3cmd wie folgt installiert werden:

```
1 sudo pip install s3cmd
```

4.2.1. Anlegen einer Konfigurationsdatei und eines Proxy Servers

Bevor S3cmd genutzt werden kann, muss eine Konfigurations-Datei im Heimatverzeichnis des Nutzers vorhanden sein. S3cmd bietet für diesen Zweck eine Funktion. Nach dem Aufrufen von s3cmd, mit der Option `--configure`, wird der Nutzer aufgefordert einige Konfigurationsparameter einzugeben:

```
1 s3cmd --configure
```

Nach Eingabe dieses Befehls wird der Nutzer aufgefordert einige Konfigurationsparameter anzugeben. Zu beginn fordert der Konfigurationsassistent dazu auf, einen Secret Key und einen Access Key einzugeben. Ein solches Schlüsselpaar kann in der Konfigurationsdatei oder auf der grafischen Oberfläche von S3 ninja nachgelesen werden. Desweiteren fordert s3cmd den Nutzer auf, anzugeben, ob HTTPS verwendet werden soll. Sollte S3cmd genutzt werden, um auf echte S3 Buckets zuzugreifen, sollte man dieses Funktion nutzen. Für S3 ninja ist HTTPS jedoch nicht notwendig. Alle weiteren Eingabeaufforderungen können durch Drücken der Eingabe-Taste ignoriert werden.

Um mit s3cmd S3 ninja zu steuern, reichen die von s3cmd erfragten Konfigurationsparameter nicht aus. Um weitere Änderungen an der Konfiguration vorzunehmen, kann die zuvor erzeugte Konfigurationsdatei mit dem Kommandozeilen Texteditor vim geöffnet werden.

```
1 vim ~/.s3cfg
```

Die Parameter `host_base` und `host_bucket` geben den Endpunkt sowie die URI Struktur für S3 an. Diesen Parametern müssen die folgenden Werte zugewiesen sein:

```
1 host_base = s3.eu-central-1.amazonaws.com
2 host_bucket = s3.eu-central-1.amazonaws.com
```

Im Anhang D kann eine Beispiel-Konfiguration eingesehen werden.

Bei der Verwendung von s3cmd zur Steuerung von S3 Ninja besteht ein Problem bezüglich der Adressierung. Aus diesem Grund werden für diese beiden Felder nicht die IP-Adresse des Raspberry Pis angegeben. Sollte man für diese beiden Parameter eine IP-Adresse angeben (z.B 192.168.278.70:9444), hat das eine Fehlermeldung zur Folge, sobald man eine Anfrage mit s3cmd sendet.

S3 Ninja verwendet eine pfadähnliche URI Struktur für den Zugriff auf Buckets wie IP-Adresse: Port/Bucketname. Der Standard von AWS bzw. S3cmd hingegen verwendet Subdomains zur Adressierung von Buckets. Durch das Entfernen von '%(bucket)s' aus der Zeichenkette des Parameters `host_bucket` wird s3cmd angewiesen, eine pfadähnliche URI Struktur zu verwenden.

Damit unter der Adresse `s3.eu-central-1.amazonaws.com` das Raspberry Pi zu erreichen ist, ist die Datei `/etc/hosts` anzupassen. Die `hosts`-Datei wird verwendet, um Rechnernamen in IP-Adressen aufzulösen. Ein Adress-Eintrag einer `hosts` Datei für die zuvor beschriebene Konfiguration sieht wie folgt aus:

```
1 192.168.178.70 s3.eu-central-1.amazonaws.com
```

Um über `s3.eu-central-1.amazonaws.com` gestellte Anfragen an das Raspberry Pi an den richtigen Endpunkt umzuleiten, wird ein Proxy Server benötigt. Einen einfachen Proxy Server für diesen Zweck ist schnell mit `nginx` zu realisieren. `Nginx` kann mit Hilfe der Debian Paketverwaltung auf dem Raspberry Pi installiert werden:

```
1 sudo apt-get install nginx
```

Damit `nginx` Anfragen an S3 ninja weiterleitet, muss die Konfigurationsdatei `/etc/nginx/nginx.conf` angepasst werden. Die folgende Konfiguration leitet Anfragen an `s3.eu-central-1.amazonaws.com` an S3 ninjas API-Endpunkt `127.0.0.1:9444/s3` weiter:


```
1 server {
2     listen 80;
3     server_name s3.eu-central-1.amazonaws.com;
4
5     location / {
6         proxy_pass      http://127.0.0.1:9444/s3/;
7     }
8 }
```

4.2.2. Bedienung

S3cmd ist ein einfach zu verwendendes aber mächtiges Werkzeug für die Steuerung von S3. Sobald eine Konfiguration angelegt wurde, kann S3cmd mit einfachen Befehlen wie 'ls' aufgerufen werden. Buckets bzw. Objektpfade, werden URL-Ähnlich dargestellt. So kann der Inhalt eines Buckets mit dem Namen 'test' (Pfad: s3://test) wie folgt ausgegeben werden:

```
1 s3cmd ls s3://test
2 2016-08-31 11:18      178  s3://test/test.txt
3 2016-09-05 08:26  271290  s3://test/Pin1.jpg
```

Wie zu sehen ist enthält das Bucket 2 Objekte. S3cmd bietet einen großen Umfang an Befehlen für den Zugriff auf Buckets. Die Tabelle 4.1 zeigt eine Übersicht der Befehle.

Tab. 4.1.: Übersicht der s3cmd Funktionen

Auflisten aller Buckets	s3cmd ls
Auflisten aller Objekte eines Buckets	s3cmd ls s3://BUCKET/KEY
Auflisten aller Objekte in allen Buckets	s3cmd la
Anlegen eines Objektes in einem Bucket	s3cmd put DATEIPFAD s3://BUCKET/KEY
Herunterladen eines Objektes aus einem Bucket	s3cmd get s3://BUCKET/KEY LOKALE_DATEI
Löschen eines Objektes aus einem Bucket	s3cmd del s3://BUCKET/KEY
Synchronisation eines lokalen Verzeichnisse mit einem Bucket	s3cmd sync LOKALES_VERZEICHNIS s3://BUCKET/
Auflisten von Informationen über eines Bucketsoder Objektes	s3cmd info s3://BUCKET/[KEY]
Verschieben eines Objektes in ein anderes Bucket	s3cmd mv s3://BUCKET1/KEY1 s3://BUCKET2/KEY2
Kopieren eines Objektes	s3cmd cp s3://BUCKET1/KEY1 s3://BUCKET2[/KEY2]

1

Diese Übersicht zeigt nur einen Bruchteil der zur Verfügung stehenden Funktionen. Einige Funktionen, deren Zweck es ist Sicherheits-Einstellungen an Buckets vorzunehmen sind hier nicht relevant, da S3 ninja keine Sicherheitsmechanismen implementiert.

4.3. Verwendung der Java Bibliothek

Amazon stellt eine Bibliothek zur Verwendung von Amazon Web Services in Java Programmen zur Verfügung. Die Java Programme wurden mit IntelliJ Idea 14 mit der Unterstützung von Maven umgesetzt. IntelliJ Idea verfügt über einen Assistenten zum Erstellen neuer Maven Projekte. In Anhang B.1 - B.3 ist das Erstellen eines Maven Projektes in Bildform dargestellt. Nachdem ein neues Maven Projekt erstellt wurde ist auf der linken Seite die erstellte Projektstruktur zu sehen. Der Assistent zum erstellen von Maven Projekten hat u.a. eine POM Datei(pom.xml) im Projektverzeichnis angelegt. Eine POM-Datei ist

¹Vgl. s3tools.org. *s3cmd Manual*. s3cmd 1.6.1.

eine Representation eines Maven Projektes und enthält Konfigurationen, Projektabhängigkeiten sowie Informationen zur Entwicklung. Diese Datei muss wie folgt angepasst werden, um auf das AWS SDK verwenden zu können.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     http://maven.apache.org/xsd/maven-4.0.0.xsd">
6
7     <groupId>de.thesis</groupId>
8     <artifactId>storedemo</artifactId>
9     <version>1.0-SNAPSHOT</version>
10
11     <dependencies>
12         <dependency>
13             <groupId>com.amazonaws</groupId>
14             <artifactId>aws-java-sdk</artifactId>
15             <version>1.11.22</version>
16         </dependency>
17     </dependencies>
18
19
20 </project>
```

Unter src/main/java ist das Projektverzeichnis 'de.thesis' zu sehen. In diesem Verzeichnis können weitere Unterverzeichnisse und Java Klassen angelegt werden.

In diesem Verzeichnis wird nun eine Java Schnittstelle (Interface) mit dem Namen S3Service angelegt, welches die zu implementierende Klasse beschreibt. Die zu Implementierende Klasse, soll einige grundlegende S3 Operationen ausführen können.

Das Interface beschreibt die folgenden Methoden:

- createObject: Erstellen eines neuen Objektes
- getObject: Laden eines Objektes
- listObject: Auflisten aller Buckets
- createBucket: Erstellen eines neuen Buckets
- deleteBucket: Löschen eines Buckets
- deleteObject: Löschen eines Objektes

Diese Schnittstelle kann von Klassen, welche den Zugriff auf S3 benötigen, verwendet werden, unabhängig von der eigentlichen Implementation. Auf diese Weise können De-

klaration und Implementation getrennt werden. Das ist wichtig, denn in Softwareprojekten kann es immer vorkommen, dass sich die Implementation einer Klasse ändert bzw. eine weitere Implementation der selben Schnittstelle (z.B bei profilgesteuerter Software) benötigt wird. Die Definition dieser Schnittstelle ist im Anhang C.1 zu sehen. Der folgende Quellcode zeigt eine Implementierung der S3Service Schnittstelle:

```
1 public class S3ServiceImpl implements S3Service {
2
3     private final AmazonS3 s3Client;
4
5
6     public S3ServiceImpl(AmazonS3 s3Client) {
7         this.s3Client = s3Client;
8     }
9
10    //upload a file to a bucket
11    public S3Object createObject(String bucketName, String key, File file) {
12        s3Client.putObject(bucketName, key, file);
13        return s3Client.getObject(bucketName, key);
14    }
15
16    //downlaod a file from a bucket
17    public S3Object getObject(String bucketName, String key) {
18        return s3Client.getObject(new GetObjectRequest(bucketName, key));
19    }
20
21    //list all objects of a bucket
22    public List<S3ObjectSummary> listObjects(String bucketName) {
23        return s3Client.listObjects(bucketName).getObjectSummaries();
24    }
25
26    //list all buckets
27    public List<Bucket> listBuckets() {
28        return s3Client.listBuckets();
29    }
30
31    //create a new bucket
32    public Bucket createBucket(String bucketName) {
33        return s3Client.createBucket(bucketName);
```

```
34     }
35
36     //delete a bucket
37     public void deleteBucket(String bucketName) {
38         s3Client.deleteBucket(bucketName);
39     }
40
41     //delete object(file) from bucket
42     public void deleteObject(String bucketName, String key) {
43         s3Client.deleteObject(bucketName, key);
44     }
45
46     //delete a bucket
47     public String getPresignedUrl(String bucketName) {
48         return s3Client.generatePresignedUrl(bucketName, "test", new
49         Date()).toString();
50     }
51 }
```

- Zeile 1: Deklaration der Klasse S3ServiceImpl. 'implements S3Service' zeigt, dass die Klasse die Schnittstelle S3Service implementiert
- Zeile 6-8: Der Konstruktor der Klasse S3ServiceImpl wird verwendet um die Klasse zu initialisieren. Er erhält einen Parameter der Klasse AmazonS3. Die Klasse AmazonS3 ist eine Schnittstelle des AWS SDKs für Java. Sie kann z.B. für den Zugriff auf S3 und EC2 verwendet werden.
- Zeile 10 - 49: Die Implementierung der in der zuvor beschriebenen Schnittstelle definierten Methoden
- Zeile 11-14: Die Methode 'createObjekt' legt ein neues Objekt in einem Bucket an. zusätzlich zu den Angaben des Buckets und des Keys erhält sie ein Objekt von Typ File. Das Objekt verweist auf eine existierende Datei, welche in das Bucket geladen werden soll.
- Zeile 17-19: Herunterladen eines Objektes aus einem Bucket.
- Zeile 22-24: Erstellen einer Liste mit Informationen aller in einem Bucket gespeicherten Objekte.
- Zeile 27-29: Erstellen einer Liste aller vorhandener Buckets
- Zeile 32-24: Erstellen eines neues Bucket
- Zeile 37-39: Löschen eines Buckets. Alle darin angelegten Objekte werden ebenfalls gelöscht.
- Zeile 42-44: Löschen eines Objektes aus einem Bucket

- Zeile 47-49: Erzeugen einer Presigned URL

Die folgenden 5 Zeilen zeigen das Verwenden der AmazonS3 Schnittstelle sowie der zuvor beschriebenen S3Service Schnittstelle:

```
1 AmazonS3 s3Client = new AmazonS3Client(new AwsCredentialImpl());
2 s3Client.setEndpoint("http://192.168.178.70:9444/s3");
3 s3Client.setS3ClientOptions(S3ClientOptions.builder()
4     .setPathStyleAccess(true).build());
5 s3Service = new S3ServiceImpl(s3Client);
```

- Zeile 1: Initialisierung der Klasse AmazonS3Client.
- Zeile 2: Konfiguration des S3 Endpunkts
- Zeile 3-4: Konfiguration für den Zugriff im Pfad Stil
- Zeile 5: Initialisierung der Klasse S3ServiceImpl

Die Klasse AmazonS3Client, welche ebenfalls aus dem AWS SDK stammt, ist eine Implementierung der AmazonS3 Schnittstelle. Die Klasse AwsCredentialImpl ist eine Beispiel-Implementierung der AwsCredential Schnittstelle. Sie enthält das Schlüsselpaar für den Zugriff auf S3. Eine Implementierung dieser Schnittstelle wird von der Klasse AmazonS3Client zur Autorisation benötigt. Diese Klasse dient nur zu Testzwecken und sollte keinesfalls in produktiven Systemen verwendet werden. Ein vollständiges Beispiel-Projekt mit Ende-Zu-Ende Tests, welche diese Klassen verwenden befindet sich im Anhang C

4.4. Verwendung der Python Bibliothek (Boto3)

Auch für Python stellt Amazon ein SDK zur Verfügung. Das SDK für Python wird Boto3 genannt. Boto3 ist der Nachfolger von Boto. Boto3 wurde von Grund auf neu geschrieben, um native Unterstützung für Python 2.6.5+, 2.7 und 3.3+ zu bieten.²

Mit 'pip' kann Boto3 installiert und sofort in Python Programmen verwendet werden.

```
1 pip install boto3
```

Python ist eigentlich eine imperative Programmiersprache. Dennoch ist es ohne weiteres möglich, mit Python objektorientiert zu programmieren.³ Der folgende Quellcode zeigt einen, mit dem zuvor gezeigten Java Quellcode vergleichbaren, python Code zur Steuerung von S3.

```
1 class S3Service:
```

²Vgl. *AWS SDK für Python Dokumentation*. Aufgerufen am 31.08.2016. URL: <https://aws.amazon.com/de/sdk-for-python/>.

³Vgl. Johannes Ernesti und Peter Kaiser. *Python 3 Das Umfassende Handbuch*. Galileo Computing, 2012, S. 28.

```
2
3 def __init__(self, s3):
4     self.__s3 = s3
5
6 def get_object(self, bucket_name, key, filename):
7     with open(filename, 'wb') as data:
8         self.__s3.Bucket(bucket_name).download_fileobj(key, data)
9     return data
10
11 # Upload file to a bucket
12 def create_object(self, bucket_name, key, path):
13     data = open(path, 'rb')
14     self.__s3.Bucket(bucket_name).put_object(Key=key, Body=data)
15     return self.__s3.Bucket(bucket_name).Object(key)
16
17
18 def list_objects(self, bucket_name):
19     return self.__s3.Bucket(bucket_name).objects.all()
20
21 # Create a new bucket
22 def create_bucket(self, bucket_name):
23     self.__s3.create_bucket(Bucket=bucket_name)
24     return self.__s3.Bucket(bucket_name)
25
26 # List all Buckets
27 def list_buckets(self):
28     return self.__s3.buckets.all()
29
30 # Delete a bucket
31 def delete_bucket(self, bucket_name):
32     self.__s3.Bucket(bucket_name).delete()
```

- Zeile 3-4: Konstruktor der Klasse S3Service
- Zeile 6-9: Herunterladen eines Objektes aus einem Bucket
- Zeile 12-15: Erstellen eines neuen Objektes
- Zeile 18-19: Auflisten aller Objekte eines Buckets
- Zeile 22-24: Erzeugen eines neuen Buckets
- Zeile 27-28: Auflisten aller Buckets
- Zeile 31-32: Löschen eines Buckes

Die folgenden 3 Zeilen zeigen wie die S3Service Klasse initialisiert wird.

```
1 s3 = boto3.resource(service_name='s3',
2 endpoint_url='http://192.168.178.70:9444/s3')
3 s3.meta.client.meta.events.unregister('before-sign.s3', fix_s3_host)
service = S3Service(s3)
```

- Zeile 1: Erzeugen einer Boto3 Resource mit S3 ninja Endpunkt als Konfiguration
- Zeile 2: Diese Zeile ist notwendig, um Boto3 so zu konfigurieren, dass es URIs im Pfad Stil verwendet und den gegebenen Endpunkt nicht überschreibt
- Zeile 3: Initialisieren des S3Service Klasse

Im Gegensatz zu Java ist es in Python nicht nötig, eine Schnittstelle zu definieren bzw. zu implementieren. In Python ist Mehrfachvererbung sowie sogenanntes Ducktyping möglich. Deshalb ist es nicht notwendig, in Fällen in welchen mit Java eine Schnittstelle benötigt wird, mit Python eine solche zu definieren.

4.4.1. Boto3 auf der Kommandozeile

Obwohl Python viele Sprachelemente gängiger Scriptsprachen implementiert, handelt es sich um eine interpretierte Programmiersprache. Der Unterschied zwischen einer Programmier- und einer Scriptsprache liegt im sogenannten Compiler. Ähnlich wie Java oder C# verfügt Python über einen Compiler, der aus dem Quelltext ein Kompilat erzeugt, den sogenannten Byte-Code. Dieser Byte-Code wird dann in einer virtuellen Maschine, dem Python-Interpreter, ausgeführt.⁴

Durch die Eingabe von 'python' auf der Kommandozeile öffnet sich der interaktive Modus oder auch Python-Shell genannt. Der Interpreter verarbeitet jede Zeile sofort nach der Eingabe und führt diese aus. Das folgende Beispiel zeigt wie Boto3 direkt auf der Kommandozeile im interaktiven Modus benutzt werden kann.⁵

```
1 Python 2.7.11 (default, Jan 22 2016, 08:29:18)
2 [GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
3 Type "help", "copyright", "credits" or "license" for more information.
4 >>> import boto3
5 >>> from botocore.utils import fix_s3_host
6 >>> s3 = boto3.resource(service_name='s3',
7 endpoint_url='http://192.168.178.70:9444/s3')
>>> s3.meta.client.meta.events.unregister('before-sign.s3', fix_s3_host)
```

⁴Johannes Ernesti und Peter Kaiser. *Python 3 Das Umfassende Handbuch*. Galileo Computing, 2012, S. 28.

⁵Vgl. Johannes Ernesti und Peter Kaiser. *Python 3 Das Umfassende Handbuch*. Galileo Computing, 2012, S. 31.


