

Platform as a Service

Datenverarbeitung in der Cloud

Das Studienheft und seine Teile sind urheberrechtlich geschützt. Jede Nutzung in anderen als den gesetzlich zugelassenen Fällen ist nicht erlaubt und bedarf der vorherigen schriftlichen Zustimmung des Rechteinhabers. Dies gilt insbesondere für das öffentliche Zugänglichmachen via Internet, Vervielfältigungen und Weitergabe. Zulässig ist das Speichern (und Ausdrucken) des Studienheftes für persönliche Zwecke.

© Fernstudienzentrum Hamburg · Alle Rechte vorbehalten

Falls wir in unseren Studienheften auf Seiten im Internet verweisen/verlinken, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf Inhalt und Gestaltung haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

CCOM05B

Platform as a Service

Datenverarbeitung in der Cloud

Autor: Sebastian Stein und Torsten Schreiber

Fachlektor: Christoph Siebeck

Werden Personenbezeichnungen aus Gründen der besseren Lesbarkeit nur in der männlichen oder weiblichen Form verwendet, so schließt dies das jeweils andere Geschlecht mit ein.

Falls wir in unseren Studienheften auf Seiten im Internet verweisen, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf die zukünftige Gestaltung und den Inhalt der Seiten haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

Platform as a Service

Datenverarbeitung in der Cloud

Inhaltsverzeichnis

Einleitung	1
1 Platform as a Service	3
1.1 Serviceorientierte Architektur (SOA)	3
1.2 PaaS als SOA-Nachfolger	4
1.3 PaaS-Typen	5
1.4 Architektur von Anwendungen auf PaaS	6
1.5 Verfügbarkeit von Anwendungen auf PaaS	9
1.6 Kompatibilität	10
Zusammenfassung	11
2 PaaS im Praxiseinsatz	13
2.1 Heroku, GitHub und Amazon S3	13
2.2 Nutzerkonten anlegen	14
2.3 Entwicklungsumgebung einrichten	21
2.3.1 Heroku CLI installieren	22
2.3.2 Git installieren und konfigurieren	23
2.4 Anwendung bereitstellen	37
2.5 Anwendung ausprobieren	41
Zusammenfassung	43
3 Schlussbetrachtung	45
Anhang	
A. Lösungen der Aufgaben zur Selbstüberprüfung	46
B. Glossar	47
C. Weiterführende Webseiten	50
D. Abbildungsverzeichnis	51
E. Sachwortverzeichnis	53
F. Einsendeaufgabe	55

Einleitung

Herzlich willkommen zum Studienheft Platform as a Service (PaaS) – Datenverarbeitung in der Cloud. Nachdem wir uns in anderen Heften bereits ausführlich mit virtuellen Maschinen, Cloud-Betriebssystemen und Software as a Service beschäftigt haben, werden wir in diesem Heft die Eigenschaften von PaaS-Angeboten kennenlernen und als praktische Übung eine Anwendung auf einer Cloud-Plattform bereitstellen.

PaaS wird, als eine der drei grundlegenden Cloud-Computing-Arten, im Cloud Computing Stack zwischen Software as a Service (SaaS) und Infrastructure as a Service (IaaS) eingeordnet.

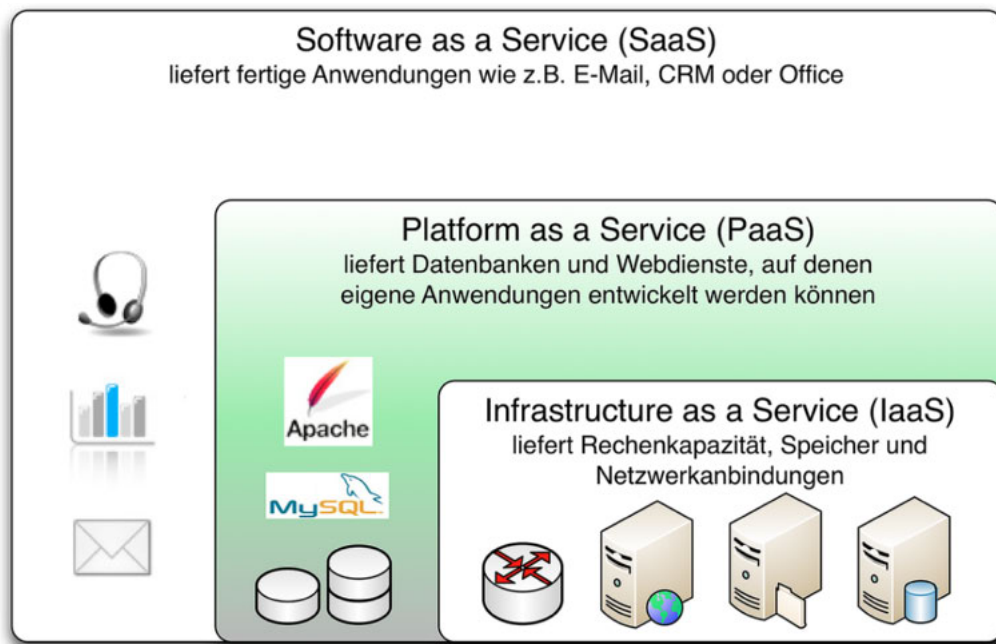


Abb. 0.1: Platform as a Service im Cloud Computing Stack

Anders als z.B. bei SaaS-Angeboten werden bei PaaS-Angeboten keine fertigen Anwendungen bereitgestellt. PaaS-Angebote stellen heute, wie der Name schon vermuten lässt, „nur“ eine hoch skalierbare und ausfallsichere Plattform für die Bereitstellung von webbasierten Anwendungen bereit. Das heißt, es werden hauptsächlich Laufzeitumgebungen für webbasierte Anwendungen und Datenbankdienste bereitgestellt.

