HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins INSTITUTE FOR
NETWORKED SOLUTIONS

SDDC
Software Defined Data Center

Studienarbeit

Hochschule für Technik Rapperswil
Institute for Networked Solutions

Herbstsemester 2015

Autoren: Silvan Adrian, Fabian Binna
Betreuender Dozent: Prof. Beat Stettler
Betreuer: Urs Baumann
Projektpartner: Institute for Networked Solutions

Abstract

Unter „Software Defined“ versteht man die Zentralisierung der Intelligenz in Kontrollern. Gerade moderne Data Center werden immer häufiger von Software Kontrollern gesteuert, damit die Dynamik der Bereitstellung von neuen Services massiv erhöht werden kann. Ziel ist, die Ressourcen Storage, Network und Compute abstrahiert als skalierbare Pools der „Service Ebene“ zur Verfügung zu stellen.
Eine RESTful API, die den Umgang mit Services, die wiederum Pakete von Ressourcen darstellen, sorgt für einen zentralen Punkt, an den diverse Systeme und Business Applikationen anknüpfen können. Damit die breite Auswahl von Libraries und Produkten in einem Data Center angesprochen werden kann, verwaltet eine generische API die Kontroller und ermöglicht den abstrakten Umgang mit Ressourcen. Die beiden abstrakten Ebenen, RESTful API und generische API, werden mit einem Workflow verbunden. Der Workflow kümmert sich um den zeitlich korrekten Ablauf der Instanziierung. Die Software kann als Webservice in einem Docker Container ausgerollt werden und benötigt danach nur noch eine Konfiguration der generischen API und der Kontroller.

Inhaltsverzeichnis

Teil I

Einführung

1 Management Summary

SDDC

Software Defined Data Center

Ausgangslage

Unter „Software Defined“ versteht man die Zentralisierung der Intelligenz in Kontrollern. Gerade moderne Data Center werden immer häufiger von Software Kontrollern gesteuert, damit die Dynamik der Bereitstellung von neuen Services massiv erhöht werden kann. Ziel ist, die Ressourcen Storage, Network und Compute abstrahiert als skalierbare Pools der „Service Ebene“ zur Verfügung zu stellen. Dies soll als eine generische Middleware/Application Programming Interface (API) realisiert werden, um verschiedene Kontroller möglichst einfach in Business Applikationen zu integrieren.

Vorgehen/Technologie

Nach der Analyse verschiedenster Cloud Provider und Libraries wurde eine möglichst systemunabhängige Schnittstelle definiert. Damit dies überhaupt möglich ist, musste ein strikt einzuhaltender Contract erstellt werden. Ein Modul beinhaltet die Konfiguration für eine Ressource und beim erstellen einer Ressource wird ein Identifier zurückgegeben, wobei Modul und Identifier völlig abstrakt sind. Damit die Software in Business Applikationen integriert werden kann wurde eine RESTful API definiert, die Services abstrahiert. Diese beiden Schnittstellen müssen mit einem Workflow zusammengefügt werden. Der Workflow bestimmt, bei Bestellung eines Services, welche Module zum Einsatz kommen und persistiert die Identifier, um die erstellten Ressourcen später wieder anzusprechen.

Ergebnis

Das Resultat ist ein Webservice, der mit Hilfe von Spring implementiert wurde. Die Datenhaltung läuft auf Postgres und wird über ein OR-Mapping an die Domain angehängt. Der Webservice kann über die RESTful API angesprochen werden. Die generische API kann in einer Extensible Markup Language (XML) Datei definiert werden. Beim Start des Webservices wird die generische API mit Spring Dependency Injection zusammengesetzt. So kann die generische API beliebig mit neuen Kontrollern erweitert und an den Webservice angehängt werden. Für Demonstrationszwecke entstand eine Webseite, die es erlaubt Services zu bestellen, anzusehen, zu bearbeiten und wieder zu löschen.

2 Aufgabenstellung

Unter “Software Defined” versteht man die Zentralisierung der Intelligenz in Kontrollern. Gerade moderne Data Center werden immer mehr von Software Kontrollern gesteuert, damit die Dynamik der Bereitstellung von neuen Services massiv erhöht werden kann. So gibt es bereits Kontroller für Storage, Netzwerk und Compute Ressourcen.

Ziel ist es, die Ressourcen Storage, Network und Compute abstrahiert als skalierbare Pools der “Service Ebene” zu Verfügung zu stellen. Alle modernen Kontroller können über API’s angesprochen werden, allerdings unterscheiden sich hier die verschiedenen Hersteller zum Teil stark.

Ziel dieser Arbeit ist die Entwicklung einer generischen Middleware/API, um verschiedene Kontroller möglichst einfach in Business Applikationen zu integrieren. Nach der Definition einer systemunabhängigen Schnittstelle sollen die verschiedenen Kontroller danach als Treiber an die API angehängt werden können. Zur Demonstrationszwecken soll eine rudimentäre, ca. 3 Seitige Webpage erstellt werden, welche die erstelle API benützt. Dabei soll je mind. ein Storage, Compute und Network Kontroller eingebunden werden.

Teil II

Technischer Bericht

3 Analyse

3.1 Ist-Situation

3.1.1 API

Es gibt einige Beispiele von generischen APIs, welche zwar mehrere Anbieter ansteuern können, jedoch wird meistens zwischen Public und Private Cloud Anbietern APIs unterschieden wodurch keine Hybrid Services möglich sind. Zudem wird bei Public Cloud Anbietern vielmals nur zwischen Storage und Compute unterschieden und Network Service Angebote sind eher selten. Ausnahmen sind hier Clouds DNS oder ein Cloud Loadbalancer. Für die Private Clouds hat sich bisher OpenStack und CloudStack durchgesetzt, daher sind Hybrid Services nicht einfach so möglich. Ebenfalls ist es nicht gerade einfach OpenStack oder CloudStack aufzusetzen und der Aufwand ist für manche KMUs zu gross oder zu teuer.

3.1.2 Dashboard

Im Dashboard Bereich gibt es nicht gerade viele Angebote, einer der bekanntesten ist Bitnami. Bitnami erlaubt es einem seine Cloud Instanzen an einem Ort zu managen, wodurch das handhaben von vielen verschiedenen Instanzen einiges vereinfacht wird. Jedoch fehlt hier völlig die Unterstützung für Private Clouds und bietet auch keine Self Hosted Lösung an, um die eigene Umgebung einzubauen.

3.2 Soll-Situation

3.2.1 API

- Die API sollte auf einem möglichst stabilen Stand sein.
- Es müssen die wichtigsten Provider zur Verfügung stehen.
- Die API muss gut dokumentiert sein.
- Es sollen verschiedene Services angesprochen werden können (Compute, Storage, Network...).

- Keine grosse Einarbeitung, das heisst die Programmiersprache sollte nicht komplett neu sein.

Zudem sind alle Eingenschaften, die das Implementieren der Software erleichtern ein Pluspunkt. Von Vorteil wären zusätzliche Funktionen wie z.B SSL oder Pricing.

3.2.2 User-Dashboard

Das User-Dashboard soll eine Möglichkeit für Benutzer bieten, um Services abonnieren zu können. Dabei soll sowohl Infrastructure as a Service (IaaS), Platfrom as a Service (PaaS) oder Software as a Service (SaaS) abonniert werden können und eine Auswahl unter vielen verschiedenen Cloud Anbietern bieten. Dabei soll der User zwischen einzelnen Angeboten der Anbieter spezifischen Services wählen können bspw.: bei Google Cloud: Cloud DNS, Loadbalancer etc. Es kann daher sein, dass nicht alle Anbieter die gleichen Services bieten und daher die Services mehr oder weniger Anbieter spezifisch sind.

SDDC

Unser Projekt soll deshalb eine einiges generische Möglichkeit bieten, um Services abonnieren zu können und möglichst viele Cloud Anbieter zu unterstützen. Dies soll möglich werden indem ein Dashboard eine RESTful API anspricht und die Generic API alle Schritte durchführt, die nötig sind für die Erstellung des Services.

3.2.3 Admin-Dashboard

Dem Admin soll eine Möglichkeit geboten werden um die Software administrieren zu können.

3.3 API Analyse

3.3.1 Libcloud[1]

Sprache: Python
Wichtigste Provider: Rackspace, Amazon web services, CloudStack, OpenStack, DigitalOcean, Eucalyptus, Joyent, Linode, exoscale,NephoScale, Google Cloud Platform, Zerigo, CloudSigma, iKoula, KVM,XEN,VMWare ESX

3.3.2 jClouds[2]

Sprache: Java
Wichtigste Provider: OpenStack, Docker, DigitalOcean, Google Cloud Platform, Rackspace, HP Cloud, CloudStack, Amazon web services, abiquo, CloudSigma, joyent

3.3.3 elibcloud[3]

Sprache: Erlang
elibcloud ist ein Wrapper für libcloud.

3.3.4 fog[4]

Sprache: Ruby
Wichtigste Provider: CloudSigma, CloudStack, GoGrid, Google Cloud Platform, Joyent, Libvirt, Linode, OpenStack, OpenVZ, Rackspace, Zerigo, IBM, HP

3.3.5 pkgcloud[5]

Sprache: JavaScript (Node.js)
Wichtigste Provider: Amazon, Azure, DigitalOcean, Joyent, OpenStack, Rackspace, Google, HP

3.3.6 libvirt[6]

Sprache: C
Wichtigste Provider: Xen, KVM, OpenVZ, VMware ESX, VirtualBox, IBM PowerVM

3.4 Support

3.4.1 Compute

Die grösste Auswahl an Providern liefert Libcloud.JClouds hingegen unterstützt auch Docker, was ein grosser Vorteil gegenüber Libcloud ist. Im Abschnitt 3.5.6 wird genau aufgeführt, welche Provider von welchen APIs unterstützt werden. Für den private Cloud Bereich bietet sich hier eher Libvirt an, da neben XEN, KVM, Qemu auch VMWare ESX unterstützt wird.

3.4.2 Storage (Object/Blob)

libcloud

- PCextreme AuroraObjects
- Microsoft Azure (blobs)
- CloudFiles
- Google Storage
- KTUCloud Storage
- Numbus.io
- Ninefold
- OpenStack Swift
- Amazon

jclouds (BlobStore)

- AWS
- Azure
- HP Helion
- Rackspace

fog

- S3
- Google Storage
- CloudFiles

pkgcloud

- Amazon
- HP
- Azure
- OpenStack
- Google
- Rackspace

libvirt

- GlusterFS
- FiberChannel
- Sheepdog
- NFS
- SCSI
- lvm
- iSCSI
- filesystems

3.4.3 Network

Libvirt allein bietet hier mit Abstand die beste Unterstützung von Network Konfigurationen (VLANs, Bridges, etc.).

3.4.4 Other

Database

pkgcloud

- IrisCouch
- MongoHQ
- MongoLab
- Rackspace
- RedisToGo

DNS

libcloud

- AuroraDNS
- DigitalOcean
- Gandi
- Google
- Host Virtual
- Linode
- Rackspace
- AWS Route53
- Softlayer
- Zerigo

fog

- AWS Route53
- Blue Box
- DNSimple
- Linode
- Rackspace
- Rage4
- Slicehost
- Zerigo

pkgcloud

- Rackspace

Load Balancer

libcloud

- Brightbox
- CloudStack
- DimensionData
- Amazon
- Google
- GoGrid
- Ninefold
- Rackspace
- Softlayer

jclouds

- AWS Elastic LoadBalancer
- Rackspace

pkgcloud

- Rackspace

Orchestration
pkgcloud (beta)

- OpenStack
- Rackspace

CDN
fog

- CloudFront

3.5 Fazit

Im Gesamtbild schneidet libcloud sehr gut ab. Es bietet deutlich am meisten Compute und Storage Provider. Die Dokumentation ist sehr ausführlich, mit konkreten Ratschlägen zur Implementation (z.B. Thread Safe). Zusätzlich bietet libcloud Module für SSL und Pricing. Jclouds ist eine Library für Java, was für uns am besten ist, da wir am meisten Erfahrung mit Java haben. Es gibt jedoch nicht viele Compute Provider, dafür unterstützt jclouds als einziger Docker. Der einzige Vorteil von fog ist die Möglichkeit CDNs als Service anzubieten. Pkgcloud unterstützt eine breite Auswahl von Services (z.B. Database, Load Balancer, DNS). Elibcloud ist ein erlang Wrapper für libcloud und unterstützt somit das gleiche wie libcloud (solange die Version auf dem neusten stand ist). Erlang würde sich für eine parallele Umgebung eignen, es sind jedoch keinerlei Erlang Kenntnisse im Team vorhanden.
Libvirt ist allerdings die einzige API, die sich völlig auf Private Anbieter konzentriert (wird auch von OpenStack verwendet), wäre also die Beste Lösung, wenn es darum geht eine Private Cloud aufzubauen.
Wir entscheiden uns für libvirt. Die Gründe dafür sind im Abschnitt 3.5.7 aufgeführt

3.5.1 libcloud

- ⊕ Grösste Auswahl an Compute und Storage Provider.
- ⊕ Am besten dokumentiert. Für jede Methode existiert eine Tabelle, die zeigt welche Provider damit angesprochen werden können.
- ⊕ Ist zwar nicht Thread-Safe. Es werden jedoch konkrete Lösungsvorschläge gemacht.
- ⊕ SSL und Pricing Module vorhanden.
- ⊖ Team hat wenig Erfahrung mit komplexen/grossen Python Projekten.

3.5.2 jclouds

- ⊕ Unterstützt Docker.
- ⊕ Java Library. Das Team hat am meisten Erfahrung mit Java.
- ⊕ Code Examples für fast jeden Provider.
- ⊖ Kleine Auswahl an Compute Providern.

3.5.3 fog

- ⊕ Es ist möglich ein CDN als Service anzubieten.
- ⊖ Mässige Dokumentation. Es existieren zwar Examples, die sind aber nicht besonders aussagekräftig.

- ⊖ Kleine Auswahl an Compute Providern.

3.5.4 pkgcloud

- ⊕ Grösste Auswahl an Services.
- ⊕ Database as a Service
- ⊕ Orchestration
- ⊕ Explizite Unterstützung von Network.
- ⊖ Mässige Dokumentation. Es existieren zwar Examples, die sind aber nicht besonders aussagekräftig.
- ⊖ Kleine Auswahl an Compute Providern.

3.5.5 libvirt

- ⊕ Grösste Auswahl an Private Cloud Anbietern.
- ⊕ Grosse Storage Unterstützung.
- ⊕ Explizite Unterstützung von Network.
- ⊕ Bietet Java Library (language Bindings)
- ⊕ Thread-Safe
- ⊖ Ungenügende Dokumentation. Es existieren zwar Examples und sonst existiert nur eine Javadoc zur Library.

3.5.6 Compute Vergleich

	Libcloud	jClouds	fog	pkgcloud	libvirt
AWS					
Abiquo					
PcextremeAuroraCompute					
Azure					
Blue Blocks					
Brightbox					
CloudFrames					
CloudSigma					
CloudStack					
Cloudwatt					
DigitalOcean					

Docker					
DimensionData					
Dreamhost					
Enomaly Elastic Computing Platform					
ElasticHosts					
Eucalyptus					
Exoscale					
Gandi					
Go2Cloud					
Google Compute Engine					
GoGrid					
HostVirtual					
HP Public Cloud (helion)					
IBM SmartCloud Enterprise					
Ikoula					
Joyent					
Kili Public Cloud					
KTU Public Cloud					
Libvirt					
Linode					
NephoScale					
Nimbus					
Ninefold					
OpenHosting					
OnApp					
OpenNebula					
OpenStack					
Opsource					
Outscale					
Packet					
ProfitBricks					
Rackspace Cloud					
RimuHosting					
RunAbove					
ServerLove					
skalicloud					
SoftLayer					
vCloud					
VCL					
Voxel VoxCLOUD					
vps.net					

Vmware vSphere					
Vultr					
XEN					
KVM					
OpenVZ					
IBM PowerVM					
Hyper-V					
Virtuozzo					
Bhyve					
VMware ESX					
VirtualBox					

3.5.7 API Vergleichsmatrix

	Entwicklungsstand	Provider	Doku	Serviceauswahl	Sprache	Zusatz Features
libcloud	**v0.18.0:** Die API ist stabil und wird von namhaften Firmen verwendet.	*ca.: 35 Unterstützt die wichtigsten Provider	gut	Wenige Services, dafür umfangreiche Funktionen	Python	SSL, Pricing, Übertragen von Files, Ausführen von Scripts, Unit Testability, Mocks
jclouds	**v1.9.1:** Die API ist stabil und wird von namhaften Firmen verwendet.	*ca.: 15 Unterstützt die wichtigsten Provider und als einziger Docker	genügend	Wenige Services, dafür umfangreiche Funktionen	Java	Thread-Safe, Unit Testability, Dateiübertragung, Scriptausführung
fog	**v1.34.0:** Die API ist stabil, wird jedoch von keinen namhaften Firmen verwendet	sehr kleine Auswahl	ungenügend	Wenige Services. Unterstützt als einziger CDNs.	Ruby	
pkgcloud	**V1.2.0:** Wird von keinen namhaften Firmen verwendet. Teilweise Betastand.	sehr kleine Auswahl, die wichtigsten sind jedoch dabei.	ungenügend	Breite Auswahl	JavaScript (Node.js)	Unit Testability
Libvirt	**1.2.20:** Bietet eine stabile API in C (bietet auch Language Bindings für Java und Python.)	Grosse Auswahl für Private Anbieter (Xen, KVM etc.)	genügend	Grosser Umfang schön unterteilt auf Compute (Domain), Storage, Network	C (Java/Python)	Private Anbieter Unterstützung, Thread-Safe, bietet Mock zum testen
	gut genügend ungenügend	*Verschiedene Standorte vom gleichen Provider zusammengefasst				

3.6 Bitnami Analyse

3.6.1 Einführung

Bitnami bietet ein Dashboard für einige Cloud Anbieter (VMware, AWS, Google Cloud, Azure, Digitalocean), um ganz einfach vorgegebene viel verwendete WebApps,Datenbanken oder Technologie Stacks schnell in der Cloud zu starten. Dabei wird beim jeweiligen Anbieter eine Compute Instanz erstellt und das Image zum Anbieter kopiert, von welchem die Instanz dann startet. Diese Analyse soll dabei helfen eine gute Lösung für unser Dashboard bzw. RESTful API/Generic API zu konzipieren und auf bereits bewährtes zurückgreifen zu können.