```
8 >>>
9 >>> for bucket in s3.buckets.all():
10     ...     print(bucket.name)
11     ...
12 test
13 meinthesisbucket
14 test2
15 newbucket
16 >>> quit()
```

- Zeile 1-3: Ausgabe des Editors mit Informationen zu Version und Lizenz
- Zeile 4-7: Boto3 Konfiguration wie im vorigen Kapitel beschrieben
- Zeile 9-10: for-Schleife zur Ausgabe der Namen aller Buckets
- Zeile 12-15: Ausgabe der Schleife (Zeile 10)
- Zeile 16: Beenden des Editors

5. Bewertung

5.0.1. Performance

Leider gerät das Raspberry Pi während der Verwendung von S3 ninja schnell an seine Grenzen. So erreicht der Prozessor des Raspberry Pis beim Verarbeiten von 4-5, mit wenigen Millisekunden Abstand versendeten, Nachrichten seine maximale Auslastung. Der Empfang der Antworten, kann dann bis zu einigen Sekunden dauern. Die Abbildung 5.1 zeigt die Auslastung des Raspberry Pis mit dem Programm htop während der Verarbeitung von ca. 500 Anfragen pro Minute, welche in kurzen Abständen mit JMeter versendet wurden. JMeter ist eine kostenlose Software zum Ausführen von Lasttests.

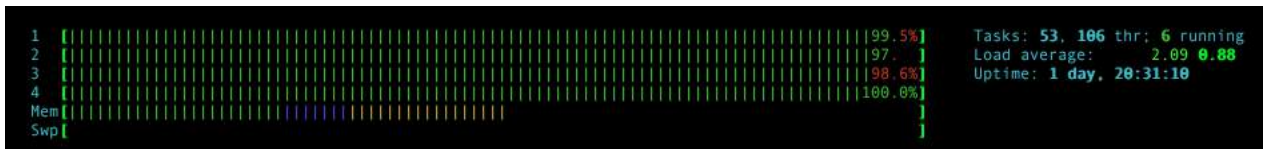


Bild 5.1.: Raspberry Pi Auslastung mit htop

Wie zu sehen ist, ist der Prozessor am Maximum seiner Kapazitäten angelangt. Das geschieht aber nur dann, wenn die Anfragen sich auf den Inhalt eines Buckets beziehen. Der Grund dieser hohen Auslastung ist auf den Quellcode von S3 ninja zurück zu führen. Der Anhang E zeigt die Bucket-Implementierung aus dem S3 ninja Quellcode. Bereits die Deklaration der Klassen-Attribute in Zeile 42 und 43 vermuten lässt, wird bei der Verwendung dieser Klasse häufig auf das Dateisystem zugegriffen. Da Zugriffe auf das Dateisystem die Laufzeit einer Software deutlich verlangsamen, ist der genannte Effekt darauf zurück zu führen. In nahezu jeder Methode der Klasse wird über die Schnittstelle 'File' auf das Dateisystem zugegriffen.

Eine GET Anfrage an die URI '/s3' hingegen liefert eine Resource mit einer Liste aller angelegter Buckets. Da lediglich die Buckets selbst Teil der Information sind und nicht deren Inhalt, sind die Antwortzeiten dieses Endpunkts deutlich schneller. Anfragen an diese URI einer S3 Ninja Instanz auf einem Raspberry Pi erfordern nahezu keinen Rechenaufwand. Somit ist das Raspberry Pi in der Lage, auf mehrere tausend Anfragen pro Minute zu Antworten. Die Abbildung 5.2 zeigt einen JMeter-Bericht nach dem Senden von ca.

500 Anfragen an 4 Endpunkte innerhalb einer Minute. Hierbei handelt es sich um die Endpunkte zum auflisten von Buckets und Objekten sowie Herunter- und Hochladen von Objekten. Auch zu sehen ist, dass etwa jede 20. Anfrage einen Fehler verursacht. Das liegt daran, dass das Raspberry Pi manche Anfragen verwirft, weil es nicht schnell genug darauf antworten kann.

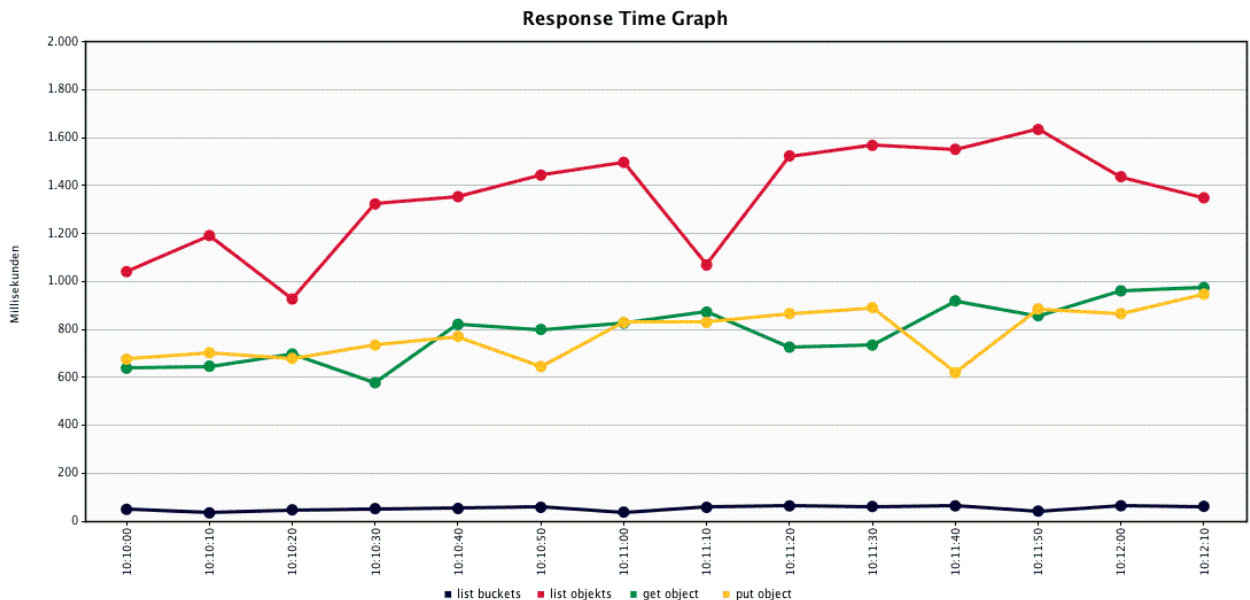


Bild 5.2.: JMeter Response Time Graph

Zu Entwicklungs- bzw. Testzwecken sollten 500 Anfragen pro Minute, je nach Art der Verwendung, ausreichend sein.

5.1. Nachteile der Umsetzung

Hohe Kosten bei kleinen Datenmengen und niedrigen Zugriffszahlen

Die Kosten des Amazons S3 Dienstes ergeben sich aus dem verwendeten Speicher und den Zugriffszahlen auf Daten. Die Tabelle 5.1 zeigt eine Übersicht der monatlich anfallenden Preise pro GB. Die Tabelle 5.2 zeigt eine Übersicht der monatlichen Preise pro 1000 Anforderung. Ausgehend der in den Tabellen gezeigten Preise ist von einer Beispiel Nutzung des S3 Dienstes auszugehen, mit weniger als 100GB genutzten (Standard-)Speicher, sowie weniger als 1.000.000 Zugriffe, auf die in S3 gespeicherten Objekte, pro Monat.

Damit können folgende Maximalpreise errechnet werden:

Tab. 5.1.: Preisübersicht S3 Speicher (Frankfurt)

Quelle: AWS Preisliste, Stand: 30.08.2016

	Standardspeicher	Glacier-Speicherung
Erstes TB pro Monat	\$0.0324 pro GB	\$0.0120 pro GB
Nächste 49 TB pro Monat	\$0.0319 pro GB	\$0.0120 pro GB
Nächste 450 TB pro Monat	\$0.0314 pro GB	\$0.0120 pro GB
Nächste 500 TB pro Monat	\$0.0308 pro GB	\$0.0120 pro GB
Nächste 4 000 TB pro Monat	\$0.0303 pro GB	\$0.0120 pro GB
Über 5 000 TB pro Monat	\$0.0297 pro GB	\$0.0120 pro GB

Tab. 5.2.: Preisübersicht S3 Anforderungen (Frankfurt)

Quelle: AWS Preisliste, Stand: 30.08.2016

Für Anforderungen des Typs	Preise
PUT-, COPY-, POST- oder LIST-Anforderungen	\$0.0054 pro 1 000 Anforderungen
GET und alle anderen Anforderungen	\$0.0043 pro 10 000 Anforderungen
Löschanforderungen	kostenlos

Maximalpreis für verwendeten Speicher: $100(\text{GB}) * 0,0324\$ = 3,24\$$

Maximalpreis für den Zugriff: $1000 * 0,0054\$ = 5,40\$$

Daraus ergibt sich ein maximaler Gesamtbetrag von 8,64\$ pro Monat. Ausgehend von den aktuellen Preisen für das Raspberry Pi - Modell 3 inklusive Zubehör, wie Anschlusskabel, SD-Karte und ggf. Gehäuse ergibt sich ein Preis von ca. 60-70 Euro (Vergleich amazon.com, Stand 30.08.2016). Die Beispiel Rechnung zeigt, dass bei kleinen Datenmengen und niedrigen Zugriffszahlen, die Verwendung von S3 Ninja auf einem Raspberry Pi gegenüber der Verwendung des S3 Dienstes auf Amazon deutlich unwirtschaftlicher sein kann.

Inaktivität und Mangel an Unterstützung in Entwicklergemeinschaften

Wie bereits geschildert, sind einige Funktionen von S3 in der Emulation S3 Ninja nicht implementiert. Leider wird an dem Projekt S3 Ninja nicht weiter Entwickelt. Die letzten Änderungen wurden im Mai 2015 vorgenommen (Stand 18.8.2016).¹

Des weiteren werden in Entwickler-Gemeinschaften, wie Stackoverflow.com nahezu keine Informationen zu S3 Ninja bereit gestellt.

¹S3 Ninja Readme. Aufgerufen am 30.08.2016. URL: <https://github.com/scireum/s3ninja>.

5.2. Vorteile der Umsetzung

Aufwand

Die Installation und Konfiguration eines Raspberry Pis sowie die Installation von S3 Ninja beansprucht wenig Zeit. S3 Ninja kann sofort nach Abschluss der Installation und Konfiguration ohne Weiteres mit den AWS SDKs verwendet werden. In Java oder Python Projekten ist es möglich, durch Konfigurationsdateien und Laufzeitparameter Profile festzulegen. Somit ist es möglich, die selbe Implementation einer S3 Schnittstelle sowohl für S3 Ninja als auch für den S3 Service von AWS zu verwenden.

Der folgende Quellcode zeigt eine Beispiel (Bean-)Konfiguration unter Verwendung des Spring Frameworks in Java:

```
1 @Configuration
2 public class AwsIntegrationConfiguration {
3
4     @Profile("!aws")
5     @Configuration
6     public static class DevelopmentAndTestingAwsIntegrationConfiguration {
7
8         @Bean
9         public AmazonS3 amazonS3Client() {
10             AmazonS3 s3Client = new AmazonS3Client(new AwsCredentialImpl());
11             s3Client.setEndpoint("http://192.168.178.70:9444/s3");
12             s3Client.setS3ClientOptions(S3ClientOptions.builder()
13                 .setPathStyleAccess(true).build());
14             return s3Client;
15         }
16     }
17
18     @Profile("aws")
19     @Configuration
20     public static class ProductiveProfileAwsIntegrationConfiguration {
21
22         @Bean
23         public AmazonS3 amazonS3Client() {
24             return new AmazonS3Client(new AwsCredentialImpl());
25             //with correct AWS Credentials (preferable not stored in code)
26         }
27     }
28 }
```

Spring ist ein Framework, welches das Entwickeln von Java/EE Software erleichtern soll. Das Spring Framework ist, nicht nur auf Grund seiner vielen Konfigurationsmöglichkeiten, ein sehr häufig genutztes Java Framework. Die Annotation `@Configuration` gibt an, dass es sich um eine Klasse zur Konfiguration der Applikation handelt, welche von Spring zur Laufzeit ausgewertet wird. Die `@Profile` annotation stellt sicher, dass eine Konfiguration nur dann verwendet wird, sollte das angegebene Profil verwendet, bzw. nicht verwendet werden. Die beiden statischen Klassen in den Zeilen 8-16 sowie 18-27, können genutzt werden, um die Konfiguration des jeweiligen Profils umzusetzen. In diesem Beispiel wird eine Instanz der Klasse 'AmazonS3Client' mit der S3 Konfiguration für Testzwecke bzw. für den produktiv Betrieb erzeugt. Spring bietet zudem sogenannte Dependency Injection. Es ermöglicht das Injizieren solcher per Configuration erzeugte Klassen mit einer Annotation in anderen Klassen zu verwenden. Die Möglichkeit für diese Art der Konfiguration bietet die Erweiterung für Spring, Spring-Boot. Ohne diese Erweiterung müssten vergleichsweise Komplexe XML-Dokumente zur Konfiguration angelegt werden.

Geringe Kosten bei großen Datenmengen und hohen Zugriffszahlen

Sollte bei der Verwendung von S3 Ninja eine sehr hohe Datenmenge bzw. Datenaustausch-Rate erwartet werden, kann S3 Ninja eine sehr Kostengünstige alternative sein. Betrachtet man das Beispiel aus 5.1, sieht man, dass bei großen Datenmengen das Raspberry Pi einen deutlichen wirtschaftlichen Vorteil erbringen kann. Sollten Datenmengen von über 10Terabyte (TB) erwartet werden, so ist Anhand der Tabellen 5.1 und 5.2 folgender Mindestpreis zu errechnen:

Minimalpreis für Verwendeten Speicher: $10.000(10TB) * 0,0324\$ = 324\$$

Ohne die erwarteten Zugriffszahlen zu berücksichtigen liegt, der Mindestpreis bereits bei 324\$ pro Monat. Diese Rechnung zeigt, dass die Verwendung von S3 Ninja auf einem Raspberry Pi deutlich wirtschaftlicher sein kann, selbst, wenn für den Benötigten Speicheraufwand zusätzliche Festplatten gekauft werden müssen.

Mobilität

Da es sich bei einem Raspberry um einen kleinen Einplatinencomputer handelt, ist es sehr einfach zu transportieren und kann es auch verwendet werden, ohne über eine bestehende Internet Verbindung zu verfügen. Außerdem kann das Raspberry Pi, aufgrund seines geringen Energieverbrauchs, mit einem Akku betrieben werden.

5.3. Alternativen

In Software Projekten gibt es zwei populäre Ansätze zum Testen und Entwickeln mit externen Abhängigkeiten wie Amazon Web Services. So genannte Developer Stacks und Docker sind zwei mögliche Alternativen. Welcher Ansatz für die Umsetzung eines Projektes in Frage kommt, kann von verschiedenen Faktoren abhängig sein.

- erwartete Kosten
- Projektumfang
- Teamgröße
- benötigter Aufwand

5.3.1. Amazon Web Services Developer Stack

Ein Set von Instanzen, welches als Kollektiv genutzt und verwaltet werden soll, bezeichnet man üblicherweise als Stack. In Software Projekten verwendet man häufig eine Vielzahl dieser Stacks für unterschiedlichste Umgebungen. Eine typische Sammlung von Stacks enthält:

- Einen Entwicklungs-Stack, welcher von Entwicklern genutzt wird für das Beheben von Fehlern, Ausführen automatischer Tests oder Entwickeln neuer Funktionen
- Einen Test-Stack, auf welchem Änderungen getestet werden können, bevor sie veröffentlicht werden
- Einen Produktiv-Stack, welcher die bereits veröffentlichte Version eines Endprodukts bereitstellt

2

Diese drei Stacks sind in der Regel bis auf die zur Verfügung gestellte Rechenleistung identisch. Auf diese Weise kann jeder Entwickler mit einem der Produktiv-Umgebung identischen Stack arbeiten und testen.

5.3.2. Docker

Eine weitere Alternative ist Docker. Docker ist eine kostenlose Software, mit dessen Hilfe ganze Sammlungen von Anwendungen durch Virtualisierung bereitgestellt werden können. Eine Sammlung von Anwendungen, werden in Docker als sogenannte Container bereitgestellt. Diese Container können lokal ausgeführt und für die Entwicklung sowie Testen einer Software genutzt werden.³ Auf diese Weise kann S3 ninja ebenfalls zur Verfügung gestellt werden. Ein Container für die Verwendung von S3 ninja innerhalb einer von

²Vgl. *AWS OpsWorks*. Aufgerufen am 30.08.2016. URL: <http://docs.aws.amazon.com/opsworks/latest/userguide/welcome.html>.

³Vgl. *WHAT IS DOCKER?* Aufgerufen am 30.08.2016. URL: <https://www.docker.com/what-docker>.

Docker bereitgestellten Virtualisierung kann unter <https://github.com/66pix/docker-s3ninja> heruntergeladen werden.

5.4. Fazit und Ausblick

In der vorliegenden Arbeit wurde S3 ninja auf dem Raspberry Pi hauptsächlich als Testumgebung betrachtet. Das liegt unter anderem daran, dass S3 Ninja über keine Sicherheitsmechanismen verfügt aber auch daran, dass es lediglich zu Test- und Entwicklungszwecken entwickelt wurde, wie in dem GitHub Repository nachzulesen:

*S3 ninja emulates the Amazon S3 API for development and test purposes.*⁴ S3 Ninja ist ein ideales Werkzeug, für das Testen der AWS S3 Schnittstelle sowie kleinerer nicht öffentlicher Projekte.

Das Raspberry Pi ist auf Grund seines Preises, Flexibilität sowie durch die Möglichkeit es in den verschiedensten Programmiersprachen zu programmieren ein empfehlenswertes Werkzeug. Es bietet sowohl die Möglichkeit als Test- bzw. Experimentierplattform genutzt zu werden, aber auch als solides Endgerät in unzähligen Anwendungsfällen.

In einer weiteren Arbeit, wäre es interessant einen Cluster mit Raspberry Pis als Experimentierplattform zu entwerfen. Auf diesem Cluster könnte man versuchen einige weitere Dienste von AWS als Emulation zu verwenden. Einer der wesentlichsten Aspekte der Cloud Computing Dienstleistung von Amazon, ist die Skalierbarkeit, weshalb es interessant ist einen Cluster für eine weitere Emulation zu verwenden. Auch in diesem Zusammenhang wäre Docker eine Option.

Des Weiteren könnte man S3 Ninja mit einigen Sicherheitsmechanismen ausstatten. S3 Ninja mit einer geeigneten Absicherung der API, könnte mit Hilfe eines Raspberry Pis in kürzester Zeit zu einem Netzwerklaufwerk umfunktioniert werden. So könnte man unter Anderem mit S3cmd lokale Verzeichnisse mit S3 ninja synchronisieren. Der folgende Befehl erlaubt das Synchronisieren mit S3:

```
1 s3cmd sync LOKALES_VERZEICHNIS s3://BUCKET[/PREFIX]
```

Diesen Befehl kann man in definierten Abständen automatisch vom Betriebssystem ausführen lassen.

Abschließend lässt sich sagen, dass die Bedienung von S3 mittels S3cmd nach einigen Konfigurationsschritten problemlos möglich ist. Des Weiteren war die Verwendung

⁴*S3 Ninja Readme*. Aufgerufen am 30.08.2016. URL: <https://github.com/scireum/s3ninja>.

des Amazon SDKs in Python und Java ebenfalls erfolgreich sowie lehrreich. Mit dem Raspberry Pi stand eine hervorragende Lernplattform zur Verfügung.

Literatur

- [1] *Amazon S3 – Preise*. Aufgerufen am 30.08.2016. URL: <https://aws.amazon.com/de/s3/pricing/>.
- [2] *AWS OpsWorks*. Aufgerufen am 30.08.2016. URL: <http://docs.aws.amazon.com/opsworks/latest/userguide/welcome.html>.
- [3] *AWS SDK für Python Dokumentation*. Aufgerufen am 31.08.2016. URL: <https://aws.amazon.com/de/sdk-for-python/>.
- [4] Thomas Barton. *E-Business mit Cloud Computing*. Springer, 2014.
- [5] Christian Baun u. a. *Cloud Computing: Web-basierte dynamische IT-Services (Informatik im Fokus)*. Springer, 2011.
- [6] *Einplatinenrechner (SBC Computer) - zuverlässige Arbeitstiere der Industrie*. Aufgerufen am 30.08.2016. URL: <https://www.distronek.de/computer/>.
- [7] Johannes Ernesti und Peter Kaiser. *Python 3 Das Umfassende Handbuch*. Galileo Computing, 2012.
- [8] *Gabler Wirtschaftslexikon - Cloud Computing*. Aufgerufen am 15.08.2016. URL: <http://wirtschaftslexikon.gabler.de/Definition/cloud-computing.html>.
- [9] *Getting Started with Amazon Simple Storage Service*. Aufgerufen am 10.08.2016. URL: <http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>.
- [10] Bernard Golden. *Amazon Web Services FOR DUMMIES*. Wiley, 2013.
- [11] Daniel Kampert. *Raspberry Pi - Der praktische Einstieg*. Galileo Computing, 2014.
- [12] Michael Kofler, Charly Kühnast und Christoph Scherbeck. *Raspberry Pi: Das umfassende Handbuch. Komplett in Farbe - inkl. Schnittstellen, Schaltungsaufbau, Steuerung mit Python u.v.m.* Rheinwerk, 2015.
- [13] *Raspberry Pi 3 Model B*. Aufgerufen am 30.08.2016. URL: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.
- [14] *Raspbian Website*. Aufgerufen am 10.08.2016. URL: <https://www.raspbian.org>.
- [15] *RFC: Hypertext Transfer Protocol Version 2 (HTTP/2)*. Aufgerufen am 14.08.2016. URL: <https://tools.ietf.org/html/rfc7540>.
- [16] *RFC1918*. Aufgerufen am 30.08.2016. URL: <https://tools.ietf.org/html/rfc1918>.

- [17] Leonard Richardson. *RESTful Web APIs*. O'Reilly Media, 2013.
- [18] *S3 Ninja Readme*. Aufgerufen am 30.08.2016. URL: <https://github.com/scireum/s3ninja>.
- [19] *S3 Ninja Webseite*. Aufgerufen am 14.08.2016. URL: <http://s3ninja.net>.
- [20] s3tools.org. *s3cmd Manual*. s3cmd 1.6.1.
- [21] *SD Card Speed Classes, Grades, Bus Modes, and File Systems Explained*. Aufgerufen am 30.08.2016. URL: <http://www.pcper.com/reviews/Editorial/SD-Card-Speed-Classes-Grades-Modes-and-File-Systems-Explained>.
- [22] Daniel Stenberg. *Curl Manual*. Curl 7.40.0. 2014.
- [23] Jim Webber, Savas Parastatidis und Ian Robinson. *REST in Practice Hypermedia and Systems Architecture*. O'Reilly, 2011.
- [24] *Welcher Raspberry Pi? Alle Modelle im Vergleich*. Aufgerufen am 30.08.2016. URL: http://praxistipps.chip.de/welcher-raspberry-pi-alle-modelle-im-vergleich_41923.
- [25] *WHAT IS DOCKER?* Aufgerufen am 30.08.2016. URL: <https://www.docker.com/what-docker>.
- [26] *XML/Regeln/XML-Deklaration*. Aufgerufen am 15.08.2016. URL: <https://wiki.selfhtml.org/wiki/XML/Regeln/XML-Deklaration>.

A. Beispiel Anwendung mit Spring - Rückgabe eines XML Dokuments (Autowerkstatt)

A.1. Car Model

Listing A.1: car model

```
1 package de.thesis.persistance;
2
3 import javax.xml.bind.annotation.XmlAccessType;
4 import javax.xml.bind.annotation.XmlAccessorType;
5 import javax.xml.bind.annotation.XmlRootElement;
6 import java.io.Serializable;
7
8 /**
9  * Created by daniel on 15.08.16.
10 * <p>
11 * This is a Basic Car Model
12 */
13 @XmlRootElement(name = "car")
14 @XmlAccessorType(XmlAccessType.PROPERTY)
15 public class Car implements Serializable {
16
17     private static final long serialVersionUID = 1L;
18
19     private String licensePlate;
20     private Integer initialRegistration;
21     private Integer mileage;
22     private String model;
23
24
25     public String getLicensePlate() {
```

```
26     return licensePlate;
27 }
28
29 public void setLicensePlate(String licensePlate) {
30     this.licensePlate = licensePlate;
31 }
32
33 public Integer getInitialRegistration() {
34     return initialRegistration;
35 }
36
37 public void setInitialRegistration(Integer initialRegistration) {
38     this.initialRegistration = initialRegistration;
39 }
40
41 public Integer getMileage() {
42     return mileage;
43 }
44
45 public void setMileage(Integer mileage) {
46     this.mileage = mileage;
47 }
48
49 public String getModel() {
50     return model;
51 }
52
53 public void setModel(String model) {
54     this.model = model;
55 }
56
57 @Override
58 public String toString() {
59     return "Car [licensePlate=" + licensePlate + ", model=" + model
60         + ", mileage=" + mileage + ", initialRegistration=" +
61     initialRegistration + "];"
62 }
```

A.2. Car List Wrapper

Listing A.2: car list wrapper

```
1 package de.thesis.persistance;
2
3 import javax.xml.bind.annotation.XmlRootElement;
4 import java.io.Serializable;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 /**
9  * Created by daniel on 15.08.16.
10 * <p>
11 * This class basically wraps a list of car models
12 */
13 @XmlRootElement(name = "cars")
14 public class CarList implements Serializable {
15
16     private static final long serialVersionUID = 1L;
17
18     private List<Car> cars = new ArrayList<>();
19
20     public List<Car> getCars() {
21         return cars;
22     }
23
24     public void setCars(List<Car> employees) {
25         this.cars = employees;
26     }
27
28 }
```

A.3. Car Controller

Listing A.3: car controller

```
1 package de.thesis.webapi;
2
```

```
3 import de.thesis.persistance.Car;
4 import de.thesis.persistance.CarList;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestMethod;
8 import org.springframework.web.bind.annotation.ResponseStatus;
9 import org.springframework.web.bind.annotation.RestController;
10
11 /**
12  * Created by daniel on 15.08.16.
13  * <p>
14  * This example controller provides basic information about car models
15  */
16 @RestController
17 public class CarController {
18
19     private CarList cars;
20     private Car carA;
21     private Car carB;
22
23     private CarList getCars() {
24         cars = new CarList();
25         carA = new Car();
26         carA.setInitialRegistration(1990);
27         carA.setLicensePlate("F-A-123");
28         carA.setMileage(200000);
29         carA.setModel("Volkswagen T4");
30
31         carB = new Car();
32         carB.setInitialRegistration(1991);
33         carB.setLicensePlate("F-B-456");
34         carB.setMileage(300000);
35         carB.setModel("Porsche 911");
36
37         cars.getCars().add(carB);
38         cars.getCars().add(carA);
39
40         return cars;
41     }
42 }
```

```
43 @RequestMapping(method = RequestMethod.GET, value = "/cars")
44 @ResponseStatus(HttpStatus.OK)
45 public CarList readAll() {
46     return getCars(); //returns all cars as list in a
XML-Document
47 }
48
49 }
```

A.4. Spring Main Klasse

Listing A.4: spring main class