Nach dem Durcharbeiten dieses Heftes werden Sie in der Lage sein:

- PaaS in den Kontext des Cloud Computings einzuordnen,
- den Einsatzzweck von PaaS-Angeboten zu erklären,
- die den PaaS-Angeboten zugrunde liegenden Technologien zu nennen,
- mögliche Problemfelder beim Einsatz von PaaS-Angeboten zu nennen,
- eine Anwendung auf der Cloud-Computing-Plattform Heroku bereitzustellen.

Sebastian Stein und Torsten Schreiber

1 Platform as a Service

In dieser Lektion lernen Sie einige Eigenschaften von Platform as a Service kennen. Sie erfahren, was in diesem Zusammenhang unter serviceorientierter Architektur zu verstehen ist. Außerdem stellen wir Ihnen die verschiedenen PaaS-Typen kurz vor und schauen uns die Architektur von Anwendungen auf PaaS an.

„Platform as a Service“-Angebote stellen hauptsächlich Laufzeitumgebungen für web-basierte Anwendungen und Datenbankdienste bereit. Sie richten sich damit hauptsächlich an die sogenannten Independent Software Vendors (ISV¹) und an Unternehmen, die ihre Geschäftsanwendungen selbst programmieren und bereitstellen wollen.

Neben der Bereitstellung einer Laufzeitumgebung und Datenbanken für die produktive Instanz einer Anwendung stellen PaaS-Angebote in der Regel auch Laufzeitumgebungen für Entwicklungs- und Testinstallationen der Anwendung bereit. Zusätzlich zu den Laufzeitumgebungen und Datenbanken werden sehr häufig weitere Werkzeuge, die die Entwicklung bzw. den gesamten Lebenszyklus von Software unterstützen, mit angeboten. Damit sollen PaaS-Angebote die Entwicklung und Bereitstellung von neuen web-basierten Anwendungen beschleunigen. Ein Entwickler (-Team) kann somit sehr schnell eine neue innovative Anwendung im Internet bereitstellen, ohne sich um die Einrichtung der Laufzeitumgebungen und Datenbanken, die Beschaffung von Servern oder die Anmietung von Rechenzentrumsflächen kümmern zu müssen. Viele der im letzten Heft behandelten SaaS-Angebote werden über ein PaaS-Angebot bereitgestellt.

1.1 Serviceorientierte Architektur (SOA)

Der Cloud-Computing-Ansatz PaaS hat seine Wurzeln in der sogenannten Serviceorientierten Architektur (SOA). SOA haben wir bereits in einem anderen Heft kurz behandelt.

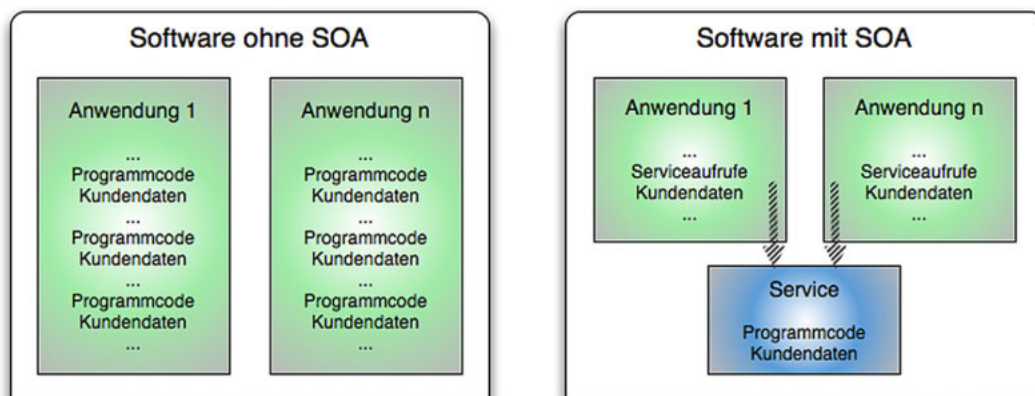


Abb. 1.1: SOA-Prinzip

1. ISV – Independent Software Vendor (engl. für unabhängige Softwareanbieter) sind Unternehmen, deren Kerngeschäft die Produktion und der Verkauf von Software ist.

Die Abbildung 1.1 stellt den Ansatz dar, den das SOA-Architekturmuster verfolgt. Anwendungen werden dabei in wiederverwendbare Dienste aufgeteilt und so koordiniert, dass der Zweck der übergeordneten Anwendung erfüllt wird. Damit wird der für eine komplexe Anwendung neu zu entwickelnde Code minimiert und dadurch die Entwicklung beschleunigt.

Dieses Prinzip wird immer wieder auf verschiedenen Ebenen innerhalb der Informationstechnologie angewendet. Auf der Ebene der IT-Infrastruktur werden nach dem SOA-Prinzip z.B. Datenbanken und Webserver so koordiniert, dass diese eine Plattform für eine Unternehmenssoftware bilden. D.h., es werden standardisierte Datenbankserver für mehrere Anwendungen gleichzeitig genutzt und nicht für jede neue Anwendung ein neuer Datenbankserver in Betrieb genommen.