3.6.2 Cloud Anbieter

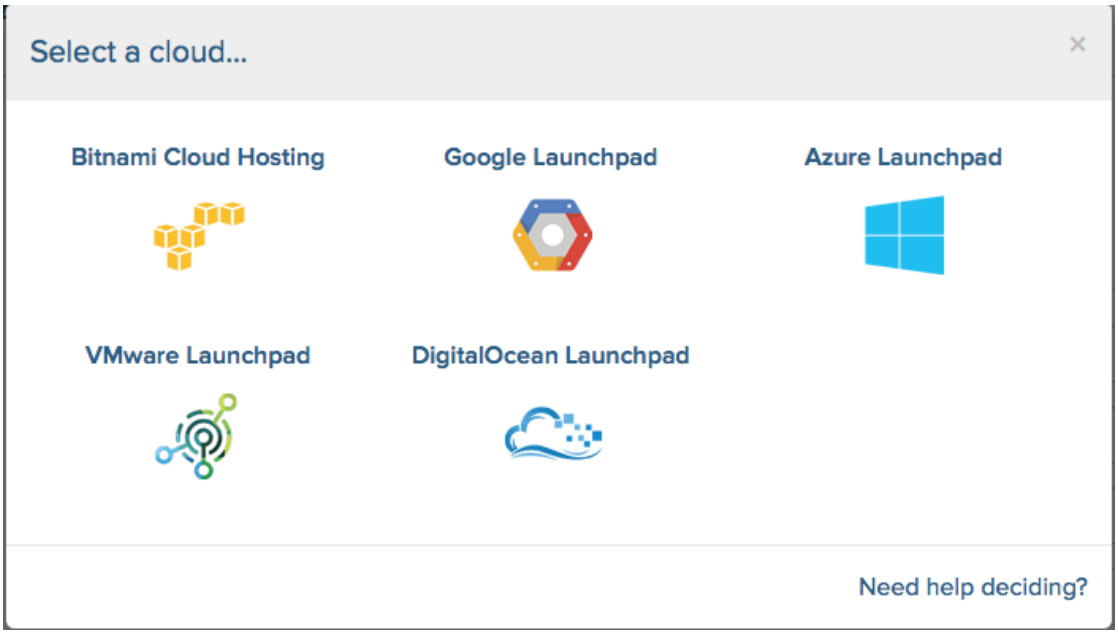


Abbildung 3.1: Cloud Anbieter

Bitnami Cloud Hosting[7]

Beim Bitnami Cloud Hosting steckt AWS dahinter, hier sind die meisten Einstellungen möglich im Vergleich zu den anderen Dashboards.

Übersicht

Kurze Übersicht über die möglichen Konfigurationen einer AWS Instanz.

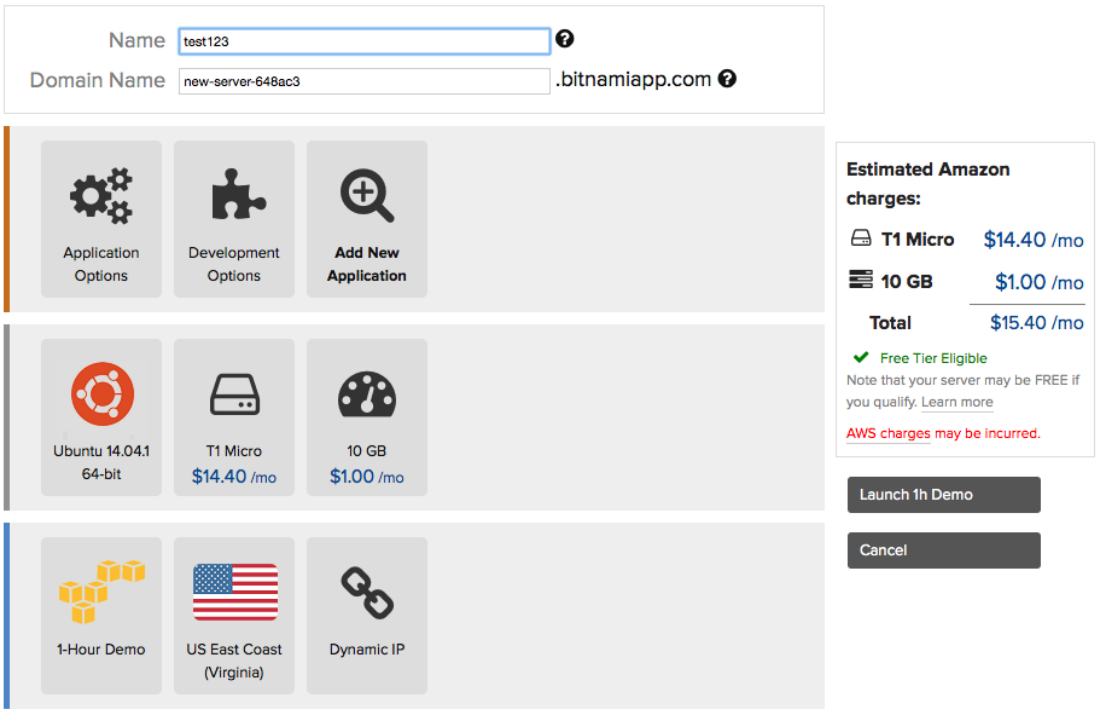
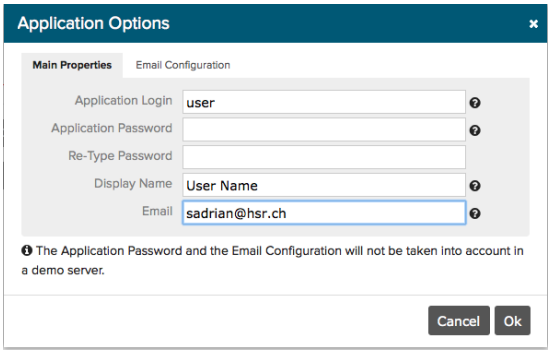


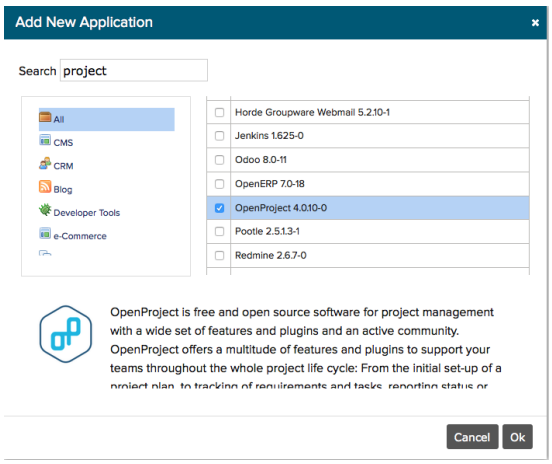
Abbildung 3.2: AWS Übersicht

Applikationseinstellungen



(a) AWS Application Options

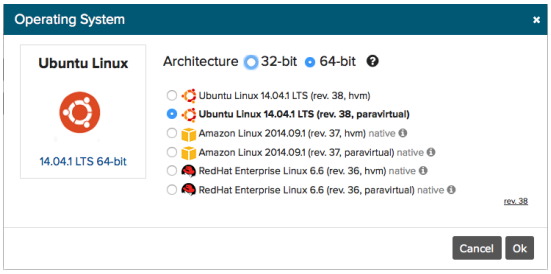
Applikationsauswahl



(b) AWS Apllikationsauswahl

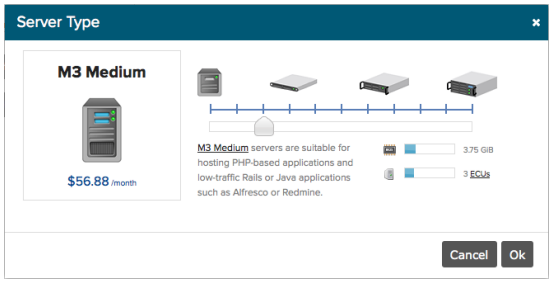
Abbildung 3.3: AWS Applikationseinstellungen

Betriebssystem



(a) AWS Betriebssystem

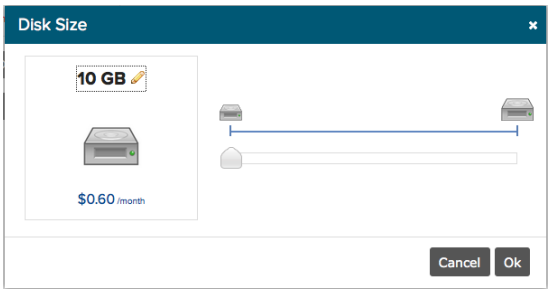
Servertyp



(b) AWS Servertyp

Abbildung 3.4: AWS Betriebssystem und Servertyp

Diskgrösse



(a) AWS Diskgrösse

Region,IP,Account

(b) AWS Region,IP und Account

Abbildung 3.5: AWS Diskgrösse + Region,IP,Account

Management

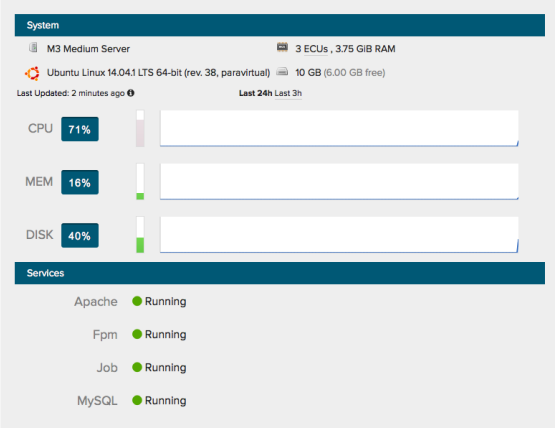
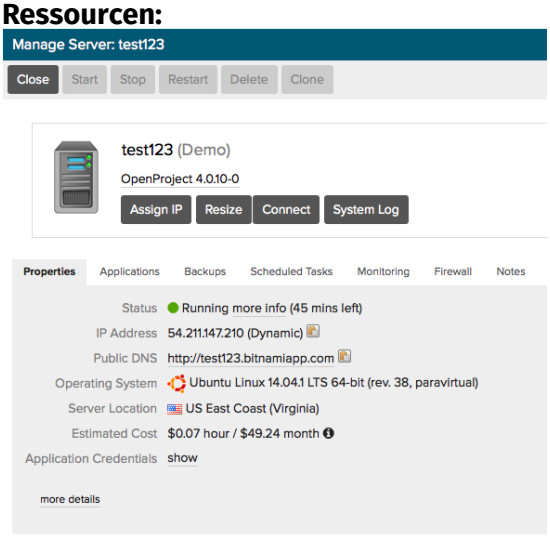
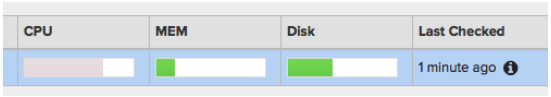


Abbildung 3.6: AWS Ressourcen Management

Ressourcen:



(a) AWS Server Management



(b) AWS Server Ressourcen

Abbildung 3.7: AWS Ressourcen

Digitalocean Launchpad[8]

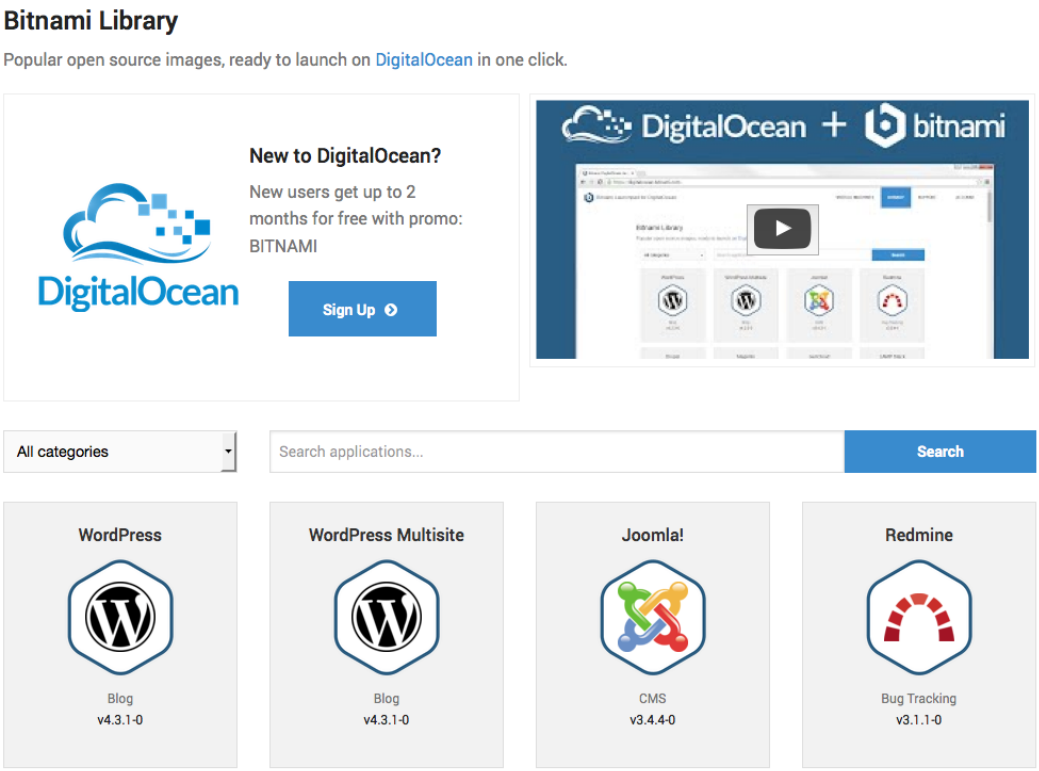


Abbildung 3.8: Digitalocean Launchpad

Authorize Application

Um Instanzen erstellen zu können muss das Bitnami Dashboard Zugriff auf den User-Account beim Cloud Anbieter haben.

