Projektplan

Silvan Adrian
Fabian Binna

1 Änderungshistorie

Datum	Version	Änderung	Autor
17.09.15	1.00	Erstellung des Dokuments	Gruppe
18.09.15	1.01	Organisationsstruktur + Einführung + Projektübersicht	Silvan Adrian
18.09.15	1.02	Qualitätsmassnahmen	Fabian Binna
20.09.15	1.03	Text verbessert und zusätzliche Infos eingefügt	Silvan Adrian
25.09.15	1.04	Verbesserungen	Silvan Adrian
25.09.15	1.05	Sprint Planungsübersicht eingefügt	Silvan Adrian
26.09.15	1.06	Testing erweitert/verbessert + Logging	Fabian Binna

Inhaltsverzeichnis

2 Einführung

2.1 Zweck

Dieses Dokument beschreibt die Planung der SA SDDC.

2.2 Gültigkeitsbereich

Dieses Dokument ist während des ganzen Projekts gültig und wird laufend aktualisiert.

2.3 Referenzen

2.3.1 APIs:

http://developer.openstack.org/

2.3.2 SCRUM Guide:

http://www.scrumguides.org

3 Projektübersicht

Es soll eine generische API erstellt oder eine bestehende API verwendet werden, um über ein Dashboard direkt Services zu abonnieren, dabei werden direkt Services von verschiedenen Cloud Anbietern angeboten und in einem Dashboard zusammengefasst (IaaS,PaaS,SaaS). Die API soll dabei modular aufgebaut sein, damit neue Anbieter einfach hinzugefügt werden können (z.B.: als Plugins).

3.1 Zweck und Ziel

Gelerntes aus verschiedenen Modulen anwenden, sich mit der gestellten Aufgabenstellung auseinandersetzen und diese umsetzen.

3.2 Primäre Features

- Generische API
- Spartanisches Dashboard
- Unterstützung von Private Cloud Anbietern
- Unterstützung von Docker
- Unterstützung von ausgewählten Public Cloud Anbietern

3.3 Erweiterte Features

- Unterstützung für mehr Public Cloud Anbieter
- Dashboard um Funktionen erweitern und verbessern
- Plugins über Dashboard installieren

3.4 Lieferumfang

- Source-Code (Dashboard/API)
- Dokumentation (Projekt und Software)
- Benutzerdokumentation

3.5 Annahmen und Einschränkungen

Die Applikation setzte sich aus einem Dashboard und einer generischen API zusammen, beides sollte dabei verteilt betrieben werden können. Die generische API soll dabei so viele Anbieter wie möglich abdecken.

4 Projektorganisation

Wir setzen beim Project auf Scrum, wodurch lediglich die Projektrollen Product Owner, Entwickler und Scrum-Master vorhanden sind.

4.1 Organisationsstruktur

4.2 Externe Schnittstellen

Projektbetreuer:
Beat Stettler (Beat.Stettler@ins.hsr.ch)
Urs Baumann (Urs.Baumann@ins.hsr.ch)

5 Managment Abläufe

5.1 Kostenvoranschlag

Gemäss Vorgaben wird mit 480 Stunden gesamt gerechnet + die einberechneten Risiken von 41 Stunden.
Gesamtaufwand: 521 Stunden
Risikoaufwand: 41 Stunden
Aufwand pro Woche: ca.: 37 Stunden
Aufwand pro Woche pro Teammitglied: ca.: 19 Stunden

5.2 Sprints

Sprint	Beschreibung	Beginn	Ende
Sprint 1	Grundgerüst API, API ansprechbar und Services können abonniert werden, allerdings nicht über ein Dashboard sondern direkt über API	05.10.15	19.10.15
Sprint 2	Sehr einfaches Dashboard für einen Anbieter, um Anbieter spezifische Services abonnieren zu können	19.10.15	02.11.15
Sprint 3	Zusätzliche Anbieter einbinden und deren spezifische Services	02.11.15	16.11.15
Sprint 4	Dashboard erweitern und verbessern (Login, Administrationsmöglichkeiten)	16.11.15	30.11.15
Sprint 5	API & Dashboard abschliessen und letzter Release	30.11.15	07.12.15

5.3 Besprechungen

Daily Meetings (ca. 10 - 15min) werden während der Pause oder über Skype stattfinden, um sich gegenseitig über Probleme, abgeschlossene Arbeitspakete etc. zu informieren. Nach jedem Abschluss eines Sprints wird ein Sprintreview geplant und durchgeführt, um zu prüfen ob alles gemäss Plan erledigt wurde + die aktuellste Version der Software released.