Um die Wiederverwendbarkeit von webbasierten Diensten zu erhöhen, wurden Protokolle entwickelt, über die Anwendungen mit webbasierten Diensten kommunizieren können. Zwei dieser Protokolle sind SOAP und XML-RPC. Beide Protokolle basieren auf XML² und dienen dazu, Daten zwischen verschiedenen Systemen auszutauschen und Prozeduren auf entfernten Systemen aufzurufen. Um die Verwendung von durch SOA entstandenen Diensten zu vereinfachen, wurde mit UDDI (Universal Description, Discovery and Integration) ein Verzeichnisdienst standardisiert, der es ermöglichen sollte, im Internet verfügbare webbasierte Dienste sowie deren Ein- und Ausgabe-Parameter ausfindig zu machen. Allerdings wird UDDI nur selten benutzt und hat dadurch nur wenig Bedeutung.

1.2 PaaS als SOA-Nachfolger

Der SOA-Hype der 2000er-Jahre versprach den Anwendungsentwicklern standardisierte Kommunikationskanäle zwischen webbasierten Diensten sowie eine höhere Effizienz in der Anwendungsentwicklung, da bereits vorhandener Code sehr einfach wiederverwendet werden konnte. Durch SOA sollten die Entwicklung von Unternehmensanwendungen beschleunigt und die Kosten verringert werden. Tatsächlich wurden in den meisten Fällen die Kosten höher und die Entwicklungszeit länger, da zusätzliche Arbeit in die Standardisierung des Codes gesteckt werden musste und die Wiederverwendbarkeit nur selten zum Tragen kam.

„Platform as a Service“-Angebote machen den Anwendungsentwicklern ähnliche Versprechen. Die Entwicklung von Anwendungen soll kostengünstiger und schneller voranschreiten. Zusätzlich soll die Plattform, auf der die Anwendung ausgeführt wird, automatisch skalieren und nur so viel Ressourcen in Rechnung gestellt werden, wie tatsächlich verbraucht werden. Der Entwickler soll sich nicht mehr um Infrastruktur-Themen, wie die Bereitstellung von Servern, Laufzeitumgebungen und Speicherplatz kümmern müssen. Die IT-Infrastruktur wird für den Anwendungsentwickler unsichtbar (durch Abstraktion verborgen) und er kann sich auf seine eigentliche Aufgabe, die Entwicklung seiner Anwendung, konzentrieren.

2. XML ist eine Abkürzung für Extensible Markup Language. Ins Deutsche übersetzt heißt das erweiterbare Auszeichnungssprache. XML wird verwendet, um strukturierte Daten zwischen verschiedenen Systemen (häufig über das Internet) auszutauschen.

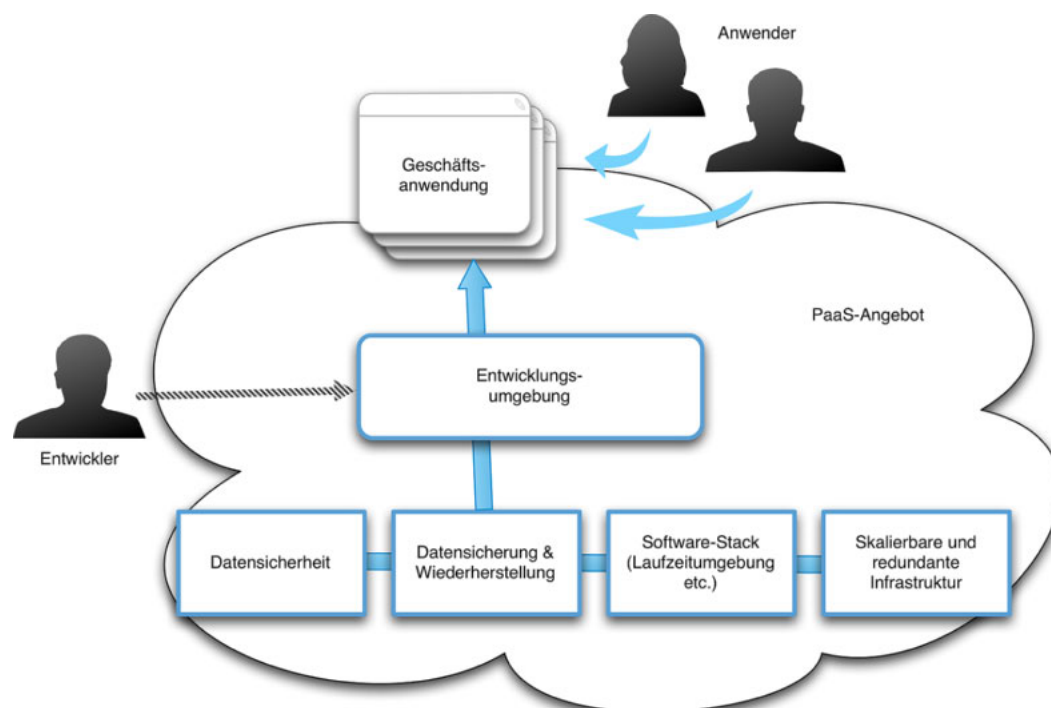


Abb. 1.2: PaaS-Modell

Ob PaaS ein Nachfolger von SOA ist oder SOA einfach nur grundlegende Bausteine für PaaS geliefert hat, ist umstritten. Fest steht, dass der Erfolg von SOA durch die zusätzliche Komplexität, die durch die Strukturierung der einzelnen webbasierten Dienste hervorgerufen wird, stark gebremst worden ist. PaaS-Angebote verstecken diese Komplexität in ihren vorgefertigten APIs undbürden dem Entwickler keine zusätzlichen Aufgaben auf. Aus diesem Grund wird PaaS ein größerer Erfolg als SOA vorausgesagt.