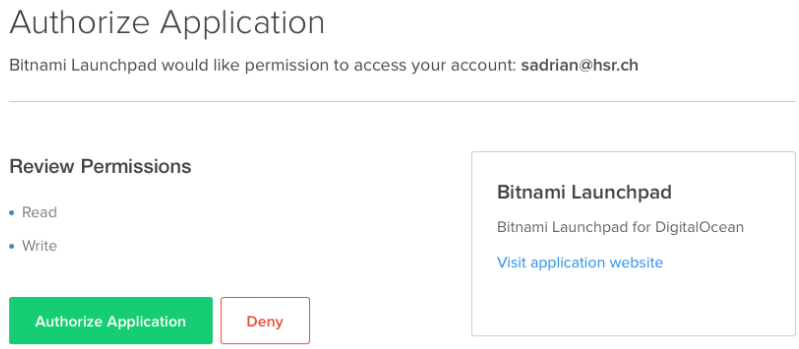
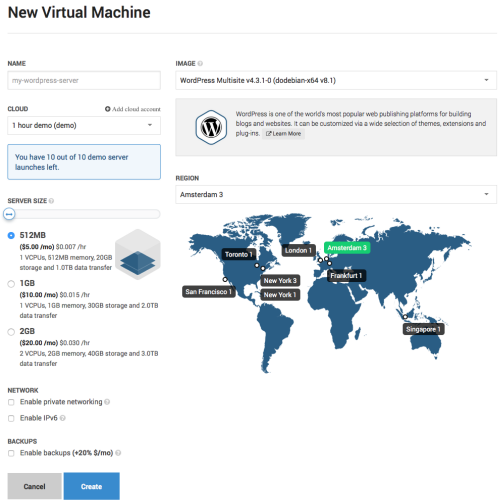


Abbildung 3.9: Digitalocean Applikation autorisieren

Instanzen

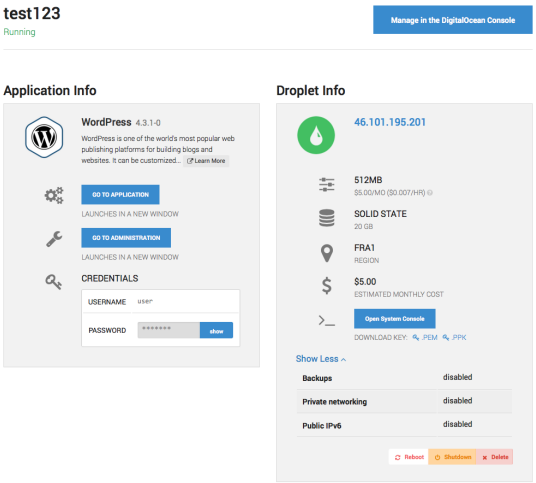
Sobald eine Applikation ausgewählt wurde kann eine Instanz mit dem App gestartet werden, dabei kann noch die Instanzgrösse und der Standort gewählt werden.



(a) Digitalocean Instanz

Instanzinfos

Sobald die Instanz erstellt wurde lässt sich deren Status überprüfen + App spezifische Links werden gesetzt und Private Key/Public Key lassen sich herunterladen.



(b) Digitalocean Instanzinfos

Abbildung 3.10: Digitalocean Instanz

Cloud Accounts

Es besteht die Möglichkeit mehrere Cloud Accounts in Bitnami zu hinterlegen.

Abbildung 3.11: Digitalocean Cloud Accounts

Und unter jedem Account können eigene VMs erstellt werden:

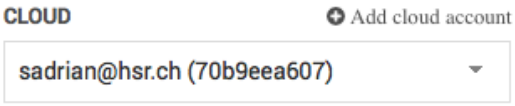


Abbildung 3.12: Digitalocean Account spezifischische VMs

Danach taucht die Instanz in der Übersicht auf:

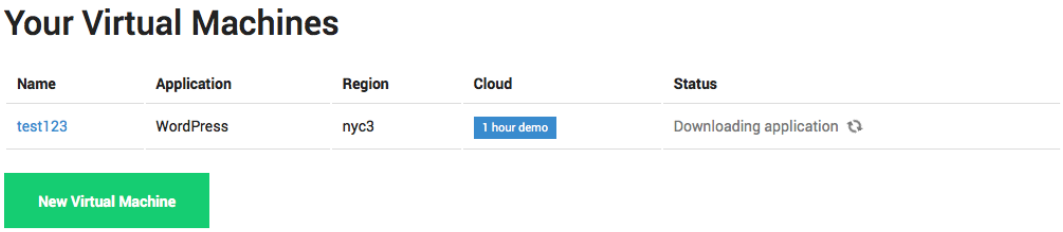


Abbildung 3.13: Digitalocean Instanzen

Azure Launchpad[9]

Beim Azure Launchpad wird wieder gleich vorgegangen, wie bei Digitalocean. Nur das sich die Namen ändern (Bspw.: Bei Digitalocean Droplet, jetzt Server).
Ebenfalls ändern sich die Instanzgrössen, welche bei Azure anders als bei Digitalocean sind + wird bei Azure mit Subscriptions und nicht anhand von Accounts unterschieden.

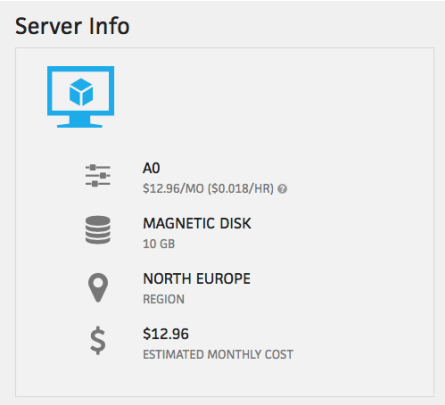
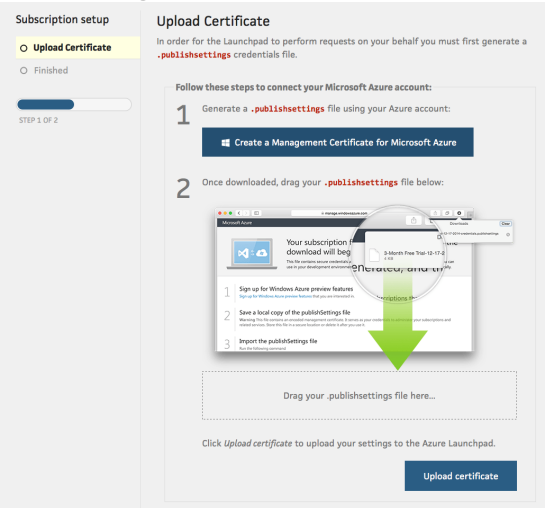


Abbildung 3.14: Azure Serverinfo

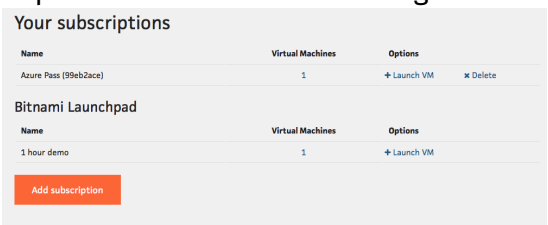
Authorization:

Bei Azure wird das Dashboard über ein Management Zertifikat autorisiert:



(a) Azure Management Zertifikat

Danach werden die vorhandenen Subscriptions vom Azure Account eingebunden:



(b) Azure Subscription

Abbildung 3.15: Azure Authorization

Sobald dann ein neuer Server erstellt wurde kann dieser auch wieder gelöscht werden und all dessen Infos eingesehen werden.

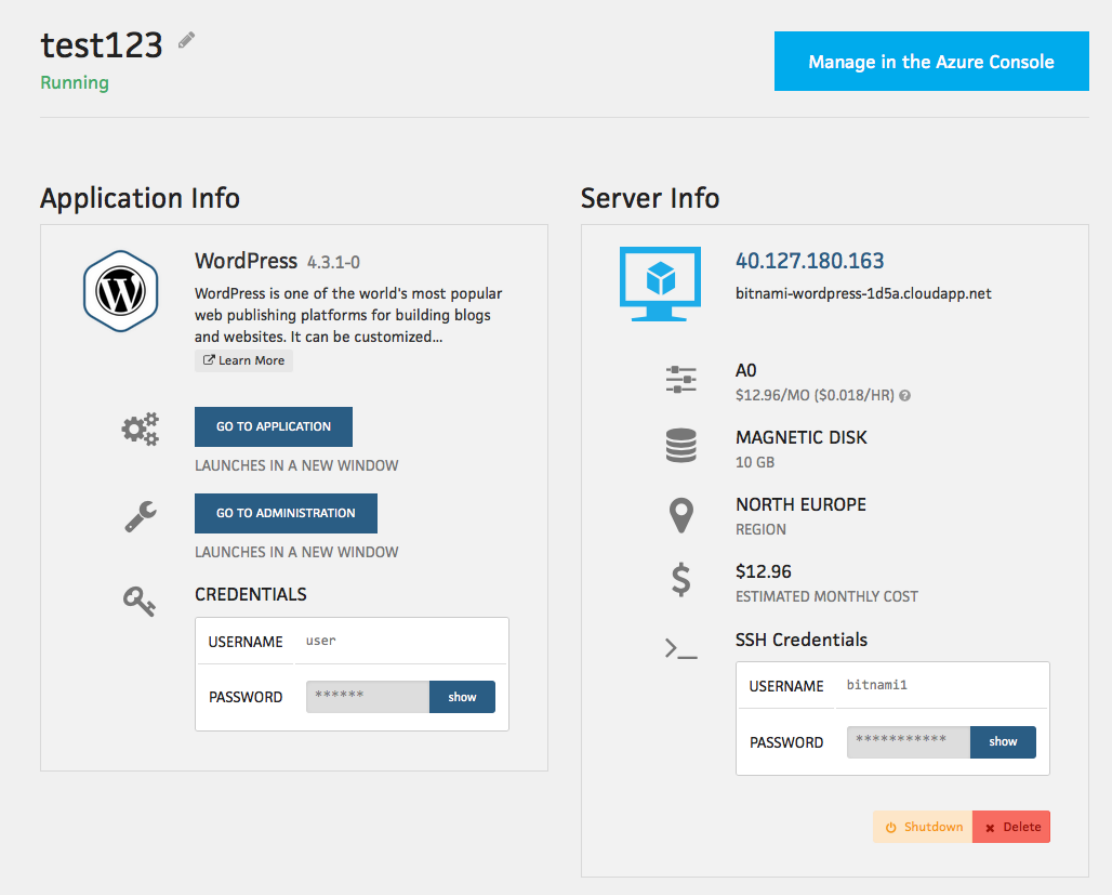


Abbildung 3.16: Azure Instanzinfos

Google Launchpad[10]

Beim Google Launchpad wieder dasselbe, wie bei Azure oder Digitalocean. Hier wird einfach mit Projekten unterschieden (schliesslich können jedem Projekt mehrere Compute Instanzen oder andere Services angefügt werden).

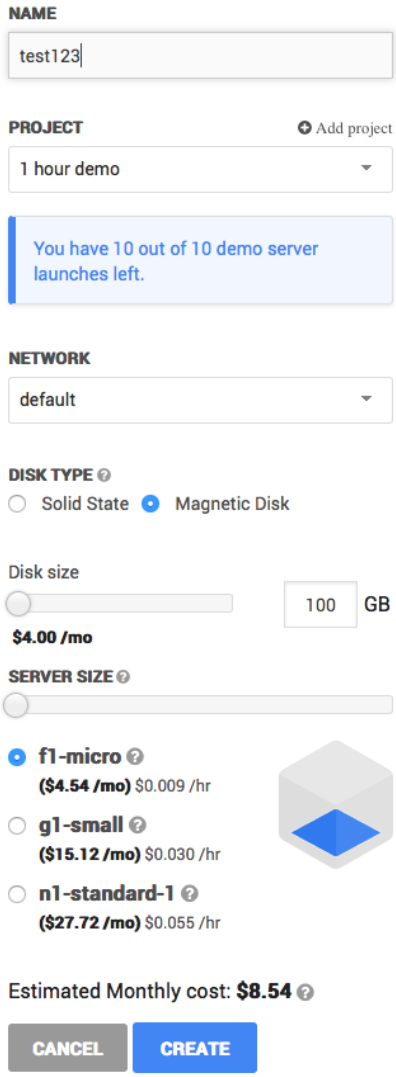
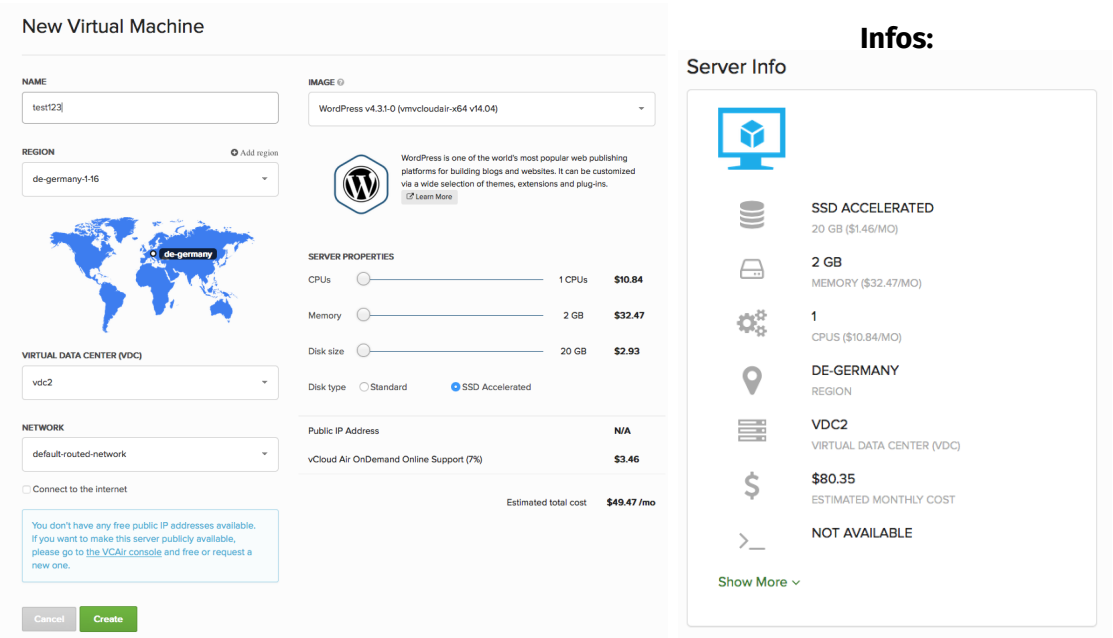


Abbildung 3.17: Google Instanzerstellung

VMware Launchpad[11]

Das VMware Launchpad kann für die VMware vCloud Air gebraucht werden und wieder die gleiche vorgehensweise, wie bei den anderen Anbietern.



(a) VMware Instanzerstellung (b) VMware Infos

Abbildung 3.18: VMware Launchpad

3.6.3 Security

Da durch das zentrale zusammenfassen von mehreren Accounts auch immer Sicherheitsrisiken zu beachten sind, wird bei Bitnami zusätzlich zum normalen Benutzerpasswort auch noch ein Vaultpasswort gesetzt, welches wohl die Logindaten symmetrisch verschlüsselt und in einem “Vault” ablegt.

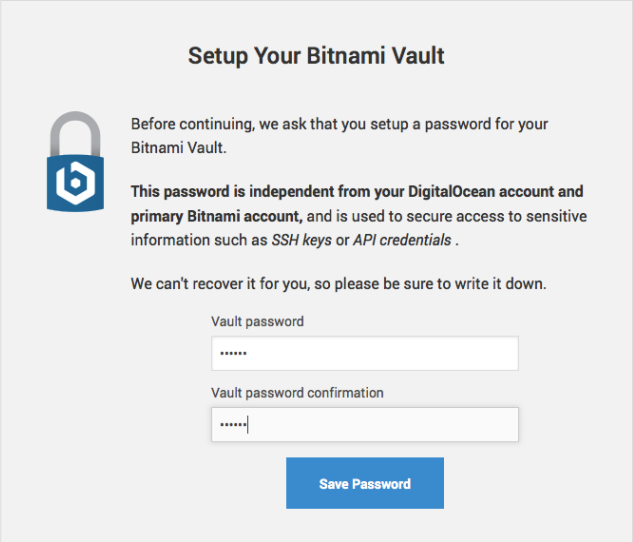


Abbildung 3.19: Bitnami Security

3.6.4 Applikationen

Die Applikationen können sich von Anbieter zu Anbieter unterscheiden, jedoch gibt es für sehr bekannte Applikationen (Bspw.: Wordpress) bei jedem ein Image. Es besteht bereits eine sehr grosse Auswahl für sehr viel verschiedene Apps und es werden immer mehr, wodurch es immer einfacher wird schnell eine Applikation in der Public Cloud aufzusetzen.

Abbildung 3.20: Applikationen

3.7 Fazit

Bitnami bietet einiges was Computing angeht und ist der einzig grössere Dashboard Anbieter, der mehrere verschiedene Public Cloud Anbieter unterstützt. Allerdings fehlen verschiedene PaaS Angebote (Bspw.: Cloud SQL bei Google etc.), Bitnami ist daher nur für eigene VM Instanzen zu gebrauchen und nicht mit einer generellen Service Unterstützung konzipiert worden. Das Dashboard bei Bitnami bietet jedenfalls einiges und gibt einem einen guten Überblick über seine eigenen abonnierten Services (VM Instanzen).

3.8 Dashboard Analyse

Das Dashboard soll dem Nutzer schnell und einfach die Übersicht über seine eigenen abonnierten Services bieten (Compute,Storage,Network). Dabei soll auf eine Anzahl von Cloud Anbietern angeboten werden, sowohl Public Cloud(z.B.: AWS, Google Cloud, Azure, Digitalocean), wie auch Private Cloud (z.B.: XEN, KVM, VMWare ESX). Die nachfolgenden Mockups sollen bereits einen ersten Eindruck der möglichen Funktionalitäten eines Dashboard geben, welches auch Einfluss auf die Konzipierung der RESTful und GenericAPI hat.
Beim Login wird zwischen Nutzern und Administrator unterschieden.