```
1 package de.thesis;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 /*
7 * Spring main class - Automatically creates configurations.
8 * Starts up the embedded tomcat an the application.
9 * */
10 @SpringBootApplication
11 public class DemoApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(DemoApplication.class, args);
15     }
16 }
```


B. IntelliJ und Maven

B.1. IntelliJ Maven Assistent Schritt 1

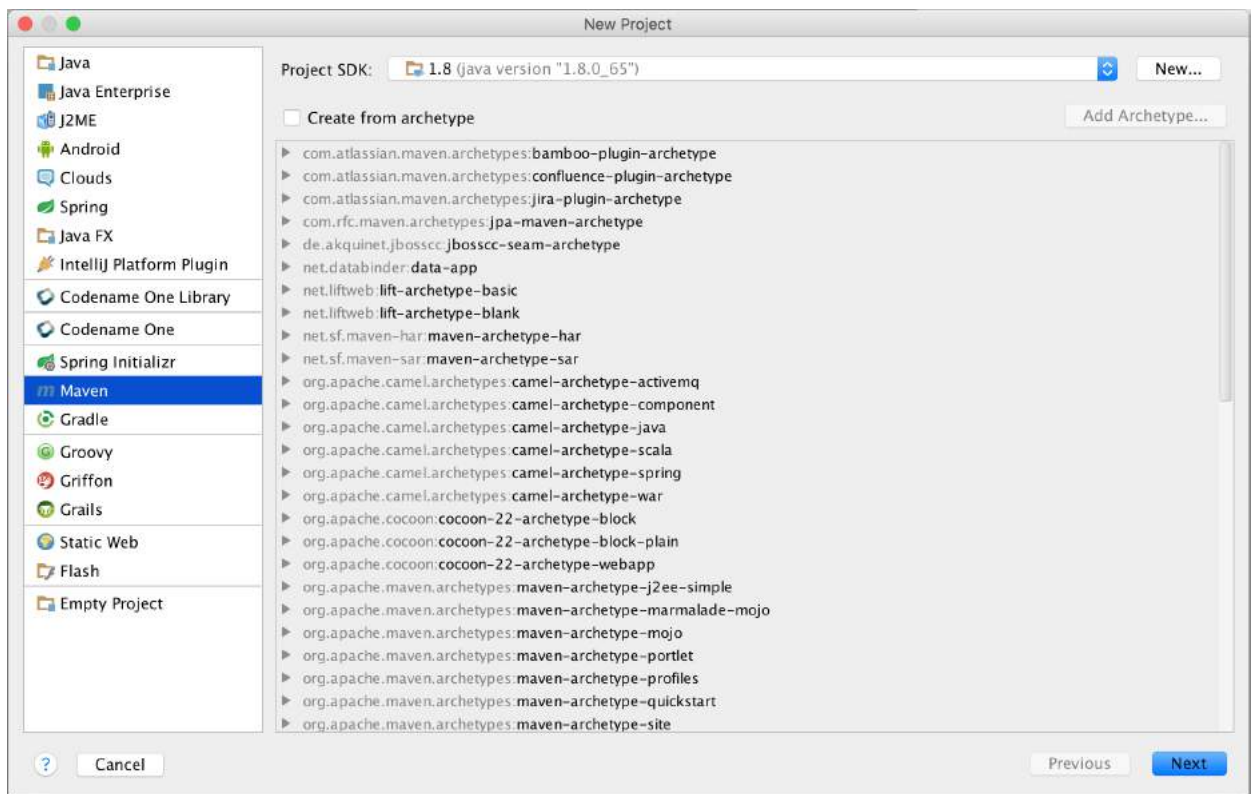


Bild B.1.: IntelliJ Maven Assistent Schritt 1

B.2. IntelliJ Maven Assistent Schritt 2

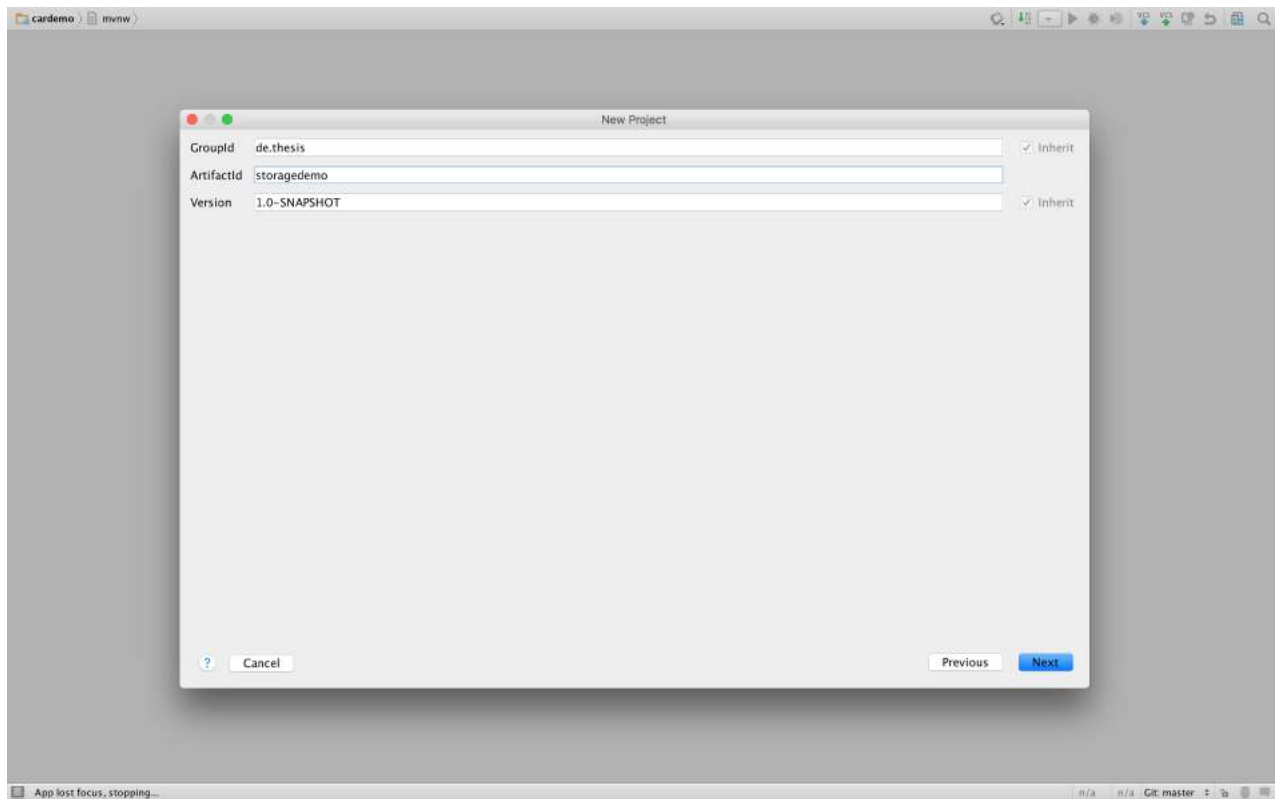


Bild B.2.: IntelliJ Maven Assistent Schritt 2

B.3. IntelliJ Projekt Struktur

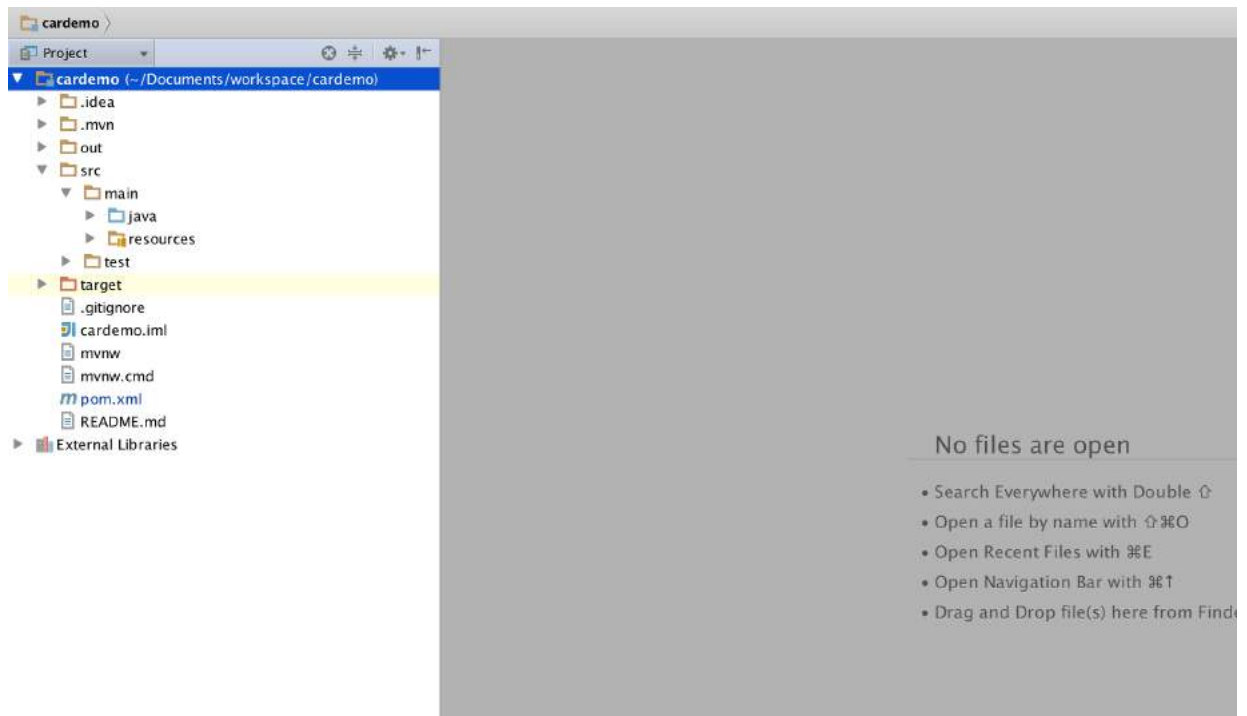


Bild B.3.: IntelliJ Projekt Struktur

C. Java SDK Beispiel

C.1. Interface Java Service

Listing C.1: S3Service Interface