PaaS hat aber auch einen entscheidenden Nachteil gegenüber dem damaligen SOA-Ansatz: Anwendungen, die auf einem PaaS-Angebot ausgeführt werden, werden nicht auf eigenen Servern ausgeführt, sondern in der Cloud. D.h., es eignet sich bisher nur für in sich abgeschlossene Anwendungen, die nicht auf Live-Daten aus dem eigenen Rechenzentrum zugreifen müssen. Dies ist ähnlich zu SaaS-Angeboten. Im Vergleich zu SaaS sind PaaS-Angebote aber ein wenig flexibler, da die eigentliche Anwendung in der Regel erst entwickelt oder angepasst werden muss und somit mehr Raum für Schnittstellen zum Datenaustausch mit dem eigenen Rechenzentrum vorhanden sind.

1.3 PaaS-Typen

PaaS-Angebote unterscheiden sich in ihrem Anwendungszweck, den Laufzeitumgebungen für verschiedene Programmiersprachen, verschiedene Datenspeichermethoden (Datenbanken etc.) und den verschiedenen Entwicklungswerkzeugen. Wenn man die verschiedenen PaaS-Angebote nach ihren unterschiedlichen Anwendungszwecken unterscheidet, kann man sie in die folgenden drei Kategorien einordnen.

- aPaaS – Application PaaS
- iPaaS – Integration and Governance PaaS
- Add-on-Entwicklungsumgebungen

Hierbei stellt aPaaS den Standardtyp der PaaS-Angebote dar. Es handelt sich dabei um die Bereitstellung einer Plattform zur Entwicklung und Ausführung einer Anwendung. Bekannte Beispiele dieses PaaS-Typs sind Microsoft Azure, force.com oder Heroku. iPaaS hingegen stellt klassische Middleware-Dienste³ bereit und vermittelt Daten zwischen verschiedenen Plattformen und On-Premise- sowie Off-Premise-Anwendungen. Das bekannteste iPaaS-Angebot ist Mule iON. Der dritte Typ, die Add-on-Entwicklungsumgebungen sind Erweiterungen von SaaS-Angeboten wie Google Apps (Office Suite), die es erlauben, die bekannten Office-Anwendungen um eigene Funktionen zu erweitern. Dies ist vergleichbar mit der Möglichkeit, Makros in Microsoft-Office-Anwendungen zu erstellen.

1.4 Architektur von Anwendungen auf PaaS

Eine moderne webbasierte Anwendung, die auf einem PaaS-Angebot entwickelt und betrieben werden soll, besteht prinzipiell aus drei Teilen:

- grafische Benutzeroberfläche
- Anwendung (Web-Services)
- permanenter Datenspeicher

Die grafische Benutzeroberfläche einer Anwendung kümmert sich um die Darstellung der Daten und bietet dem Nutzer Möglichkeiten, Daten einzugeben, auszugeben und Funktionen der Anwendung aufzurufen. Anwendungen, die auf einem PaaS-Angebot entwickelt und ausgeführt werden, erhalten häufig eine webbasierte und/oder eine mobile Oberfläche zur Ausführung auf einem Smartphone.

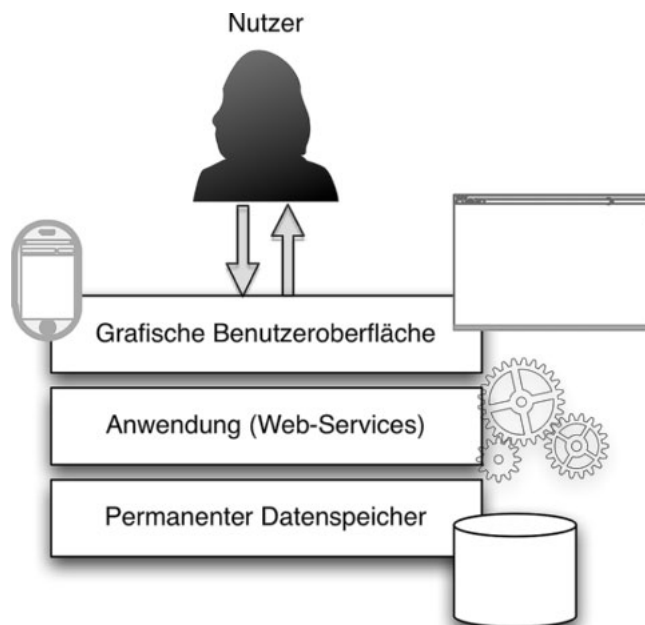


Abb. 1.3: Architektur von Anwendungen

3. Klassische Middleware-Dienste sind z.B. Meta-Verzeichnisse, die Daten zwischen mehreren Verzeichnisdiensten (z.B. Personal-Datenbank und Active Directory) miteinander synchronisieren.

Die eigentliche Anwendung besteht aus Prozeduren und Funktionen zur Datenbearbeitung und wird in der Regel als Web-Service implementiert. Diese Web-Services haben definierte Ein- und Ausgabe-Parameter und wenden die gewünschten Berechnungen auf die Daten an.