3.8.1 Homescreen Nutzer

Der Nutzer kriegt eine Übersicht über die vorhandenen Cloud Anbieter und kann gemäss Wunsch den richtigen wählen.

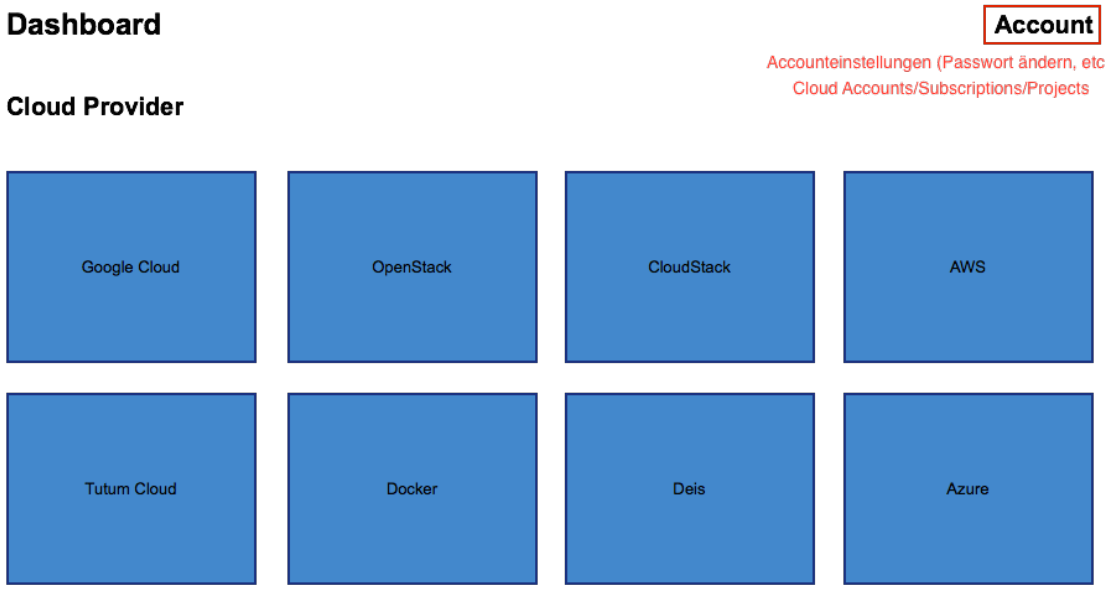


Abbildung 3.21: Homescreen Nutzer

3.8.2 Homescreen Admin

Der Administrator kriegt lediglich eine Übersicht über die vorhanden Nutzer und kann diese löschen oder ändern.

Abbildung 3.22: Homescreen Admin

3.8.3 Offerings

Sobald man einen der Cloud Anbieter gewählt hat (auf dem Homescreen) öffnet sich das jeweilige Dashboard des Anbieters mit dessen spezifischen Services.

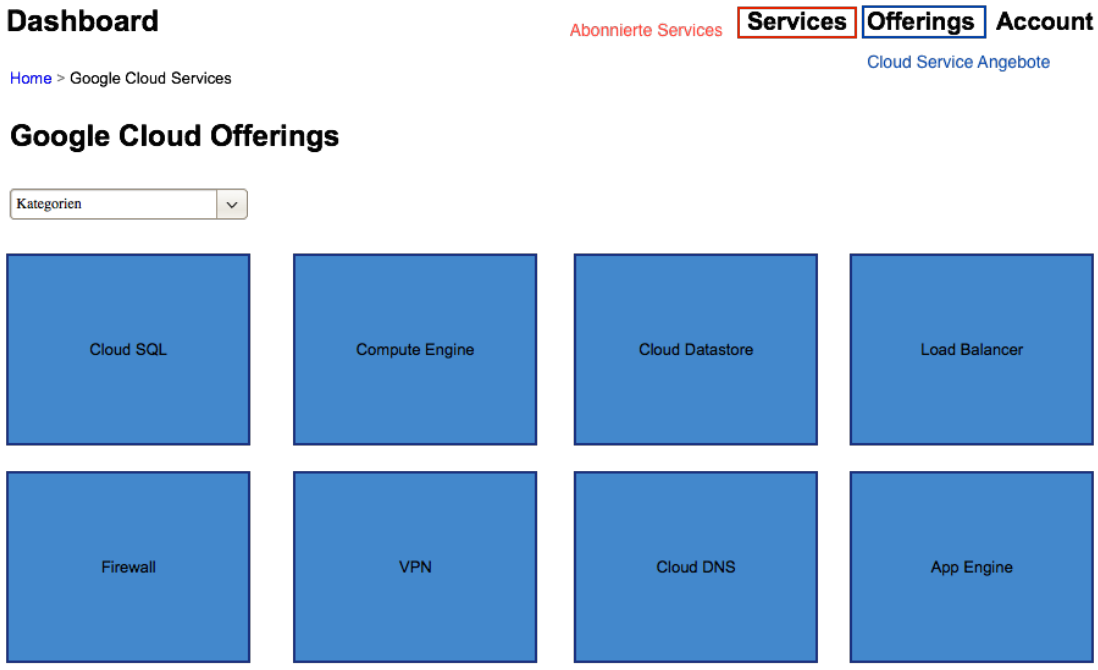


Abbildung 3.23: Homescreen Google

Compute:

Hier werden nur Compute Offerings angezeigt z.B.: App Engine, Compute Engine, Container Engine, EC2 etc., diese können nach Anbieter variieren

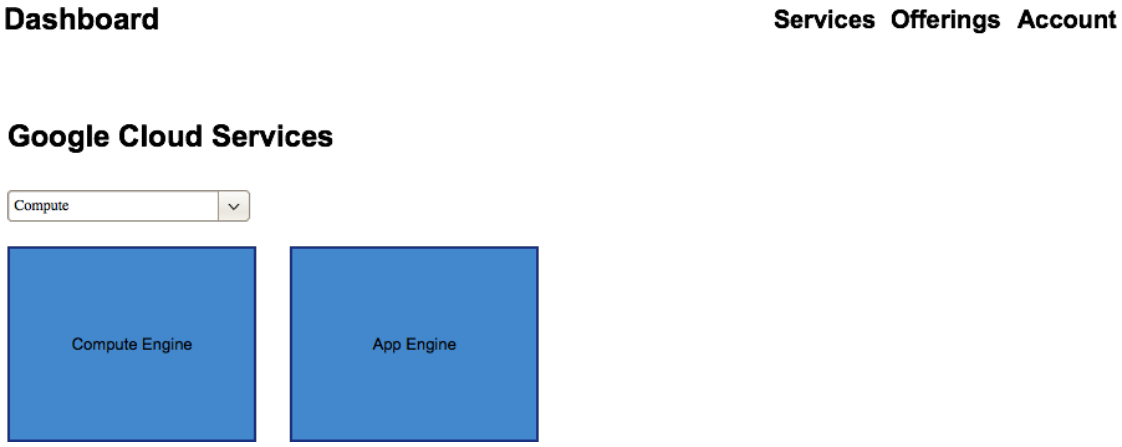


Abbildung 3.24: Homescreen Kategorie Compute

Storage:

Nur Storage spezifische Offerings anzeigen z.B.: Cloud Datastore, Cloud SQL, Cloud BigTable), die sich je nach Anbieter ändern.

Abbildung 3.25: Homescreen Kategorie Storage

Network:

Network spezifische Offerings anbieten (Firewall, VPN, Netzwerke, Cloud DNS etc.) und hier ist die Auswahl ebenfalls Anbieter spezifisch.

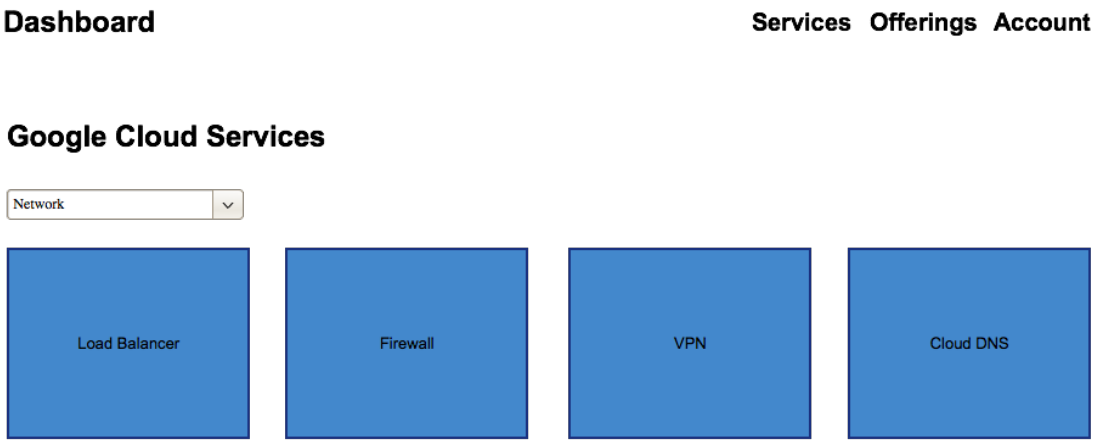


Abbildung 3.26: Homescreen Kategorie Network

3.8.4 Services

In Services kriegt man einen Überblick über all seine abonnierten Services des jeweiligen Anbieters und kann den zu bearbeitenden wählen.

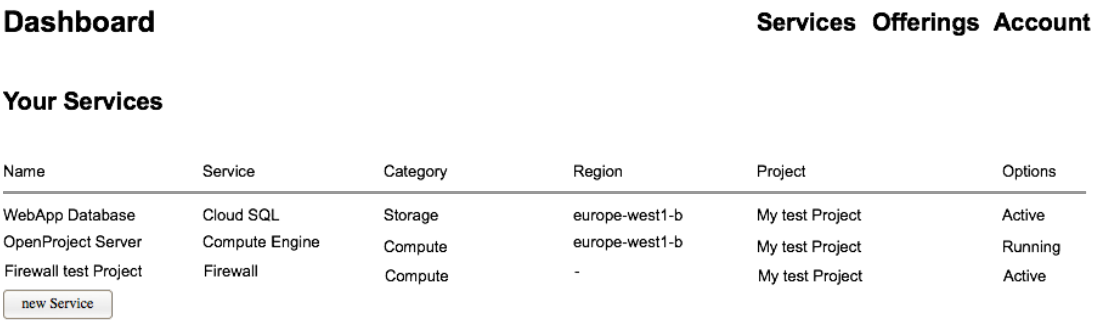


Abbildung 3.27: Services Übersicht

Compute

Wenn man einen Compute Service wählt kriegt man eine Übersicht des Services, welcher Storage, Leistung + Region (Storage und Compute werden jedoch meistens vorgegeben durch die Instanzgrössen) + werden die Kosten pro Monat angezeigt. Im Management kann dann die Instanz direkt neugestartet/heruntergefahren oder gelöscht werden + sollen noch Links (Ip etc.) zur Verfügung stehen um sich auf die Instanz verbinden zu können.

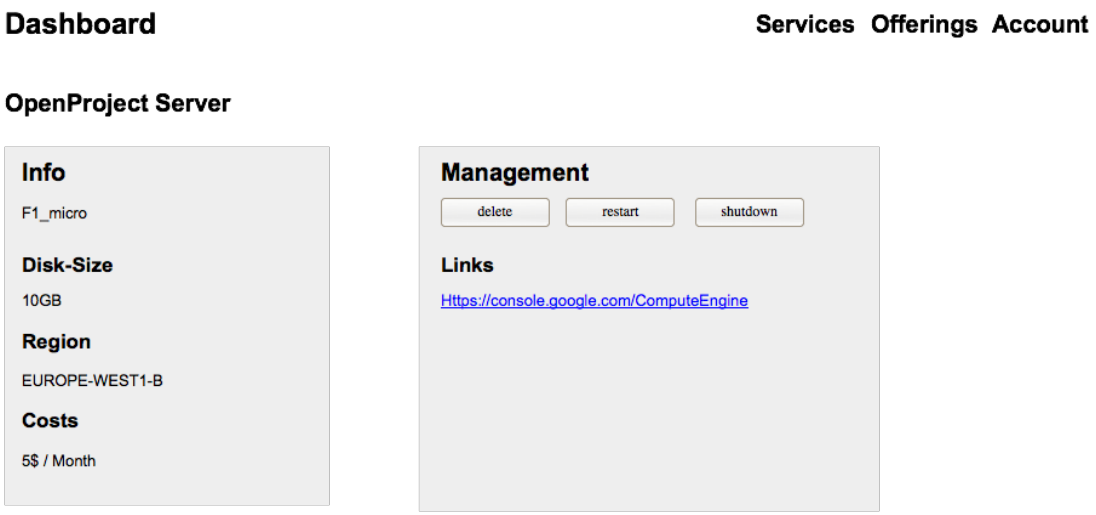


Abbildung 3.28: Services Infos Compute

Storage

Bei Storage spielt es wieder eine Rolle was für eine Art Storage es ist in diesem Beispiel ist es eine Cloud SQL Datenbank. Dabei wird vielmals anhand der Anzahl Rows oder Grösse der Datenbank abgerechnet + sollen hier auch wieder Links verfügbar sein, um auf ein Datenbankdashboard zu gelangen + die Möglichkeit geben den Service löschen zu können.

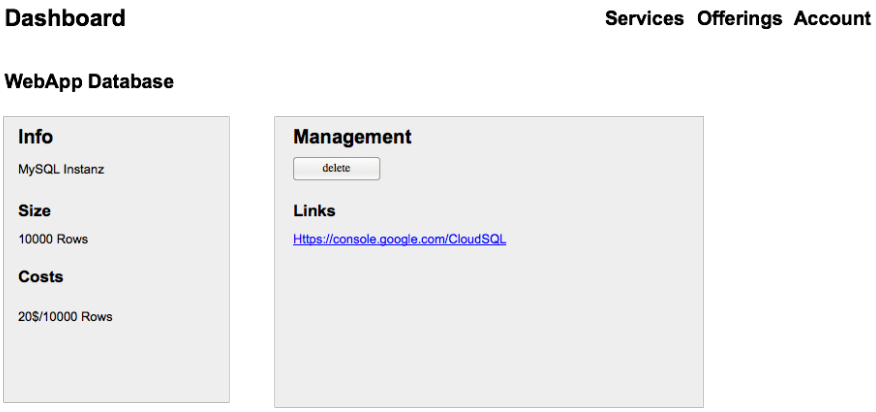


Abbildung 3.29: Services Infos Storage

Network

Bei Network kann man noch einige Dinge konfigurieren von Cloud DNS bis zu Netzwerken (Subnetze etc.) auch Firewall Regeln, da bei den Cloud Anbietern vielmals nur SSH und HTTP/S zugelassener traffic ist muss man schliesslich auch die Möglichkeit haben Firewall Regeln festlegen zu können. Das könnte Schlussendlich in etwa so aussehen:

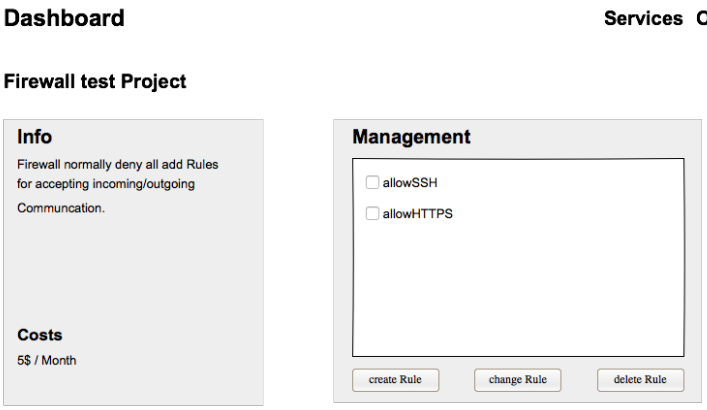


Abbildung 3.30: Services Infos Network

3.8.5 Accounts/Subscriptions/Projects/...

Für jeden Anbieter soll dem User eine Übersicht über die Account/Subscriptions-/Projects gegeben werden, dadurch vereinfacht sich die Handhabung von mehreren Accounts und alle sind in einem Dashboard zusammengefügt (-> Security beachten). Dadurch hat man immer den Überblick für welches Projekt/Account man wie viele Services abonniert hat.