```
1 package de.thesis;
2
3 import com.amazonaws.services.s3.model.Bucket;
4 import com.amazonaws.services.s3.model.S3Object;
5 import com.amazonaws.services.s3.model.S3ObjectSummary;
6
7 import java.io.File;
8 import java.util.List;
9
10 /**
11  * Created by daniel on 24.08.16.
12  */
13 public interface S3Service {
14
15     S3Object createObject(String bucketName, String key, File file) throws
16     IOException;
17
18     S3Object getObject(String bucketName, String key);
19
20     List<S3ObjectSummary> listObjects(String bucketName);
21
22     List<Bucket> listBuckets();
23
24     Bucket createBucket(String bucketName);
25
26     void deleteBucket(String bucketName);
27
28     void deleteObject(String bucketName, String key);
29 }
```

C.2. Implementierung der S3Service Schnittstelle

Listing C.2: SS3ServiceImpl

```
1 package de.thesis;
2
3 import com.amazonaws.services.s3.AmazonS3;
4 import com.amazonaws.services.s3.model.Bucket;
5 import com.amazonaws.services.s3.model.GetObjectRequest;
6 import com.amazonaws.services.s3.model.S3Object;
7 import com.amazonaws.services.s3.model.S3ObjectSummary;
8
9 import java.io.File;
10 import java.util.Date;
11 import java.util.List;
12
13 /**
14  * Created by daniel on 22.08.16.
15  * <p>
16  * This Service provides the basic S3 functions:
17  * - put object
18  * - get object
19  * - list objects
20  * - delete object
21  * - list buckets
22  * - create bucket
23  * - delete bucket
24  * <p>
25  * Usage: see S3ServiceImplIT and BucketLifecycleTest
26  */
27 public class S3ServiceImpl implements S3Service {
28
29     private final AmazonS3 s3Client;
30
31
32     public S3ServiceImpl(AmazonS3 s3Client) {
```

```
33     this.s3Client = s3Client;
34 }
35
36 //upload a file to a bucket
37 public S3Object createObject(String bucketName, String key, File file) {
38     s3Client.putObject(bucketName, key, file);
39     return s3Client.getObject(bucketName, key);
40 }
41
42 //downlaod a file from a bucket
43 public S3Object getObject(String bucketName, String key) {
44     return s3Client.getObject(new GetObjectRequest(bucketName, key));
45 }
46
47 //list all objects of a bucket
48 public List<S3ObjectSummary> listObjects(String bucketName) {
49     return s3Client.listObjects(bucketName).getObjectSummaries();
50 }
51
52 //list all buckets
53 public List<Bucket> listBuckets() {
54     return s3Client.listBuckets();
55 }
56
57 //create a new bucket
58 public Bucket createBucket(String bucketName) {
59     return s3Client.createBucket(bucketName);
60 }
61
62 //delete a bucket
63 public void deleteBucket(String bucketName) {
64     s3Client.deleteBucket(bucketName);
65 }
66
67 //delete object(file) from bucket
68 public void deleteObject(String bucketName, String key) {
69     s3Client.deleteObject(bucketName, key);
70 }
71
72 public String getPresignedUrl(String bucketName, String key) {
```

```
73     return s3Client.generatePresignedUrl(bucketName, key, new
Date()).toString();
74     }
75 }
```

C.3. AWSCredential Beispiel

Listing C.3: "AWSCredentialImpl"

```
1 package de.thesis;
2
3 import com.amazonaws.auth.AWSCredentials;
4
5 /**
6  * Created by daniel on 22.08.16.
7  * <p>
8  * This is just a minimum implementation of the AWS Credential Class.
9  * Provided key ares random generated. They don't belong to
10 * a real account.
11 * <p>
12 * NOTE!: Use this class only for testing purposes!!!
13 */
14 public class AwsCredentialImpl implements AWSCredentials {
15
16     public String getAWSAccessKeyId() {
17         return "AKIAIOSFODNN7EXAMPLE";
18     }
19
20     public String getAWSSecretKey() {
21         return "wJalrXUtnFEMI/K7MDENG/bPxrRfiCYEXAMPLEKEY";
22     }
23 }
```

C.4. S3Service Ende-Zu-Ende Test

Listing C.4: "SS3ServiceImplIT"

```
1 package de.thesis.e2e;
```

```
2
3 import com.amazonaws.services.s3.AmazonS3;
4 import com.amazonaws.services.s3.AmazonS3Client;
5 import com.amazonaws.services.s3.S3ClientOptions;
6 import com.amazonaws.services.s3.model.Bucket;
7 import com.amazonaws.services.s3.model.S3Object;
8 import com.amazonaws.services.s3.model.S3ObjectSummary;
9 import de.thesis.AwsCredentialImpl;
10 import de.thesis.S3ServiceImpl;
11 import org.junit.Before;
12 import org.junit.Test;
13
14 import java.io.File;
15 import java.util.ArrayList;
16 import java.util.List;
17
18 import static org.junit.Assert.assertEquals;
19 import static org.junit.Assert.assertTrue;
20
21 /**
22  * Created by daniel on 22.08.16.
23  * <p>
24  * This is an integration test, or even more an e2e test, that
25  * verifies, S3ServiceImpl is working correctly with S3 Ninja.
26  * It can even be used with the AWS S3 Api.
27  * In that case the reverse proxy has to be disabled and correct
28  * key have to be setup in AwsCredentialImpl.
29  * <p>
30  * Note!: The files an Buckets used in this test need to be set up
31  * correctly.
32  * Maven lifecycle can be used to migrate the correct data in test scope.
33  */
34 public class S3ServiceImplIT {
35
36     private S3ServiceImpl s3Service;
37
38     @Before
39     public void setUp() throws Exception {
40         AmazonS3 s3Client = new AmazonS3Client(new AwsCredentialImpl());
41         //s3Client.setRegion(Region.getRegion(Regions.EU_CENTRAL_1));
```



```
41     s3Client.setEndpoint("http://192.168.178.70:9444/s3"); //S3 Ninja
Endpoint
42
43     s3Client.setS3ClientOptions(S3ClientOptions.builder().setPathStyleAccess(true).bu
44     s3Service = new S3ServiceImpl(s3Client);
45     }
46     @Test
47     public void testCreateObject() {
48         S3Object object = s3Service.createObject("test", "test.txt", new
File("src/test/test.txt"));
49         assertEquals("test.txt", object.getKey());
50     }
51
52     @Test
53     public void testGetObject() throws Exception {
54         S3Object object = s3Service.getObject("test", "Pin1.jpg");
55         System.out.println();
56         assertEquals(object.getKey(), "Pin1.jpg");
57     }
58
59     @Test
60     public void testListObjects() throws Exception {
61         List<S3ObjectSummary> summaries = s3Service.listObjects("test");
62         assertTrue(summaries.size() >= 1);
63     }
64
65     @Test
66     public void testListBuckets() throws Exception {
67         List<Bucket> buckets = s3Service.listBuckets();
68         List<String> names = new ArrayList<String>();
69         for (Bucket bucket : buckets) {
70             names.add(bucket.getName());
71         }
72         assertTrue(names.contains("test"));
73     }
74
75     @Test
76     public void testGetPresignedUrl() throws Exception {
77         assertTrue(s3Service.getPresignedUrl("test", "test")
```

```
78         .contains("http://192.168.178.70:9444/s3/test/test"));
79
80     }
81 }
```

D. Beispiel Konfiguration für S3cmd

Listing D.1: s3cmdconf

```
1 default_mime_type = binary/octet-stream
2 delay_updates = False
3 delete_after = False
4 delete_after_fetch = False
5 delete_removed = False
6 dry_run = False
7 enable_multipart = True
8 encoding = UTF-8
9 encrypt = False
10 expiry_date =
11 expiry_days =
12 expiry_prefix =
13 follow_symlinks = False
14 force = False
15 get_continue = False
16 gpg_command = None
17 gpg_decrypt = %(gpg_command)s -d --verbose --no-use-agent --batch --yes
    --passphrase-fd %(passphrase_fd)s -o %(output_file)s %(input_file)s
18 gpg_encrypt = %(gpg_command)s -c --verbose --no-use-agent --batch --yes
    --passphrase-fd %(passphrase_fd)s -o %(output_file)s %(input_file)s
19 gpg_passphrase =
20 guess_mime_type = True
21 host_base = s3.eu-central-1.amazonaws.com
22 host_bucket = s3.eu-central-1.amazonaws.com
23 human_readable_sizes = False
24 invalidate_default_index_on_cf = False
25 invalidate_default_index_root_on_cf = True
26 invalidate_on_cf = False
27 kms_key =
28 limitrate = 0
29 list_md5 = False
```

D. Beispiel Konfiguration für S3cmd