Zur permanenten Speicherung der Daten einer Anwendung werden in PaaS-Angeboten verschiedene Technologien angeboten. Zu den am weitesten verbreiteten PaaS-Speichertechnologien zählen:

- SQL-Datenbanken
- NoSQL-Datenbanken
- BLOB-Speicher

SQL-Datenbanken zählen zu den relationalen Datenbanken und werden häufig in On-Premise-Unternehmensanwendungen zur konsistenten Speicherung der Anwendungsdaten verwendet. Um die Migration einer On-Premise-Unternehmensanwendung auf ein PaaS-Angebot zu erleichtern, werden in vielen PaaS-Angeboten auch SQL-Datenbanken (oft mySQL) unterstützt.

Da relationale Datenbanken (z.B. SQL-Datenbanken) zwar geclustert werden können, aber den Nachteil haben, dass eine Transaktion mit jedem weiteren Clusterknoten länger dauert, können diese nicht horizontal skaliert werden.⁴ Aus diesem Grund werden in vielen PaaS-Angeboten sogenannte NoSQL-Datenbanken als permanenter Speicher angeboten. Diese verfolgen nicht das sogenannte ACID-Prinzip⁵, sondern arbeiten nach dem BASE-Prinzip⁶, das auf eine sofortige Konsistenz verzichtet, aber immer eine schlussendliche Konsistenz herstellt. D.h., nach einer möglichst kurzen Zeitspanne sind alle Knoten des NoSQL-Clusters wieder konsistent.

SQL- und NoSQL-Datenbanken eignen sich nur bedingt zur Speicherung von großen Datenblöcken (sogenannten BLOBS). Aus diesem Grund bieten viele PaaS-Anbieter sogenannte BLOB-Speicherdienste an. Lädt man eine Datei in einen BLOB-Speicherdienst hoch (dies geschieht immer über einen Funktionsaufruf aus der entsprechenden API), so erhält man als Rückgabewert einen BLOB-Schlüssel, über den die Datei später wieder heruntergeladen werden kann. Häufig wird dieser BLOB-Schlüssel dann in der zur Anwendung gehörenden SQL- oder No-SQL-Datenbank gespeichert. Einer der bekanntesten BLOB-Speicherdienste ist Amazon S3 (Simple Storage Service).

4. Die Ursache dieses Phänomens liegt in dem sogenannten CAP-Theorem begründet. CAP ist eine Abkürzung der englischen Begriffe für Konsistenz, Verfügbarkeit und Partitionstoleranz. Das CAP-Theorem besagt, dass es für ein verteilt rechnendes System unmöglich ist, gleichzeitig die drei Eigenschaften Konsistenz, Verfügbarkeit und Partitionstoleranz zu erfüllen.

Konsistenz (engl. Consistency) bedeutet hier: Alle Knoten sehen zur selben Zeit alle Daten.

Verfügbarkeit (engl. Availability) bedeutet hier: Alle Anfragen an das System werden stets beantwortet.

Partitionstoleranz (engl. Partition tolerance) bedeutet hier: Das System arbeitet trotz willkürlicher Verluste von Nachrichten weiter.

5. ACID steht für Atomarität (engl. Atomicity), Konsistenz (engl. Consistency), Isolation und Dauerhaftigkeit (engl. Durability) und beschreibt gewünschte Eigenschaften von Verarbeitungsschritten in Datenbanksystemen.

6. Ist ein Prinzip, nach dem Datenbankänderungen nicht sofort, sondern nachgelagert repliziert werden, was in einer nachgelagerten, aber trotzdem garantierten Konsistenz mündet.

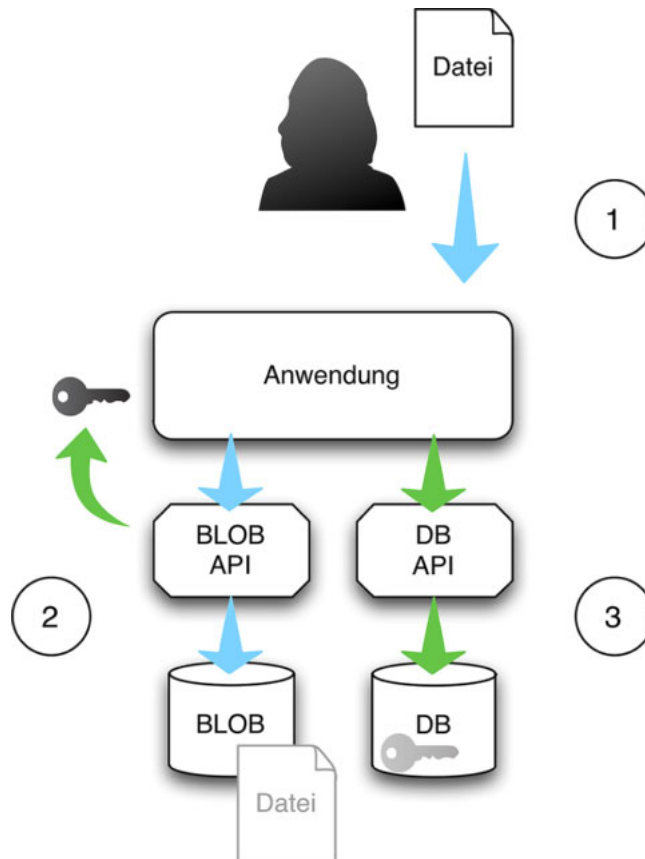


Abb. 1.4: Funktionsweise BLOB-Speicher

Zur Beschleunigung von Datenzugriffen wird von vielen PaaS-Anbietern zusätzlich zu den permanenten Speicherdiensten ein MemCached-Dienst angeboten. Dieser Dienst wird durch Cache-Server bereitgestellt, die Datenbankinhalte und BLOBS im Hauptspeicher der Server vorhalten. Dabei wird der gesamte verfügbare Hauptspeicher der Server verwendet. Ist der Hauptspeicher des Servers voll, werden die im Cache enthaltenen Daten nach und nach gelöscht. Dabei werden stets die Daten zuerst gelöscht, auf die am wenigsten zugegriffen worden ist. Durch Vorhalten der Daten im Hauptspeicher der Server werden langsame Festplattenzugriffe vermieden und dadurch die gesamte Anwendung stark beschleunigt.