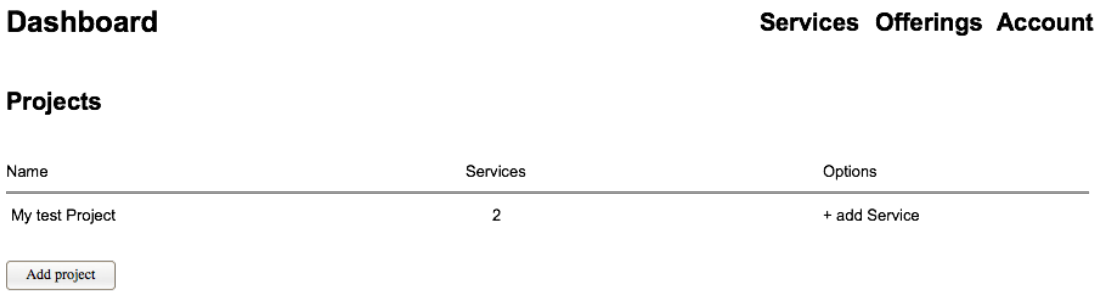


Abbildung 3.31: Accounts Übersicht

3.9 Security

Wie bei Bitnami wäre es wohl sicherer die Zugriffsdaten für die Cloud Anbieter (Benutzername/Passwort und SSH Keys) abzusichern (bei Bitnami wird dies über ein Vault sichergestellt), ansonsten könnte ein Angreifer ganze Instanzen bei verschiedenen Anbietern löschen oder sonstige Bösartige Absichten ausüben.
Dieser Vault soll auch durch ein zusätzliches Passwort geschützt sein und wird symmetrisch verschlüsselt (Mail Anbieter: Proton Mail[12] macht dies ebenfalls so) -> jedoch fragt Bitnami bei jedem login wieder nach dem Passwort und vergisst dann wieder alle Instanzen (ein Abgleich mit dem Anbieter wäre hier sicher sinnvoll).

3.10 Fazit

Das Dashboard scheint soweit umsetzbar zu sein, ein Knackpunkt wird einfach noch die Absicherung der Zugriffe auf alle Cloud Anbieter und deren Services (Wie in 3.9 beschrieben)

3.11 Use Case Skizzen

3.11.1 Akteure

Akteur	Ziel
Public User	Registrieren
Registered User	**Wenn User:** Login Logout Service anlegen (Create) Service Infos lesen (Read) Service ändern (Update) Service löschen (Delete) Benutzerinfos ändern **Wenn Admin:** Benutzer anlegen Benutzer löschen Benutzer ändern Benutzerinfos ändern
Cloud Anbieter	Service anlegen (Create) Service Infos lesen (Read) Service ändern (Update) Service löschen (Delete)

3.11.2 Use Case Diagramm

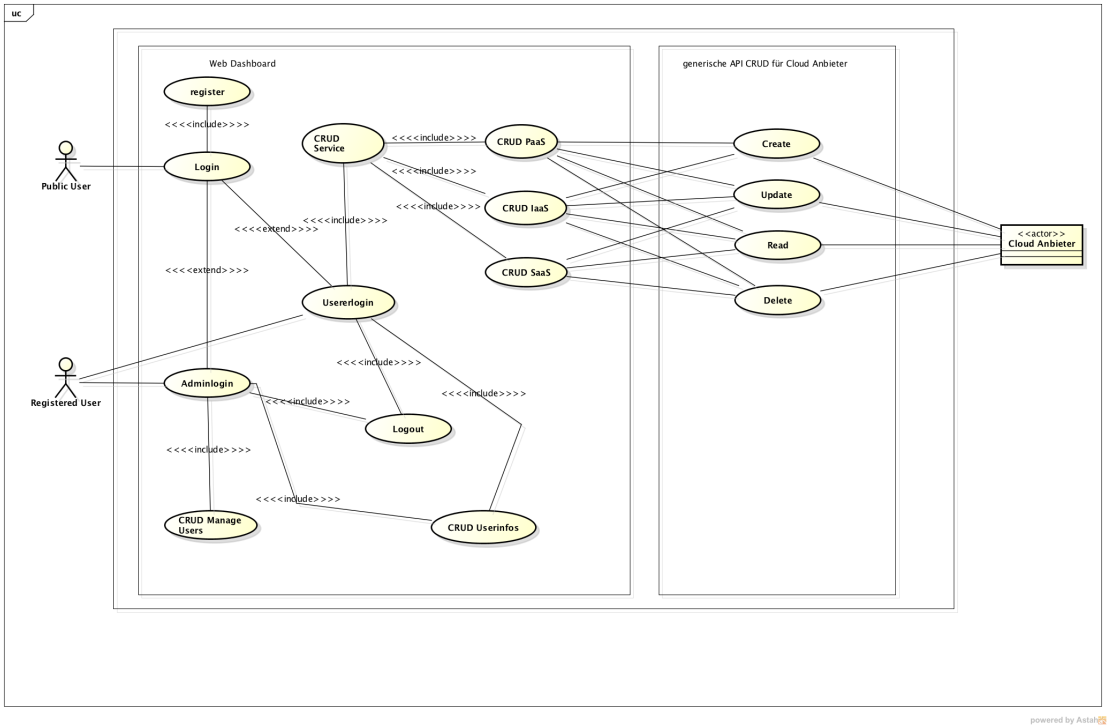


Abbildung 3.32: Use Case Diagramm Skizze

3.12 User Stories Skizzen

3.13 Rollen

3.13.1 Public User

Public User sind alle öffentlichen Besucher des Dashboards.

3.13.2 Registered User

Der registrierte Nutzer ist Anwender des Dashboards und verwendet dieses zur Aufgabenerleichterung. Bei dem Nutzer kann es sich um einen System Administrator, DevOps, Operator oder Software Entwickler handlen, da beim Dashboard für jeden was dabei ist.

3.13.3 Admin

Der Admin ist für die Instandhaltung des Dashboards zuständig und verwaltet die User.

3.14 Ziele

Im Umfang soll die Applikation in etwa folgendes bieten:

- Registrierung (Mail Adresse/Passwort)
- Login
- Administrationsoberfläche
- Benutzerinfos anpassen
- Auswahl aus mehreren Cloud Anbieter
- mehrere Cloud Accounts hinzufügen
- Abonnieren von Services (Compute/Storage/Network)
- Unterteilung der Services in Compute/Storage/Network
- Übersicht aller zur Verfügung stehenden Services
- Management der Services (erstellen/ändern/löschen)
- Links zu Serviceadministration beim Cloud Anbieter
- Übersicht über abonnierte Services
- Unterstützung Private Cloud (OpenStack,CloudStack, Docker, KVM, XEN)

- Anbieter spezifische Services anbieten
- generische API
- Anstehende Kosten anzeigen
- Einfaches hinzufügen eines Cloud Accounts (Wizard bieten)

3.15 Epic

- Service abonnieren (Compute/Storage/Network)

3.16 User Stories

3.16.1 Public User

- Als Public User möchte ich mich registrieren können
- Als Public User möchte ich mich auf Dashboard verbinden, um einloggen zu können

3.16.2 Registered User

- Als registered User möchte ich mich einloggen können
- Als registered User möchte ich eine Übersicht aller angebotenen Cloud Anbieter sehen
- Als registered User möchte ich Zugriff auf meine Accountinfos
- Als registered User möchte ich meine Accountinfos anpassen können
- Als registered User möchte ich ein Cloud Anbieter auswählen können, um auf die Übersicht der Offerings zu kommen
- Als registered User möchte ich eine Übersicht meiner abonnierten Services haben
- Als registered User möchte ich Services löschen können
- Als registered User möchte ich Compute Instanzen neustarten können
- Als registered User möchte ich Compute Instanzen herunterfahren können
- Als registered User möchte ich die kosten der Services angezeigt haben
- Als registered User möchte ich direkte Verlinkungen zu den Services haben

3.16.3 Admin

- Als Admin möchte ich Zugriff auf eine Administrationsoberfläche
- Als Admin möchte ich User erstellen können
- Als Admin möchte ich User löschen können
- Als Admin möchte ich User ändern können

3.17 Domainmodell Skizze

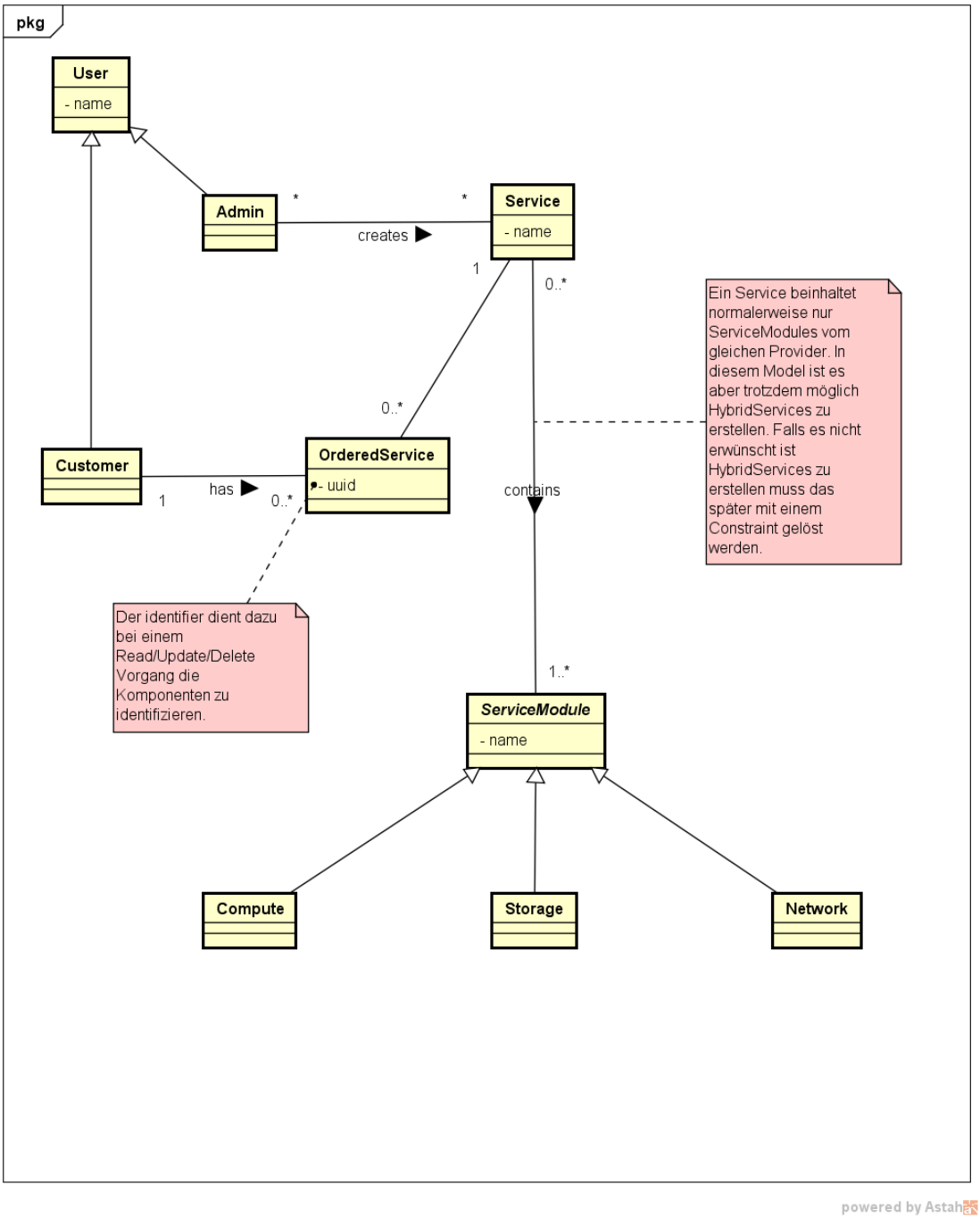


Abbildung 3.33: Domainmodell Skizze

4 Anforderungen

4.1 API

Die API definiert einen Workflow, der einen Service auf einer Cloud erstellt. Es ist offen, ob dieser Service über mehrere Cloud Anbieter hinaus geht. Der Service wird durch ein Konfigurationsfile definiert. Die Software auf den Instanzen wird durch Images installiert. Ein Service kann auch wieder gelöscht werden. Es ist nicht die Aufgabe der API existierende Services zu identifizieren. Die API muss Modular sein, das heisst es sollte möglich sein andere oder eigene Programme für die Cloud Kommunikation zu verwenden. Innerhalb der API werden Compute, Storage, Network usw. als Servicemodule bezeichnet. Diese Abstraktion ermöglicht das wiederverwenden und erweitern der API.

4.2 Customer-Dashboard

4.2.1 Homescreen

Im Homescreen werden alle zu Verfügung stehenden Services angezeigt. Hier werden die Services Offerings genannt um eine Unterscheidung zwischen Abonnierten Services (Services) und zur Verfügung stehenden Services (Offerings) machen zu können.

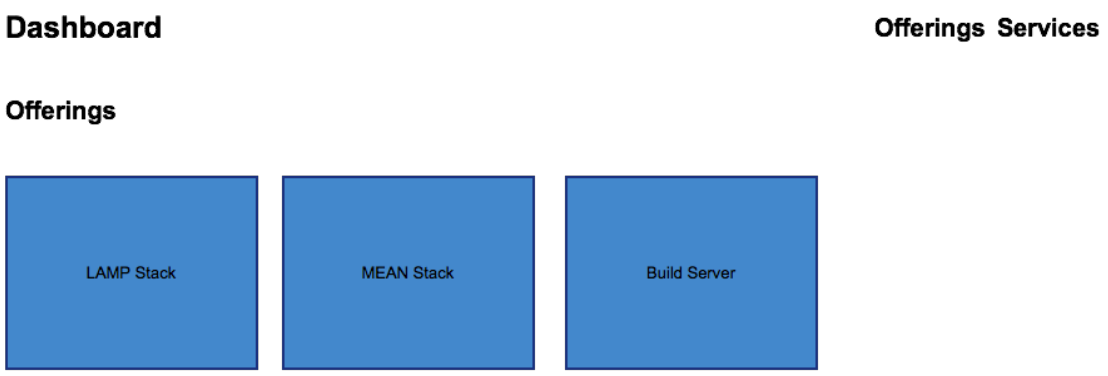


Abbildung 4.1: Homescreen Customer

4.2.2 Services Übersicht

In der Services Übersicht werden dem Customer alle abonnierten Services angezeigt und können hier auch gekündigt werden.

Abbildung 4.2: Services Übersicht

4.2.3 Service abonnieren

Sobald ein Service auf dem Homescreen ausgewählt wird und auf den “subscribe” Button geklickt wird, wird dieser abonniert und wird in der Services Übersicht angezeigt.

Abbildung 4.3: Services Settings

4.3 Admin-Dashboard

Zusätzlich zum Customer-Dashboard soll ein Admin-Dashboard zur Verfügung stellen in welchem der Admin Services und erstellen/bearbeiten/löschen kann.

Service

Ein Service hat einen bestimmten Namen und jedem Service sind eine gewisse Anzahl Servicemodule zugeteilt, um den Service abbilden zu können. Hier kann der Admin je nach Anforderung den Service verwalten.

Abbildung 4.4: Homescreen Admin

Servicemodul

Ein Servicemodul besitzt einen Namen, Provider und Kategorie. Hier kann der Admin je nach Anforderung das Servicemodul verwalten.

Dashboard **Services Servicemodules**

Servicemodules

Name	Provider	Type	Options
LAMP Instance	KVM	Compute	delete / change
MEAN Instance	Azure	Compute	delete / change
Build Server Instance	Xen	Compute	delete / change

create Servicmodule

Abbildung 4.5: Servicemodule Übersicht

4.4 Use Cases

4.4.1 Use Case Diagramm

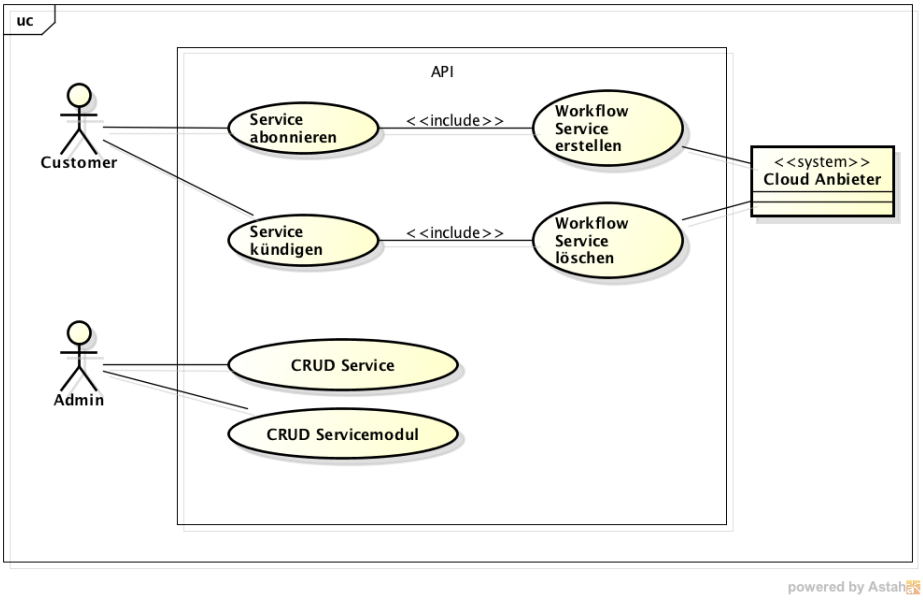


Abbildung 4.6: Use Case Diagramm

4.4.2 Aktoren & Stakeholders[13]

Customer

Als Customer möchte ich meine abonnierten Services verwalten.