6 Risikomanagement

6.1 Risiken

Risiken werden im Dokument Initiales_Risikomanagement.pdf beschrieben

6.2 Umgang mit Risiken

Für die Risiken werden Reserven eingeplant. Die Reserven werden direkt in die einzelnen Tickets eingerechnet. Falls Risiken eintreffen werden diese sofort an einem der Daily-Meetings kommuniziert und mögliche Lösungen evaluiert.

7 Arbeitspakete

Die Arbeitspakete werden in Open Project erstellt und gepflegt. Lesender Zugriff ist anonym möglich, schreibender nur eingeloggt (Projekt ist öffentlich).
Link zur Open Project Instanz : http://sddc.silvn.com

8 Infrastruktur

8.1 Entwicklungsinfrastruktur

Name	Hardware	Betriebssystem	IDE
Silvan Adrian	MacBook Pro	OSX 10.10.5	N/A
Fabian Binna	Lenovo T430s	Windows 10	N/A

8.2 Tools/Software

- **BuildServer:** Travis-CI
- **Versionsmanagement:** GIT
- **Notifications:** Slack
- **Wireframing:** Pencil

8.3 Kommunikationsmittel

- E-Mail
- Skype
- Open Project (Kommentare)
- GitHub (Issues,Kommentare)
- Whatsapp
- Slack

9 Qualitätsmassnahmen

9.1 Dokumentation

Die Dokumentation befindet sich auf einem privaten GitHub Repository. Die Texte werden in LaTex geschrieben. Die Dokumente werden versioniert.

`https://github.com/silvanadrian/SDDC_Doku.git`

9.2 Projektmanagement

Für das Projektmanagement wird OpenProject verwendet.

`sddc.silvn.com`

9.3 Entwicklung

9.3.1 Unit Testing / Test-Driven Development

Die Unit Tests kommen in einen separaten Ordner "Test". Es wird eine möglichst hohe Code Coverage angestrebt. Die Code Coverage wird mit einem Tool (z.B. eclEmma) sichergestellt.

Die Klassen werden mit Hilfe von Test-Driven Development implementiert.

9.3.2 Code Review

Nach jedem Sprint oder bei Abschluss grosser Arbeitspakete wird ein Code Review durchgeführt.

Die Review Protokolle werden auf dem GitHub Repository SDDC-Doku abgelegt.

9.3.3 Logging

Ein Logging-System (z.B. log4j) untestützt die Entwicklung und später das Operating.

TRACE: Ausführliches Debugging
DEBUG: allgemeines Debugging
INFO: allgemeine Informationen
WARN: Auftreten einer unerwarteten Situation
ERROR: Fehler
FATAL: Kritischer Fehler

9.3.4 Metrikanalyse

Die Metriken werden nach jedem Sprint protokolliert und Analysiert. Die Metriken helfen während einem Code-Review auf problematische Code Stücke hinzuweisen. Sie können auch beim Debuggen hilfreich sein.

9.3.5 Code Style Guidelines

Editor Standard

9.4 Testen

9.5 Modultest

Die Modultests werden, wie schon beschrieben ,während der Entwicklung mit Test-Driven Development erstellt. Zu jedem Modul werden Tests geschrieben, um eine möglichst hohe Code Coverage zu erreichen. Die Tests werden nochmals auf dem Build Server ausgeführt

um “it works on my machine” abzudecken.

Produkte:

- Irgend eine Unit-tester (je nach Sprache)
- Travis

9.6 Integrationstest

Bei den Integrationstests kommt eine Kombination aus automatisierten Tests, sowie eine Analyse des Ablaufs zum Einsatz. Die Tests werden an verschiedenen Schichten angesetzt (Presentation und Business-Layer). Die Unteren Schichten, die für den Integrationstest nicht relevant sind werden simuliert. Nachdem die automatische Tests durchgelaufen sind, werden die Log-Daten ausgewertet und mit den Sequenzdiagrammen abgeglichen. Es ist wichtig, das die Komponenten korrekt miteinander kommunizieren.

Produkte:

- Irgend ein Unit-tester (je nach Sprache)
- Für Presentation-Layer kommt `http://karma-runner.github.io/0.13/index.html`Karma zum Einsatz.
- Log-Daten

9.7 Systemtest

Die Systemtests richten sich nach den Use Cases. Auch hier kann mit Karma getestet werden (Das wäre dann eher ein sehr grosser Integrationstest). Die Use Cases werden von einem Tester nach Protokollangaben durchgeführt und dokumentiert. Auch hier können Log-Daten eine Hilfe für die Analyse sein.

Systemtests werden nach jedem Sprint durchgeführt um die geplante Funktionalität zu testen und so die neuste Version der Software releasen zu können.

Produkte:

- Karma
- Protokoll
- Log-Daten

9.8 Abnahmetest

Der Abnahmetest wird vom Kunden selbst durchgeführt.