Die drei Teile einer Anwendung (grafische Benutzeroberfläche, Anwendung und der permanente Datenspeicher) können von verschiedenen Entwicklern erstellt werden. Nichtsdestotrotz werden die Speicherung der Daten und die Datenbearbeitung in der Regel von denselben Entwicklern bearbeitet. Die grafische Benutzeroberfläche wird hingegen häufig von entsprechenden UX⁷-Experten gestaltet.

7. UX ist eine Abkürzung für den englischen Begriff User Experience, der mit Nutzererfahrung oder Nutzererlebnis übersetzt wird.

1.5 Verfügbarkeit von Anwendungen auf PaaS

Um die Verfügbarkeit einer Anwendung, die auf einem PaaS-Angebot ausgeführt wird, zu beeinflussen, müssen alle drei Teile der Anwendung (grafische Benutzeroberfläche, Anwendung und der permanente Datenspeicher) berücksichtigt werden.

Die **Verfügbarkeit eines Systems** wird anhand der Zeit, die das System nicht verfügbar ist, und der Gesamtzeit berechnet:

$$\text{Verfügbarkeit} = \frac{(\text{Gesamtzeit} - \text{Gesamtausfallzeit})}{\text{Gesamtzeit}}$$

Der Maximalwert der Verfügbarkeit ist 1 (entspricht 100 %) und das Minimum ist 0 – wenn die Gesamtausfallzeit der Gesamtzeit entspricht. Stellt man die obige Gleichung um, so erhält man mit $\text{Gesamtzeit} \times \text{Verfügbarkeit}$ eine kurze Formel, mit deren Hilfe man die Zeit ausrechnen kann, die das System mindestens verfügbar sein soll. Dies ist aus Kundensicht interessant.

Mit einer weiteren einfachen Umstellung der Gleichung erhält man schließlich die (für den Anbieter wichtige) vom Kunden tolerierte Gesamtausfallzeit:

$$\text{Gesamtausfallzeit} = \text{Gesamtzeit} - \text{Gesamtzeit} \times \text{Verfügbarkeit}$$

Die Gesamtausfallzeit stellt für den Anbieter eines Dienstes die Zeit dar, die er zur Verfügung hat, um bei einem Ausfall den Dienst wieder zur Verfügung zu stellen. Als Beispiel berechnen wir einmal die Gesamtausfallzeit bei einer Verfügbarkeit von 99 % über ein Jahr (365 Tage):

$$\text{Gesamtausfallzeit} = 365t - 365t \times 0,99 = 365t - 361,35t = 3,65t$$

Die Darstellung einer webbasierten Anwendung wird von Webservern bereitgestellt, die die darzustellenden Daten von Web-Services abrufen und zusammen mit HTML-Dateien und verschiedenen Medien an den Webbrowser des Nutzers ausliefern. Damit dieser Vorgang funktioniert, müssen sowohl die Webserver, die Web-Services (werden auf Applikationsservern ausgeführt) als auch die zur Speicherung der Daten verwendeten Systeme verfügbar sein. Somit ergibt sich die Gesamtverfügbarkeit der Anwendung aus der Verfügbarkeit der drei beteiligten Serverarten (Webserver, Applikations-Server und Speicherdienst). Die Verfügbarkeit von Serversystemen kann generell durch Redundanz erhöht werden.

Dazu gibt es generell zwei Ansätze:

- Lastverteilung
- Clusterbildung

Eine Lastverteilung wird durch das Vorschalten eines sogenannten Lastverteilers erreicht. Der Lastverteiler nimmt die Anfragen der Nutzer entgegen und verteilt diese nach bestimmten Algorithmen auf die angeschlossenen Serversysteme.