Aktor	Typ	Ziele
Customer	Primary	• Service abonnieren • Service kündigen

Admin

Als Admin möchte ich Services und Servicemodule verwalten können.

Aktor	Typ	Ziele
Admin	Primary	• Service erstellen • Service anpassen • Service löschen • Servicemodul erstellen • Servicemodul anpassen • Servicemodul löschen

4.4.3 Beschreibungen fully dressed

UC01: Service abonnieren

Primäraktor	Customer
Steakholders und Interessen	Customer: Möchte einen Service abonnieren
Vorbedingungen	Das Customer-Dashboard wurde geöffnet.
Nachbedingungen	Die Service Infos wurden gespeichert und der Workflow wurde angestossen
Standartablauf	1. Wiederholen bis kein Service mehr abonniert werden soll (a) Der Customer wechselt in die **Offerings Übersicht** (b) Der Customer wählt einen der vorhanden **Services** aus (c) Der Customer drückt den Button **Order Service** (d) Der Customer wird in die **Services Übersicht** weitergeleitet 2. Der Customer schliesst das Customer-Dashboard
Alternativer Ablauf	1. (a) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab (b) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab ii. geht zurück in die **Offerings Übersicht** (c) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab ii. geht zurück in die **Offerings Übersicht** (d) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab ii. geht zurück in die **Offerings Übersicht**

Spezielle Anforderungen	siehe nichtfunktionale Anforderungen
Technologie- und Datenvarianten	Keine
Auftrittshäufigkeit	mehrmals pro Woche
Offene Fragen	Keine

UC02: Service kündigen

Primäraktor	Customer
Steakholders und Interessen	Customer: Möchte einen Service kündigen
Vorbedingungen	Das Customer-Dashboard wurde geöffnet.
Nachbedingungen	Die Service Infos wurden gelöscht und der Workflow wurde angestossen
Standartablauf	1. Wiederholen bis kein Service mehr gekündigt werden soll (a) Der Customer wechselt in die **Services Übersicht** (b) Der Customer wählt einen der vorhanden **Services** aus (c) Der Customer drückt auf den link **terminate** (d) Der Customer wird in die **Services Übersicht** weitergeleitet 2. Der Customer schliesst das Customer-Dashboard
Alternativer Ablauf	1. (a) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab (b) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab ii. wählt einen anderen Service (c) Der Customer entscheidet sich um i. Schliesst das Fenster/Tab ii. wählt einen anderen Service

Spezielle Anforderungen	siehe nichtfunktionale Anforderungen
Technologie- und Datenvarianten	Keine
Auftrittshäufigkeit	mehrmals pro Woche
Offene Fragen	Keine

UC04: Service verwalten

Primäraktor	Admin
Steakholders und Interessen	Admin: Möchte einen Service verwalten
Vorbedingungen	Das Admin-Dashboard wurde geöffnet
Nachbedingungen	Das Admin-Dashboard wurde geschlossen und Änderungen wurden gespeichert
Standartablauf	1. Der Admin öffnet das Admin-Dashboard 2. Wiederholen bis kein neuer Service hinzugefügt werden muss (a) Der Admin wechselt in die **Services Übersicht** (b) Der Admin drückt auf den button **create new Service** (c) Der Admin füllt die benötigten Daten ein **(Name, welche Servicemodule)** (d) der Admin bestätigt mit Klick auf Button **Save**
Alternativer Ablauf	2. (a) Wiederholen bis kein Service mehr geändert werden muss i. Service auswählen und auf Link **edit** klicken ii. Daten ändern iii. Durch Klick auf Button **Save** bestätigen (b) Wiederholen bis kein Service mehr gelöscht werden muss i. Service auswählen und Link **delete** auswählen
Spezielle Anforderungen	siehe nichtfunktionale Anforderungen
Technologie- und Datenvarianten	Keine
Auftrittshäufigkeit	mehrmals pro Monat

Offene Fragen	Keine

UC05: Servicemodul verwalten

Primäraktor	Admin
Steakholders und Interessen	Admin: Möchte ein Servicemodul verwalten
Vorbedingungen	Das Admin-Dashboard wurde geöffnet.
Nachbedingungen	Das Admin-Dashboard wurde geschlossen und Änderungen wurden gespeichert
Standartablauf	1. Der Admin öffnet das Admin-Dashboard 2. Wiederholen bis kein neues Servicemodul hinzugefügt werden muss (a) Der Admin wechselt in die **Servicemodules Übersicht** (b) Der Admin drückt auf den button **create new Servicemodule** (c) Der Admin füllt die benötigten Daten ein **(Name, Provider,Typ)** (d) der Admin bestätigt mit Klick auf Button **Save**
Alternativer Ablauf	2. (a) Wiederholen bis kein Servicemodul mehr geändert werden muss i. Servicemodul auswählen und auf Link **edit** klicken ii. Daten ändern iii. Durch Klick auf Button **Save** bestätigen (b) Wiederholen bis kein Servicemodul mehr gelöscht werden muss i. Service auswählen und Link **delete** auswählen
Spezielle Anforderungen	siehe nichtfunktionale Anforderungen

Technologie- und Datenvarianten	Keine
Auftrittshäufigkeit	mehrmals pro Monat
Offene Fragen	Keine

4.5 Epics

4.5.1 Customer

- Service abonnieren
- Service kündigen

4.5.2 Admin

- Service verwalten
- Servicemodul verwalten

4.6 User Stories

4.6.1 Rollen

Customer

Als Customer benutze ich das Dashboard, um für mich einen Service zu abonnieren oder zu kündigen.

Admin

Als Admin verwalte ich neue Services und Servicemodule und erweitere diese um neue Funktionen/Verbesserungen.

4.6.2 Customer

Customer abonniert Service

Priorität	Hoch
Story Points	4
Story	Als Customer möchte ich einen Service abonnieren können
Akzeptanzkriterien	
A1	Der Customer kann einen Service abonnieren
A2	Storage,Compute und Network wurden, wie im Service beschrieben erstellt

Customer kündigt Service

Priorität	Hoch
Story Points	4
Story	Als Customer möchte ich einen Service kündigen können
Akzeptanzkriterien	
A1	Der Customer kann einen Service kündigen
A2	Storage,Compute und Network werden, wie im Service beschrieben gelöscht

Customer will abonnierte Services sehen

Priorität	Hoch
Story Points	2
Story	Als Customer möchte ich sehen welche Services ich abonniert habe.
Akzeptanzkriterien	
A1	Der Customer kriegt eine Liste mit seinen abonnierten Services zurück

Customer will verfügbare Services angezeigt bekommen

Priorität	Hoch
Story Points	2
Story	Als Customer möchte ich sehen welche Services ich abonnieren kann.
Akzeptanzkriterien	
A1	Der Customer kriegt eine Liste mit seinen zur Verfügung stehenden Services zurück

Customer geht in die Offerings Übersicht

Priorität	Hoch
Story Points	6
Story	Als Customer möchte ich die Offerings in einer Übersicht ansehen können
Akzeptanzkriterien	
A1	Der Customer kann im Dashboard in die Offerings Übersicht wechseln.

Customer geht in die Service Übersicht

Priorität	Hoch
Story Points	6
Story	Als Customer möchte ich meine abonnierten Services in einer Übersicht angezeigt bekommen
Akzeptanzkriterien	
A1	Der Customer kann seine abonnierten Services in einer Übersicht anzeigen

Customer will Informationen über abonnierte Services sehen

Priorität	Hoch
Story Points	6
Story	Als Customer möchte ich Informationen über abonnierte Service einsehen können.
Akzeptanzkriterien	
A1	Der Customer kann abonnierten Service auswählen und kriegt Infos zu den Servicemodulen

4.6.3 Admin

Admin erstellt Service

Priorität	Hoch
Story Points	6
Story	Als Admin möchte ich Services erstellen können
Akzeptanzkriterien	
A1	Als Admin krieg ich die Übersicht der verfügbaren Services
A2	Dem Service können Servicemodule hinzugefügt werden
A3	Service kann erstellt werden
A4	Der Service ist erstellt

Admin ändert Service

Priorität	Hoch
Story Points	6
Story	Als Admin möchte ich Services ändern können
Akzeptanzkriterien	
A2	Als Admin krieg ich die Übersicht der verfügbaren Services
A3	Dem Service können Servicemodule hinzugefügt werden
A4	Service kann geändert werden
A5	Der Service ist geändert

Admin löscht Service

Priorität	Hoch
Story Points	6
Story	Als Admin möchte ich Services löschen können
Akzeptanzkriterien	
A1	Als Admin krieg ich die Übersicht der verfügbaren Services
A2	Der Service ist gelöscht

Admin greift auf Dashboard zu

Priorität	Hoch
Story Points	1
Story	Als Admin möchte ich auf das Admin-Dashboard zugreifen können
Akzeptanzkriterien	
A1	Der Admin kann den Url des Customer-Dashboard aufrufen und kriegt das Dashboard angezeigt

Admin geht in die Service Übersicht

Priorität	Hoch
Story Points	2
Story	Als Admin möchte ich einen Überblick über die vorhanden Services
Akzeptanzkriterien	
A1	Der Admin kann die Service Übersicht öffnen

Admin Konfigurationsdatei im Servicemodul hinterlegen

Priorität	Hoch
Story Points	2
Story	Als Admin möchte ich dem Servicemodul eine Konfigurationsdatei hinterlegen
Akzeptanzkriterien	
A1	Der Admin kann dem Servicemodul eine Konfigurationsdatei hinterlegen

Admin erstellt Servicemodul

Priorität	Hoch
Story Points	4
Story	Als Admin möchte ich Servicemodule erstellen können
Akzeptanzkriterien	
A1	Das Servicemodul wird erstellt und wird in der Servicemodule Übersicht angezeigt

Admin ändert Servicemodul

Priorität	Hoch
Story Points	4
Story	Als Admin möchte ich Servicemodule ändern können
Akzeptanzkriterien	
A1	Als Admin krieg ich die Übersicht der verfügbaren Servicemodule
A2	Als Admin kann ich das Servicemodule anpassen und speichern
A3	Änderungen werden gespeichert

Admin löscht Servicemodul

Priorität	Hoch
Story Points	4
Story	Als Admin möchte ich Servicemodule löschen können
Akzeptanzkriterien	
A1	Als Admin krieg ich die Übersicht der verfügbaren Servicemodule
A2	Servicemodule ist gelöscht

Admin geht in die Servicemodule Übersicht

Priorität	Hoch
Story Points	2
Story	Als Admin möchte ich einen Überblick über die vorhanden Servicemodule angezeigt bekommen
Akzeptanzkriterien	
A1	Der Admin kann die Servicemodule Übersicht öffnen
A2	Die Servicemodule Übersicht wird angezeigt

4.7 Nicht-funktionale Anforderungen

4.7.1 Menge

- Die Software unterstützt mindestens 1 Storage Anbieter
- Die Software unterstützt mindestens 1 Compute Anbieter
- Die Software unterstützt mindestens 1 Network Anbieter
- Es soll für Compute, Storage, Network mindestens je 1 Servicemodul erstellt werden

4.7.2 Schnittstellen

- Die Software wird über HTTP/HTTPS angesprochen
- Zur Interaktion im Admin-Dashboard/Customer-Dashboard werden die herkömmlichen Schnittstellen gebraucht (Maus,Tastatur,Bildschirm)

4.7.3 Qualitätsmerkmale

Funktionalität

siehe Abschnitt API und Dashboard

Zuverlässigkeit

- Der Workflow zum erstellen eines Services soll entweder durchgeführt und abgeschlossen oder falls Unterbruch/Fehler auftritt rückgängig gemacht werden
- Die Software soll verteilt betrieben werden und eine möglichst hohe Verfügbarkeit/Zuverlässigkeit bieten

Benutzerbarkeit

- Konfigurationen können über das vorgesehene Admin-Dashboard geändert werden
- Zum verwenden der Software soll noch ein einfaches User-Dashboard bestehen

Effizienz

- Die Software soll mehrere Aufträge von Customern gleichzeitig abarbeiten können

Änderbarkeit

Die Software soll modular aufgebaut werden, damit Erweiterungen in Zukunft möglich sind.

Übertragbarkeit

Das Projekt wird in Java geschrieben und ist somit also auf Java mindestens in der Version 1.8 angewiesen.

5 Design

5.1 RESTful API

Hier wird die RESTful API beschrieben, welche die Verwaltung von Services und Servicemodulen beschreibt.

5.2 Service Controller

Im Service Controller befinden sich alle Funktionen um Services verwalten und abonnieren zu können.

5.2.1 Services

URI Path /api/services
Methods GET

5.2.2 GET

Curl

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   services"
```

Request URL

```
http://example.com/api/services
```

Response Body

```
[
  {
    "id": 1,
```

```
    "serviceName": "string",
    "modules": [
      {
        "id": 1,
        "category": "Network",
        "size": null,
        "provider": "LibVirt",
        "config": "string",
        "name": "string"
      }
    ]
  },
    {
    "id": 2,
    "serviceName": "string",
    "modules": [
      {
        "id": 2,
        "category": "Compute",
        "size": "M",
        "provider": "LibVirt",
        "config": "string",
        "name": "string"
      }
    ]
  }
]
```


5.2.3 Neuer Service

URI Path /api/services/new
Methods POST

5.2.4 POST

Curl

```
curl -X POST --header "Content-Type: application/json" --
   header "Accept: */*" -d "{
  \"id\": 1,
  \"modules\": [
    {
```

```
      \"category\": \"Compute\",
      \"config\": \"string\",
      \"id\": 1,
      \"name\": \"string\",
      \"provider\": \"LibVirt\",
      \"size\": \"S\"
    }
  ],
  \"serviceName\": \"string\"
}" "http://example.com/api/services/new"
```

Request URL

```
http://example.com/api/services/new
```

Response Body

```
[
  {
    "id": 1,
    "serviceName": "string",
    "modules": [
      {
        "id": 1,
        "category": "Compute",
        "size": "S",
        "provider": "LibVirt",
        "config": "string",
        "name": "string"
      }
    ]
  }
]
```

5.2.5 Service

URI Path /api/services/{id}
Methods GET, PUT, DELETE, POST

5.2.6 GET

Curl

```
curl -X GET --header "Accept: */*" "http://example.com/api/
    services/1"
```

Request URL

```
http://example.com/api/services/1
```

Response Body

```
[
  {
    "id": 1,
    "serviceName": "string",
    "modules": [
      {
        "id": 1,
        "category": "Network",
        "size": null,
        "provider": "LibVirt",
        "config": "string",
        "name": "string"
      }
    ]
  }
]
```

5.2.7 PUT (Service update)

Curl

```
curl -X PUT --header "Content-Type: application/json" --
    header "Accept: */*" -d "{
  \"id\": 1,
  \"modules\": [
    {
      \"category\": \"Compute\",
```

```
      \"config\": \"string\",
      \"id\": 1,
      \"name\": \"string\",
      \"provider\": \"LibVirt\",
      \"size\": \"S\"
    }
  ],
  \"serviceName\": \"string\"
}" "http://example.com/api/services/1"
```

Request URL

```
http://example.com/api/services/1
```

5.2.8 POST (Service abonnieren)

```
curl -X POST --header "Content-Type: application/json" --
   header "Accept: */*" -d "{
  \"id\": 1,
  \"modules\": [
    {
      \"category\": \"Network\",
      \"config\": \"string\",
      \"id\": 1,
      \"name\": \"string\",
      \"provider\": \"LibVirt\",
      \"size\": \"null\"
    }
  ],
  \"serviceName\": \"string\"
}" "http://example.com/api/services/1"
```

Request URL

```
http://example.com/api/services/1
```

5.2.9 DELETE

```
curl -X DELETE --header "Accept: */*" "http://example.com/
 api/services/1"
```

Request URL

```
http://example.com/api/services/1
```

5.3 OrderedService Controller

Im OrderedService Controller befinden sich alle Funktionen um die abonnierten Services verwalten zu können.

5.3.1 OrderedServices

URI Path /OrderedServices
Methods GET

5.3.2 GET

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   orderedservices"
```

Request URL

```
http://example.com/api/orderedservices
```

Response Body

```
[
  {
    "id": 1,
    "identifiers": [
      {
        "id": 1,
        "name": "string",
```

```
        "uuid": "string",
        "category": "Storage",
        "size": "M",
        "provider": "LibVirt",
        "infos": {
          "name": "string",
          "capacity": "string"
        }
      }
    ],
    "orderedServiceName": "string"
  },
   {
    "id": 2,
    "identifiers": [
      {
        "id": 2,
        "name": "string",
        "uuid": "string",
        "category": "Compute",
        "size": "L",
        "provider": "LibVirt",
        "infos": {
          "name": "string",
          "vcpu": "string",
          "memory": "string"
        }
      }
    ],
    "orderedServiceName": "string"
  }
]
```