```
30 log_target_prefix =
31 long_listing = False
32 max_delete = -1
33 mime_type =
34 multipart_chunk_size_mb = 15
35 multipart_max_chunks = 10000
36 preserve_attrs = True
37 progress_meter = True
38 proxy_host =
39 proxy_port = 0
40 put_continue = False
41 recursive = False
42 recv_chunk = 65536
43 reduced_redundancy = False
44 requester_pays = False
45 restore_days = 1
46 secret_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
47 send_chunk = 65536
48 server_side_encryption = False
49 signature_v2 = False
50 simpledb_host = sdb.amazonaws.com
51 skip_existing = False
52 socket_timeout = 300
53 stats = False
54 stop_on_error = False
55 storage_class =
56 urlencoding_mode = normal
57 use_https = False
58 use_mime_magic = True
59 verbosity = WARNING
60 website_endpoint = http://%(bucket)s.s3-website-%(location)s.amazonaws.com/
61 website_error =
62 website_index = index.html
```

E. S3 ninja Bucket Klasse

Listing E.1: "Bucket

```
1  /*
2  * Made with all the love in the world
3  * by scireum in Remshalden, Germany
4  *
5  * Copyright by scireum GmbH
6  * http://www.scireum.de - info@scireum.de
7  */
8
9  package ninja;
10
11 import com.google.common.collect.Lists;
12 import com.google.common.hash.Hashing;
13 import sirius.kernel.cache.Cache;
14 import sirius.kernel.cache.CacheManager;
15 import sirius.kernel.cache.ValueComputer;
16 import sirius.kernel.commons.Strings;
17 import sirius.kernel.health.Counter;
18 import sirius.kernel.health.Exceptions;
19 import sirius.kernel.xml.Attribute;
20 import sirius.kernel.xml.XMLStructuredOutput;
21
22 import javax.annotation.Nonnull;
23 import javax.annotation.Nullable;
24 import java.io.File;
25 import java.io.FileOutputStream;
26 import java.io.IOException;
27 import java.nio.file.FileVisitResult;
28 import java.nio.file.Files;
29 import java.nio.file.Path;
30 import java.nio.file.SimpleFileVisitor;
31 import java.nio.file.attribute.BasicFileAttributes;
```

```
32 import java.util.List;
33
34 /**
35  * Represents a bucket.
36  * <p>
37  * Internally a bucket is just a directory within the base directory.
38  * </p>
39  */
40 public class Bucket {
41
42     private File file;
43     private static Cache<String, Boolean> publicAccessCache =
44     CacheManager.createCache("public-bucket-access");
45
46     /**
47      * Creates a new bucket based on the given directory.
48      *
49      * @param file the directory which stores the contents of the bucket.
50      */
51     public Bucket(File file) {
52         this.file = file;
53     }
54
55     /**
56      * Returns the name of the bucket.
57      *
58      * @return the name of the bucket
59      */
60     public String getName() {
61         return file.getName();
62     }
63
64     /**
65      * Deletes the bucket and all of its contents.
66      */
67     public void delete() {
68         for (File child : file.listFiles()) {
69             child.delete();
70         }
71         file.delete();
72     }
73 }
```

```
71     }
72
73     /**
74      * Creates the bucket.
75      * <p>
76      * If the underlying directory already exists, nothing happens.
77      * </p>
78      */
79     public void create() {
80         if (!file.exists()) {
81             file.mkdirs();
82         }
83     }
84
85     /**
86      * Returns a list of all stored objects
87      *
88      * @return a list of all objects in the bucket.
89      */
90     public List<StoredObject> getObjects() {
91         List<StoredObject> result = Lists.newArrayList();
92         for (File child : file.listFiles()) {
93             if (child.isFile() && !child.getName().startsWith("__")) {
94                 result.add(new StoredObject(child));
95             }
96         }
97
98         return result;
99     }
100
101     /**
102      * Returns a list of at most the provided number of stored objects
103      *
104      * @param output the xml structured output the list of objects should
105      * be written to
106      * @param limit controls the maximum number of objects returned
107      * @param marker the key to start with when listing objects in a bucket
108      * @param prefix limits the response to keys that begin with the
109      * specified prefix
110      */
```

```
109     public void outputObjects(XMLStructuredOutput output, int limit,
110     @Nullable String marker, @Nullable String prefix) {
111         ListFileTreeVisitor visitor = new ListFileTreeVisitor(output,
112         limit, marker, prefix);
113
114         output.beginOutput("ListBucketResult", Attribute.set("xmlns",
115         "http://s3.amazonaws.com/doc/2006-03-01/"));
116         output.property("Name", getName());
117         output.property("MaxKeys", limit);
118         output.property("Marker", marker);
119         output.property("Prefix", prefix);
120         try {
121             Files.walkFileTree(file.toPath(), visitor);
122         } catch (IOException e) {
123             Exceptions.handle(e);
124         }
125         output.property("IsTruncated", limit > 0 && visitor.getCount() >
126         limit);
127         output.endOutput();
128     }
129
130     /**
131     * Visits all files in the buckets directory and outputs their metadata
132     * to an {@link XMLStructuredOutput}.
133     */
134     private static class ListFileTreeVisitor extends
135     SimpleFileVisitor<Path> {
136
137         Counter objectCount;
138         XMLStructuredOutput output;
139         int limit;
140         String marker;
141         String prefix;
142         boolean useLimit;
143         boolean usePrefix;
144         boolean markerReached;
```



```
143         @Nullable String prefix) {
144             this.output = output;
145             this.limit = limit;
146             this.marker = marker;
147             this.prefix = prefix;
148             objectCount = new Counter();
149             useLimit = limit > 0;
150             usePrefix = Strings.isFilled(prefix);
151             if (usePrefix) {
152                 this.prefix = prefix.replace('/', '_');
153             }
154             markerReached = Strings.isEmpty(marker);
155         }
156
157         @Override
158         public FileVisitResult visitFile(Path path, BasicFileAttributes
159         attrs) throws IOException {
160             File file = path.toFile();
161             String name = file.getName();
162
163             if (!file.isFile() || name.startsWith("__")) {
164                 return FileVisitResult.CONTINUE;
165             }
166             if (!markerReached) {
167                 if (marker.equals(name)) {
168                     markerReached = true;
169                 }
170             } else {
171                 StoredObject object = new StoredObject(file);
172                 if (!usePrefix || name.startsWith(prefix)) {
173                     if (useLimit) {
174                         long numObjects = objectCount.inc();
175                         if (numObjects <= limit) {
176                             output.beginObject("Contents");
177                             output.property("Key", file.getName());
178                             output.property("LastModified",
179 S3Controller.ISO_INSTANT.format(object.getLastModifiedInstant()));
180                             output.property("Size", file.length());
181                             output.property("StorageClass", "STANDARD");
```

```
181
182         String etag = null;
183         try {
184             etag =
com.google.common.io.Files.hash(file, Hashing.md5()).toString();
185         } catch (IOException e) {
186             Exceptions.ignore(e);
187         }
188         output.property("ETag", etag);
189         output.endObject();
190     } else {
191         return FileVisitResult.TERMINATE;
192     }
193 }
194 }
195 }
196     return FileVisitResult.CONTINUE;
197 }
198
199     public long getCount() {
200         return objectCount.getCount();
201     }
202 }
203
204 /**
205  * Determines if the bucket is private or public accessible
206  *
207  * @return <tt>>true</tt> if the bucket is public accessible,
208  * <tt>>false</tt> otherwise
209  */
210     public boolean isPrivate() {
211         return !publicAccessCache.get(getName(), new ValueComputer<String,
Boolean>() {
212             @Nullable
213             @Override
214             public Boolean compute(@Nonnull String key) {
215                 return getPublicMarkerFile().exists();
216             }
217         });
218     }
219 }
```

```
218
219 private File getPublicMarkerFile() {
220     return new File(file, "__ninja_public");
221 }
222
223 /**
224  * Marks the bucket as private accessible.
225  */
226 public void makePrivate() {
227     if (getPublicMarkerFile().exists()) {
228         getPublicMarkerFile().delete();
229         publicAccessCache.put(getName(), false);
230     }
231 }
232
233 /**
234  * Marks the bucket as public accessible.
235  */
236 public void makePublic() {
237     if (!getPublicMarkerFile().exists()) {
238         try {
239             new FileOutputStream(getPublicMarkerFile()).close();
240         } catch (IOException e) {
241             throw Exceptions.handle(Storage.LOG, e);
242         }
243     }
244     publicAccessCache.put(getName(), true);
245 }
246
247 /**
248  * Returns the underlying directory as File.
249  *
250  * @return a <tt>File</tt> representing the underlying directory
251  */
252 public File getFile() {
253     return file;
254 }
255
256 /**
257  * Determines if the bucket exists.
```

```
258     *
259     * @return <tt>true</tt> if the bucket exists, <tt>>false</tt> otherwise
260     */
261     public boolean exists() {
262         return file.exists();
263     }
264
265     /**
266     * Returns the child object with the given id.
267     *
268     * @param id the name of the requested child object. Must not contain
269     * .. / or \
270     * @return the object with the given id, might not exist, but is always
271     * non null
272     */
273     public StoredObject getObject(String id) {
274         if (id.contains("..") || id.contains("/") || id.contains("\\")) {
275             throw Exceptions.createHandled()
276                 .withSystemErrorMessage(
277                     "Invalid object name: %s. A object name
278                     must not contain '..' '/' or '\\'",
279                     id)
280                 .handle();
281         }
282         return new StoredObject(new File(file, id));
283     }
284 }
```

1

¹S3 Ninja Readme. Aufgerufen am 30.08.2016. URL: <https://github.com/scireum/s3ninja>.