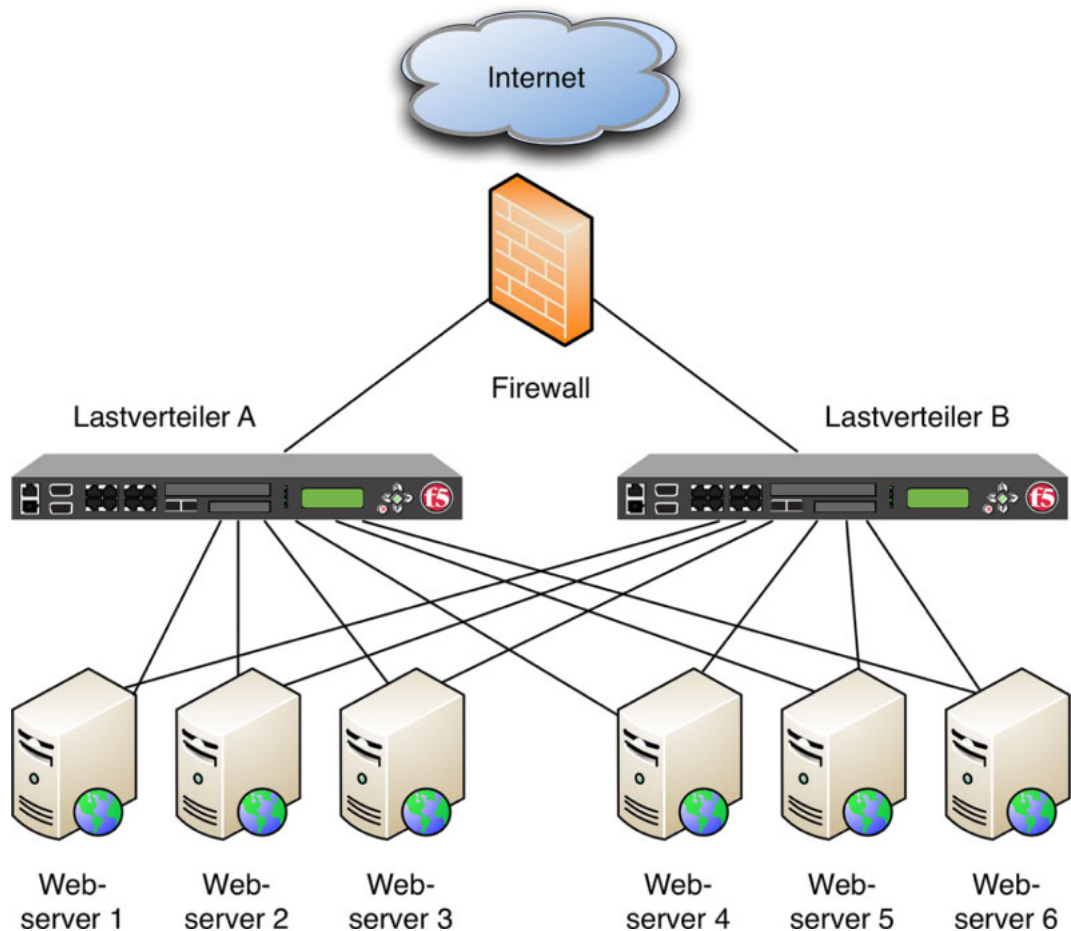


Abb. 1.5: Lastverteilung für Webserver

Die Clusterbildung wird für Serversysteme verwendet, die sich die Zustände der offenen Berechnungen merken müssen. Bei einem Ausfall eines der geclusterten Serversysteme können die verbleibenden Serversysteme die Berechnungen des ausgefallenen Systems übernehmen, da die Zustände aller offenen Berechnungen ständig zwischen den Systemen synchronisiert werden. Da die Synchronisierung der offenen Berechnungen zusätzliche Systemressourcen erfordert und dadurch keine beliebige Skalierbarkeit erreicht werden kann, versuchen die meisten PaaS-Anbieter solche Technologien einzusetzen, die eine einfache Lastverteilung erlauben.

1.6 Kompatibilität

Ein sehr wichtiger Aspekt, den es vor der Entwicklung einer Unternehmensanwendung auf einem bestimmten PaaS-Angebot zu berücksichtigen gilt, ist die Kompatibilität der Zielplattform zu anderen PaaS-Angeboten. Denn wenn der PaaS-Anbieter eines Tages nicht mehr den Ansprüchen des Kunden genügt oder er gar seine Dienste einstellt, soll es möglichst einfach sein, die Anwendung samt Daten zu einem anderen PaaS-Anbieter oder in ein eigenes Rechenzentrum zu migrieren.

Die Kompatibilität zwischen PaaS-Angeboten muss auf mehreren Ebenen geprüft werden. Zu allererst muss eine Kompatibilität bei der verwendeten Programmiersprache der Anwendung vorhanden sein. Wenn eine Anwendung in Ruby on Rails geschrieben worden ist, wird es nicht gelingen, diese auf einem PaaS-Angebot zu starten, welches nur

Laufzeitumgebungen für .Net-Anwendungen bereitstellt. Sollten zusätzlich in der Anwendung spezifische Aufrufe von Befehlen der API des PaaS-Anbieters enthalten sein, so müssen diese entweder geändert werden oder der neue PaaS-Anbieter muss die gleiche API bereitstellen.

Weiterhin muss der vom neuen PaaS-Anbieter bereitgestellte permanente Speicher zu dem alten Speicher kompatibel sein. Hierbei gilt es auch wieder zu beachten, dass die verwendeten APIs für den Zugriff auf den Speicherdienst kompatibel sind. Andernfalls müssen die entsprechenden Aufrufe in der Anwendung selbst angepasst werden. Viele PaaS-Nutzer verwenden aufgrund hoher Kompatibilitätsanforderungen verschiedene PaaS-Anbieter für die Laufzeitumgebung und den permanenten Speicher. Dadurch wird die Abhängigkeit zu den PaaS-Anbietern etwas verringert. Denn wenn man beim Ausfall eines PaaS-Anbieters nur die Daten oder nur die Laufzeitumgebung migrieren muss, ist der Aufwand geringer, als wenn man gezwungen ist, sowohl Laufzeitumgebung als auch Daten gleichzeitig zu einem neuen PaaS-Anbieter zu migrieren.