5.3.3 OrderedService

URI Path /api/OrderedServices/{id}
Methods GET, DELETE

5.3.4 GET

Curl

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   orderedservices/1"
```

Request URL

```
http://example.com/api/orderedservices/1
```

Response Body


```
[
  {
    "id": 1,
    "identifiers": [
      {
        "id": 1,
        "name": "string",
        "uuid": "string",
        "category": "Storage",
        "size": "M",
        "provider": "LibVirt",
        "infos": {
          "name": "string",
          "capacity": "string"
        }
      }
]
```


5.3.5 DELETE

Curl

```
curl -X DELETE --header "Accept: */*" "http://example.com/api
   /orderedservices/1"
```

Request URL

```
http://example.com/api/orderedservices/1
```

5.4 ServiceModule Controller

Im ServiceModules Controller befinden sich alle Funktionen um die Servicemodule verwalten zu können und alle Grössen,Provider und Kategorien zu holen.

5.4.1 ServiceModules

URI Path /api/servicemodules
Methods GET

5.4.2 GET

Curl

```
  curl -X GET --header "Accept: */*" "http://example.com/api/
   servicemodules"
```

Request URL

```
http://example.com/api/servicemodules
```

Response Body


```
[
  {
    "id": 1,
    "category": "Compute",
    "size": "M",
    "provider": "LibVirt",
    "config": "string",
    "name": "string"
  },
  {
    "id": 2,
    "category": "Compute",
```

```
    "size": "M",
    "provider": "LibVirt",
    "config": "string",
    "name": "string"
  }
]
```

5.4.3 ServiceModule

URI Path /api/servicemodules
Methods GET,PUT,POST,DELETE

5.4.4 GET

CURL

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   servicemodules/1"
```

Request URL

```
http://example.com/api/servicemodules/1
```

Response Body

```
{
  "id": 1,
  "category": "Compute",
  "size": "M",
  "provider": "LibVirt",
  "config": "string",
  "name": "string"
}
```

5.4.5 PUT (update ServiceModule)

CURL

```
curl -X PUT --header "Content-Type: application/json" --
   header "Accept: */*" -d "{
  \"category\": \"Compute\",
  \"config\": \"string\",
  \"id\": 1,
  \"name\": \"string\",
  \"provider\": \"LibVirt\",
  \"size\": \"S\"
}" "http://example.com/api/servicemodules/1"
```

Request URL

```
http://example.com/api/servicemodules/1
```

5.4.6 POST (neues ServiceModule)

```
curl -X POST --header "Content-Type: application/json" --
   header "Accept: */*" -d "{
  \"category\": \"Compute\",
  \"config\": \"string\",
  \"id\": 1,
  \"name\": \"string\",
  \"provider\": \"LibVirt\",
  \"size\": \"S\"
}" "http://example.com/api/servicemodules/new"
```

Request URL

```
http://example.com/api/servicemodules/new
```

Response Body

```
{
  "id": 1,
```

```
  "category": "Compute",
  "size": "S",
  "provider": "LibVirt",
  "config": "string",
  "name": "string"
}
```

5.4.7 DELETE

```
curl -X DELETE --header "Accept: */*" "http://example.com/api
   /servicemodules/1"
```

Request URL

```
http://example.com/api/servicemodules/1
```

5.4.8 Kategorien

URI Path /api/servicemodules/categories
Methods GET

5.4.9 GET

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   servicemodules/categories"
```

Response Body

```
[
  "Compute",
  "Storage",
  "Network"
]
```

5.4.10 Grössen

URI Path /api/servicemodules/sizes
Methods GET

5.4.11 GET

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   servicemodules/sizes"
```

Response Body

```
[
  "S",
  "M",
  "L"
]
```

5.4.12 Provider

URI Path /api/servicemodules/providers
Methods GET

5.4.13 GET

```
curl -X GET --header "Accept: */*" "http://example.com/api/
   servicemodules/providers"
```

Response Body

```
[
  "LibVirt"
]
```

5.5 Generic API

5.6 Operationen

5.6.1 LibVirt Connect

- connect(uri: String, ReadOnly: Boolean)

Contracts

Operation	connect(uri : String, ReadOnly: Boolean)
Cross References	UC01
Preconditions	Eine URI wurde mitgegeben und ReadOnly True oder False gesetzt
Postconditions	• Client ist mit dem Hypervisor verbunden

5.6.2 Compute

- createCompute(module: ServiceModule)
- deleteCompute(identifier: Identifier)

Contracts

Operation	createCompute(module: ServiceModule) : Identifier
Cross References	UC01
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Compute Instanz wurde erstellt und gibt Identifier zurück

Operation	deleteCompute(identifier: Identifier)
Cross References	UC01
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Compute Instanz wurde gelöscht

5.6.3 Storage

- createStorage(module: ServiceModule)
- deleteStorage(identifier: Identifier)

Contracts

Operation	createStorage(module: ServiceModule) : Identifier
Cross References	UC01
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Storage wurde erstellt und gibt Identifier zurück
Operation	deleteStorage(identifier: Identifier)
Cross References	UC02
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Storage wurde gelöscht

5.6.4 Network

- createNetwork(module: ServiceModule)
- deleteNetwork(identifier: Identifier)

Contracts

Operation	createNetwork(module: ServiceModule) : Identifier
Cross References	UC01
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Netzwerk wurde erstellt und gibt Identifier zurück
Operation	deleteNetwork(identifier: Identifier)
Cross References	UC02
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Netwerk wurde gelöscht

5.6.5 Allgemein

- getInformations(identifier : Identifier)

Contracts

Operation	getInformations(identifier: Identifier)
Preconditions	Client ist mit Hypervisor verbunden
Postconditions	• Informationen werden in einer Map zurückgegeben

5.7 Configfile

Das Configfile beinhaltet die Config für das jeweilige Servicemodul (Compute,Storage,Network), dabei gibt es für Storage und Compute 3 Grössen (S,M,L), dazu wird dann das nötige Configfile hinterlegt womit der Service konfiguriert und aufgesetzt wird. Mit Ausnahme von Network, wo es keine Grössen gibt. Die Nachfolgenden Konfigurationen stützen sich hauptsächlich auf Libvirt, bei Public Cloud Anbietern wird die Grösse, Betriebssystem, Storage und Netzwerkkonfiguration vielmals bereits vorgegeben.

5.7.1 Allgemein

Da je nach Bedarf Compute, Storage oder Netzwerk zuerst erstellt werden muss wird dies über einen Workflow behandelt, welcher entscheidet welches Servicemodul zuerst erstellt werden soll.

5.7.2 Compute

In Compute werden Memory, vCPU, Festplatte und Netzwerk (IP,VLAN) zugewiesen.

Name

Jeder Compute Instanz muss einen eindeutigen Namen gegeben werden, dieser wird durch den Workflow erstellt.

Memory

Memory wird entweder als fester Wert mitgegeben oder wird über die Instanzgrösse beim Cloud Anbieter vorbestimmt (micro,medium,large).

vCPU

vCPU's werden entweder als Wert mitgegeben oder durch den Cloud Anbieter vorgegeben (die Instanzgrösse).

Boot Disk

Ebenfalls muss noch eine Boot Disk übergeben werden, dies kann ein bereits bestehender Storage Pool,Volume oder eine Datei sein (Template mit vorinstallierter

Software) Die Grösse wird hier durch die ausgewählte Disk/Datei bestimmt, bei Cloud Anbietern jedoch durch die Instanzgrösse.

Network

Beim Netzwerk kann eine IP oder MAC Adresse mitgegeben oder automatisch zugewiesen werden, dies geschieht über eine Virtual Bridge oder Eingabe des VLANs.

5.7.3 Storage

Name

Storage besitzt einen festen eindeutigen Namen über welchen der Pool angesprochen werden kann.

Grösse

Der Storage benötigt eine gewisse Grösse um erstellt zu werden oder falls es sich um eine Partition oder Datei handelt wird sie dadurch vorgegeben.

Typ

Der Storage kann sowohl lokaler als auch Netzwerk Speicher sein, hier wird zwischen verschiedenen Typen unterschieden. Z.B.: können NFS Storages eingebunden werden oder GlusterFS bzw. Sheepdog.

5.7.4 Network

Name

Network besitzt einen eindeutigen Namen, welcher nur einmal auf dem System vorhanden sein darf.

IP Family

Je Nach Konfiguration muss noch angegeben werden ob IPv4 oder IPv6 Adressen verwendet werden sollen.

IP Range

Bei der Konfiguration Bspw.: einer Bridge kann auch noch ein IP Range mitgegeben werden, welcher an die Angeschlossenen Compute Instanzen an der Bridge verteilt werden sollen.

5.8 Architektur

5.9 Systemübersicht

5.10 Logische Architektur

- services: Web-Schnittstelle (RESTful)
- genericapi: Abstraktion für Compute, Storage und Network
- util: Tools zur Bearbeitung von Configfiles
- presistence: OR-Mapping

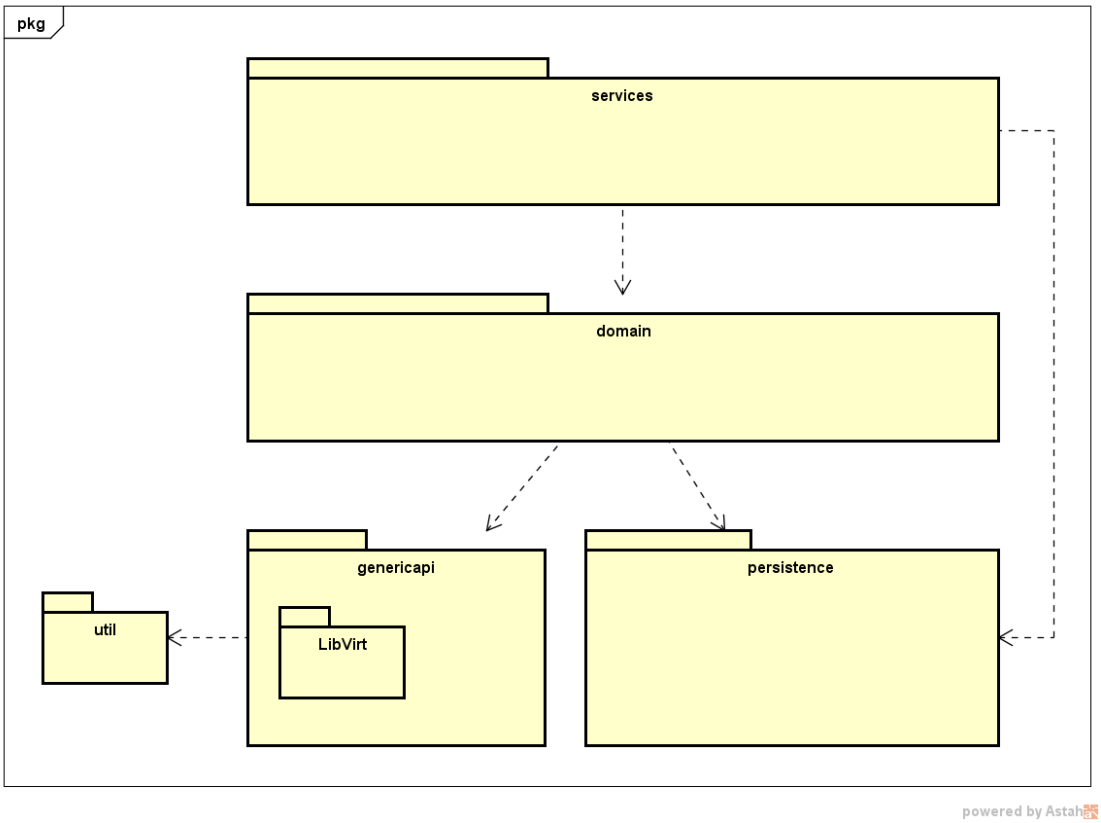


Abbildung 5.1: Logische Architektur

5.11 Klassenstruktur

5.11.1 services

In der Schicht services wird die RestAPI implementiert.

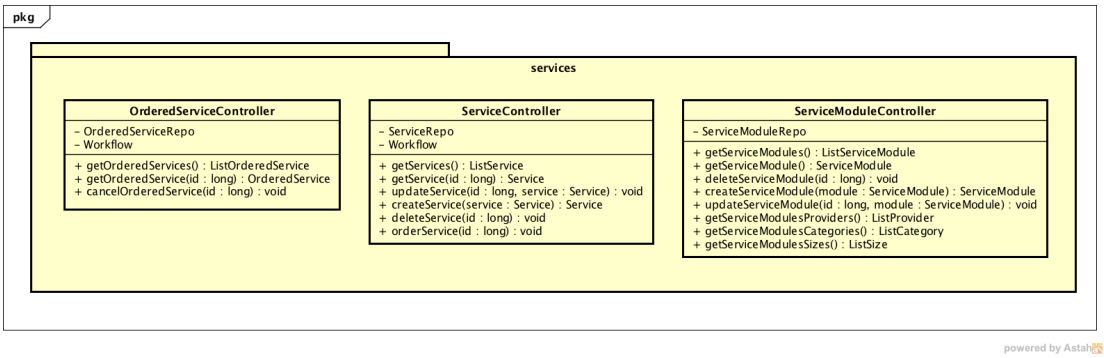


Abbildung 5.2: Klassendiagramm Services

5.11.2 domain

Die Domain beinhaltet die Entitäten, die mittels OR-Mapper mit der Datenbank synchronisiert werden. Der Workflow erstellt/löscht einen Service.

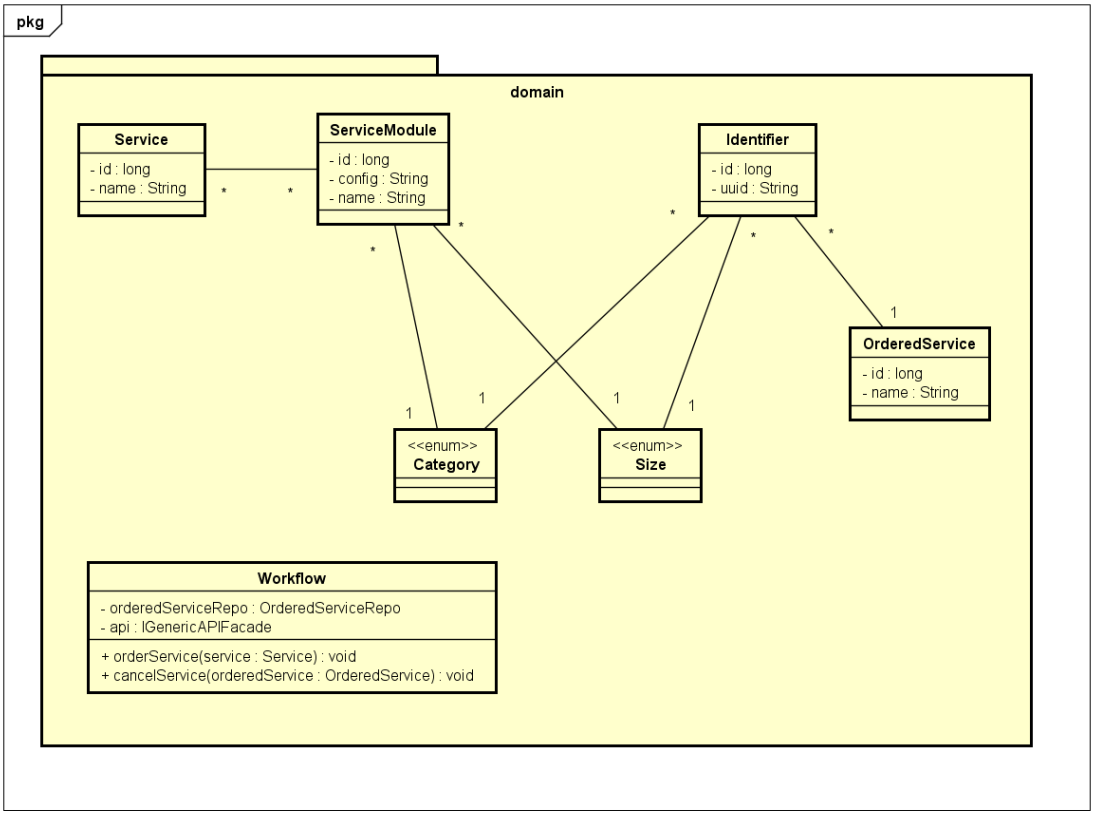


Abbildung 5.3: Klassendiagramm Domain

5.11.3 GenericAPI

ServiceModuleHandler

Der ServiceModuleHandler entscheidet an welchen ResourceController ein Service-Module/Identifier übergeben wird. Der Entscheidungsalgorithmus kann auf verschiedenste Arten implementiert werden, deshalb gibt es ein Interface.

ResourceController

Der ResourceController ist eine Abstraktion eines Controllers, die nicht auf irgendwelche Bibliotheksabhängigkeiten (z.B. LibVirt, JClouds) eingeht und lediglich dem ServiceModuleHandler die nötigen Informationen anbietet.

LibVirtController

Der LibVirtController ist eine weitere Abstraktion eines Controllers, die aber speziell einer Bibliothek (in diesem Fall LibVirt) zugewiesen ist. Die Klasse kümmert sich um die Instanziierung und kann, wenn nötig, Helfermethoden anbieten.

LibVirt<category>Controller

Die konkrete Klasse des Controllers implementiert nur noch die abstrakten Methoden des ResourceControllers.

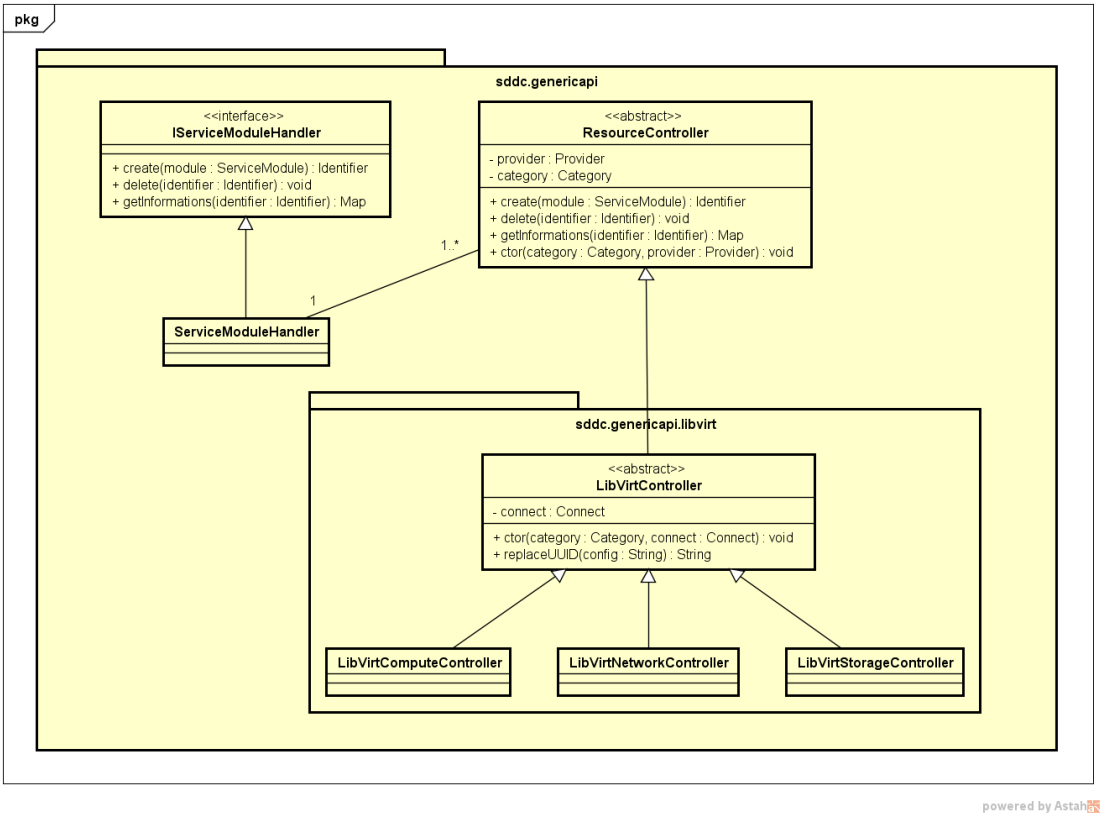


Abbildung 5.4: Klassendiagramm Libvirt Generic API

5.12 Dependency Injection

Die GenericAPI ist relativ komplex ausgefallen und kann erweitert werden. Um unnötigen Code in den oberen Schichten zu vermeiden, wird die GenericAPI mit Hilfe von Dependency Injection aufgesetzt. Es ist möglich in einem XML die gesamte GenericAPI nach belieben zusammenzusetzen.

5.13 Deployment

Spring liefert einen Application Server mit, bei welchem es sich um einen Tomcat Server handelt. Der Datenbankserver wird über den Java Database Connectivity (JDBC) Treiber angesprochen.

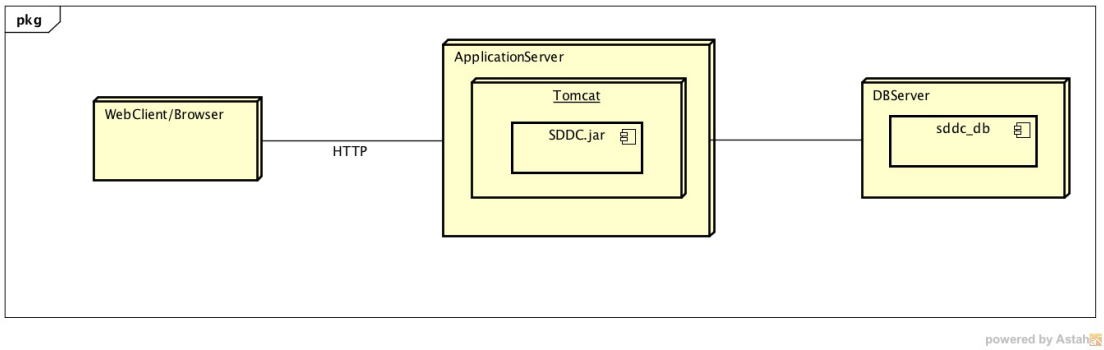


Abbildung 5.5: Deployment Diagramm

5.14 Persistierung

Für die Persistierung wird ein OR Mapper verwendet, wodurch die Java Objekte auf die Datenbank abgebildet werden.

5.14.1 Service

Der Service wird mit einer automatisch generierten ID und einem Namen abgespeichert, dabei ist der Name des Service Unique und kann nur einmal verwendet werden, was das auffinden eines Service vereinfacht.

5.14.2 Servicemodule

Das Servicemodul wird mit einer automatisch generierten ID abgespeichert und besitzt einen Namen (zur Identifizierung Unique). Category, Size und Provider sind Enums und geben feste Werte vor z.B.: Size: S,M,L. Das Configfile ist ein Text, wodurch es keine

Rolle Spielt ob nun ein XML, JavaScript Object Notation (JSON) oder YAML Ain't Markup Language (YAML) abgespeichert wird.

5.14.3 Orderedservice

Der Orderedservice besitzt eine automatisch generierte ID + einen nicht Uniquen Namen, um einen Service auch mehrfach abonnieren zu können.

5.14.4 Identifier

Der Identifier besitzt eine generierte ID + den Namen des dazugehörigen Servicemodul. Kategorie, Size und Provider werden ebenfalls vom Servicemodul übernommen. Die Universally unique identifier (UUID) ist je nach Provider verschieden und wird durch diesen vorgegeben.

5.14.5 Identifier Infos

Die Identifier Infos beinhalten die vorhanden Infos zu einem abonnierten Servicemodul, welche auf dem Dashboard ausgegeben werden.

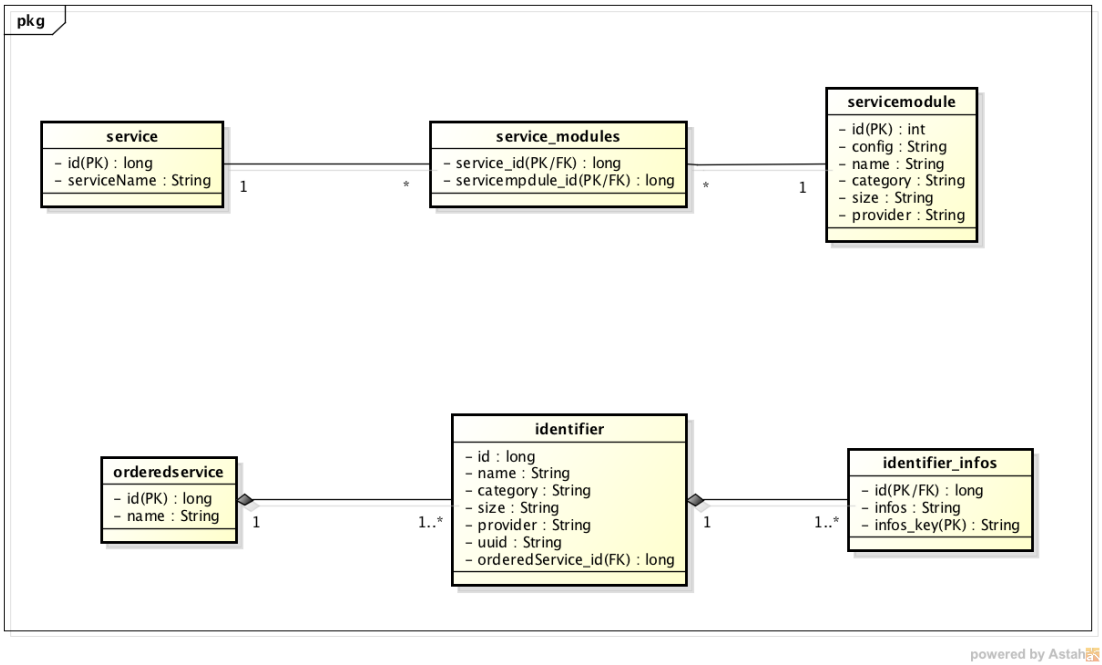


Abbildung 5.6: Datenmodell

5.15 Sequenzdiagramme

5.15.1 service bestellen

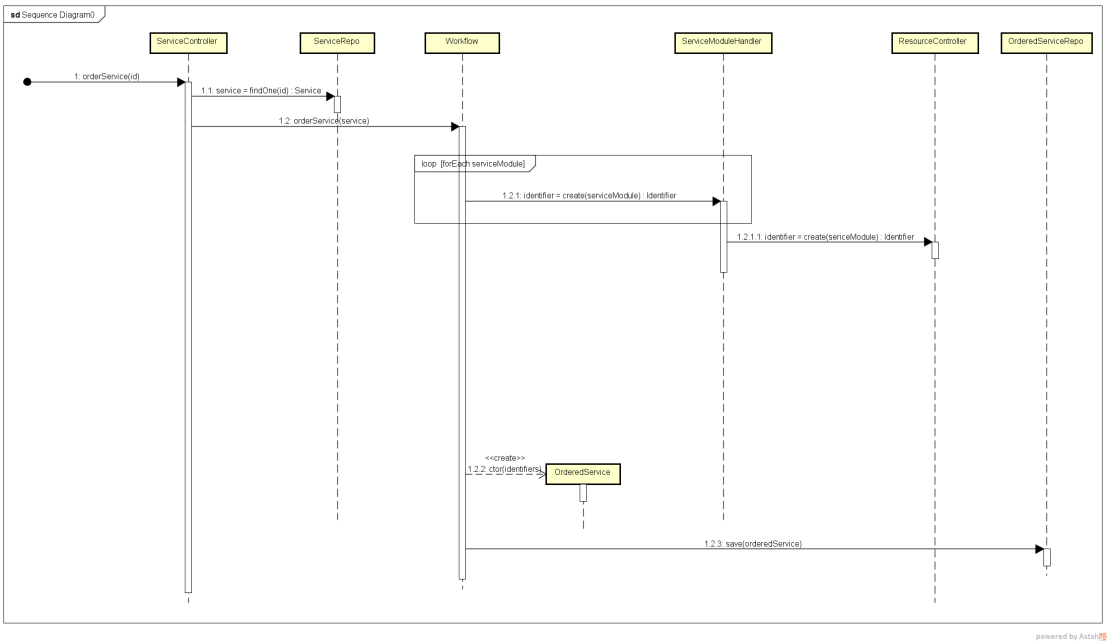


Abbildung 5.7: Sequenzdiagramm Service bestellen

Teil III

Ergebnis

6 Resultat

Das Resultat umfasst einen kompletten Webservice mit RESTful API gegen die ein Dashboard entwickelt wurde.

6.1 Dashboard

Das Dashboard demonstriert die Möglichkeiten des Webservices. Es ist möglich Services/Servicemodule zu erstellen, bearbeiten, anzusehen und wieder zu löschen. Moderne Frameworks und Libraries, wie AngularJS und Bootstrap ermöglichen ein responsives Design sowie eine klare Abkopplung von Model und View.

6.2 RESTful API

Die RESTful API ermöglicht verschiedensten Aktoren (z.B. Business Applications) den Zugriff auf den Webservice. Es ist also möglich das andere Systeme über unsere Software Services (z.B. neue Compute-Instanz) erstellen. Die RESTful API wurde mit dem Spring Framework implementiert.

6.3 Workflow

Der Workflow konzentriert sich auf das Wesentliche und ist daher konzeptionell erweiterbar. Grundsätzlich kann davon ausgegangen werden, dass zuerst die Netzwerkeinstellungen und danach die Storage und Compute Instanzen abgearbeitet werden müssen. Falls ein Fehler während eines Prozesses auftritt wird der gesamte Service wieder gelöscht (Rollback). Eine Fehlerbehebung während des Prozesses ist praktisch unmöglich.

6.4 Persistence

Für die Persistierung der Daten kommt Postgres zum Einsatz. Die Struktur der Datenbank wird über einen OR-Mapper aus der Domain erstellt.

6.5 Generic API

Die GenericAPI ermöglicht das Zusammenstellen von verschiedensten Kontrollern. Dies geschieht über eine XML Datei. Es wird ein simples aber strikt einzuhaltendes Interface geboten, das die Implementation neuer Kontroller ermöglicht. Durch den modularen Aufbau kann die Software für verschiedenste Umgebungen erweitert werden.

7 Ausblick

Die Software ist im Verlauf des Projekts gewachsen. Wir haben die Architektur immer wieder verfeinert um eine möglichst saubere Lösung zu schaffen. Während dessen haben sich neue Möglichkeiten und Ideen eröffnet. Der Workflow erfüllt seinen Zweck, er könnte jedoch noch intelligenter sein. Die Service Module könnten in einer Graphstruktur angeordnet werden. Das erlaubt eine reibungslose Abarbeitung der Servicemodule, bei dem die Ressourcen automatisiert verbunden werden.
Es ist jetzt schon relativ einfach neue Kontroller zu implementieren und auch die Konfiguration der generischen API ist gelöst. Es wäre schön wenn neue Kontroller wie Plug-ins behandelt werden könnten. Der Umbau wäre nicht übermässig komplex, benötigt aber etwas Zeit.

Teil IV

Appendix

Literatur

[1] *Python library for interacting with many of the popular cloud service providers using a unified API.* URL: https://libcloud.apache.org/.

[2] *The Java Multi-Cloud Toolkit.* URL: https://jclouds.apache.org/.

[3] *Erlang wrapper around Libcloud.* URL: https://github.com/esl/elibcloud.

[4] *The Ruby cloud services library.* URL: https://github.com/fog/fog/.

[5] *pkgcloud is a standard library for node.js that abstracts away differences among multiple cloud providers.* URL: https://github.com/pkgcloud/pkgcloud.

[6] *libvirt: The virtualization API.* URL: http://libvirt.org/.

[7] *Bitnami Launchpad for AWS.* URL: https://aws.bitnami.com.

[8] *Bitnami Launchpad for DigitalOcean.* URL: https://digitalocean.bitnami.com.

[9] *Bitnami Launchpad for Azure.* URL: https://azure.bitnami.com.

[10] *Bitnami Launchpad for Google.* URL: https://google.bitnami.com.

[11] *Bitnami Launchpad for VMware.* URL: https://vmware.bitnami.com.

[12] *Secure email: ProtonMail is free encrypted email.* URL: https://protonmail.ch.

[13] Craig Larman. *UML 2 und Patterns angewendet Objektorientierter Softwareentwicklung*. mitp, 2005. ISBN: 978-3-8266-1453-8.

Glossar

GenericAPI Die GenericAPI beschreibt den Teil der Software in dem auf das Data Center zugegriffen wird. 33, 88, 93

Ressource Eine Ressource beschreibt irgendeine Entität in einem Data Center z.B. eine Compute-Instanz. 5

Service Ein Service bündelt mehrere Servicemodul zu einem Gesamtpaket, das von der SDDC-Software angeboten wird. 46–48

Servicemodul Das Servicemodul beinhaltet alle nötigen Informationen um eine Ressource in einem Data Center zu instantiieren. 48

Workflow Der Workflow entscheidet in welcher Reihenfolge die Servicemodul in einem Service abgearbeitet werden. 1, 5

Abkürzungsverzeichnis

API Application Programming Interface

IaaS Infrastructure as a Service

PaaS Platfrom as a Service

SaaS Software as a Service

XML Extensible Markup Language

YAML YAML Ain't Markup Language

JSON JavaScript Object Notation

UUID Universally unique identifier

JDBC Java Database Connectivity