Zusammenfassung

PaaS-Angebote stellen Laufzeitumgebungen, Speicherdienste und Entwicklungswerkzeuge bereit. Hierdurch sollen Anwendungsentwickler befähigt werden, Anwendungen zu erstellen und mit einer hohen Verfügbarkeit und Skalierbarkeit zu betreiben, ohne sich um die darunterliegende Hardware oder den darunterliegenden Software-Stack kümmern zu müssen.

Der am meisten genutzte PaaS-Typ ist Application PaaS – abgekürzt durch aPaaS.

Anwendungen, die auf PaaS-Angeboten entwickelt und betrieben werden, bestehen in der Regel aus den drei Teilen grafische Benutzeroberfläche, Anwendung (Web-Services) und dem permanenten Datenspeicher. PaaS-Anbieter bieten verschiedene Speichertechnologien zur permanenten Speicherung der Anwendungsdaten an.

Die Verfügbarkeit einer Anwendung, die auf einem PaaS-Angebot betrieben wird, ist abhängig von der Verfügbarkeit der einzelnen Dienste (Webserver, Applikationsserver und Speicherdienste). Um die Verfügbarkeit eines Dienstes zu erhöhen, können Lastverteiler oder Cluster verwendet werden.

Die Kompatibilität verschiedener PaaS-Angebote hinsichtlich der verfügbaren Laufzeitumgebungen für verschiedene Programmiersprachen und der verschiedenen Speicherdienste ist sehr wichtig für die Unabhängigkeit des Kunden zum PaaS-Anbieter.

Aufgaben zur Selbstüberprüfung

Überprüfen Sie nun bitte Ihr neu erworbenes Wissen. Lösen Sie die Aufgaben zunächst selbstständig und vergleichen Sie anschließend Ihre Lösungen mit den Angaben im Anhang.

- 1.1 Das Architekturmuster SOA sieht vor, Anwendungen in wiederverwendbare Dienste aufzuteilen – richtig oder falsch?
- 1.2 Welche Arbeiten werden dem Entwickler durch PaaS-Angebote abgenommen?
- 1.3 Wie heißt der Standard-PaaS-Typ?
- 1.4 Welche zwei Arten von grafischen Benutzeroberflächen werden häufig für Anwendungen, die auf PaaS-Angeboten betrieben werden, erstellt?
- 1.5 Mit welcher Formel lässt sich die Gesamtausfallzeit berechnen?
- 1.6 PaaS-Angebote bieten in der Regel Programmiersprachen-unabhängige Laufzeitumgebungen an – richtig oder falsch?

2 PaaS im Praxiseinsatz

In dieser Lektion werden wir das PaaS-Angebot Heroku mit einer einfachen, vorgefertigten „Ruby on Rails“-Anwendung ausprobieren. Auf diese Weise bekommen Sie ein Gefühl für die Handhabung von PaaS-Angeboten.

2.1 Heroku, GitHub und Amazon S3

Heroku (<https://heroku.com>) ist ein PaaS-Angebot, das Laufzeitumgebungen und Datenbanken für Anwendungen bereitstellt. Aufgrund der einfachen und sehr zuverlässigen Nutzung wird es von vielen bekannten Anbietern von Internetdiensten verwendet. Heroku hat sich auf die Bereitstellung von Laufzeitumgebungen für das Webframework Ruby on Rails spezialisiert, das die Programmiersprache Ruby verwendet. Ruby on Rails ist ein Framework, das 2003 von David Heinemeier Hansson entwickelt worden ist und von sehr bekannten Internetanwendungen wie Twitter und Groupon genutzt wurde.

Der Quellcode-Speicherdienst GitHub wurde ebenfalls mit Ruby on Rails entwickelt und bietet ein auf dem Versions-Verwaltungssystem GIT basierendes Quellcode-Hosting an. Neben dem für Open-Source-Projekte kostenlosen Angebot gibt es auch kostenpflichtige Angebote, die von vielen Internetanbietern verwendet werden. Der Vorteil von GitHub liegt in der einfachen Bedienung und dem einfachen Teilen von Quellcode zwischen verschiedenen Entwicklern. Neben vielen unbekannten Projekten werden auch sehr bekannte Open-Source-Projekte wie der Quellcode des offiziellen Linux-Kernels auf GitHub gehostet.

Da die meisten Anbieter von PaaS-Laufzeitumgebungen aus Sicherheitsgründen keinen Schreibzugriff auf das darunterliegende Dateisystem erlauben, werden zur Speicherung von Dateien häufig zusätzliche PaaS-Speicherdienste verwendet. Das bekannteste Beispiel hierfür ist Amazon S3 (Simple Storage Service). Amazon S3 erlaubt es, über eine API Dateien in Buckets zu speichern. Der Inhalt dieser Buckets kann dann wiederum über die dazugehörigen Schlüssel abgerufen werden. Das haben Sie im Abschnitt „Architektur von Anwendungen auf PaaS“ kennengelernt.

Da dieser Dienst zwar am Anfang kostenlos ausprobiert werden kann, aber nach dem Einrichten der Testanwendung sehr schnell ungewollt hohe Kosten verursachen kann, verzichten wir in diesem Heft auf die praktische Anbindung des Dienstes.

2.2 Nutzerkonten anlegen

Beginnen wir mit dem Anlegen von Nutzerkonten bei den zwei für unser Vorhaben benötigten PaaS-Angeboten. Zuerst erstellen Sie ein Nutzerkonto für heroku.com, dem PaaS-Angebot, das unseren „Ruby on Rails“-Quellcode ausführen soll.

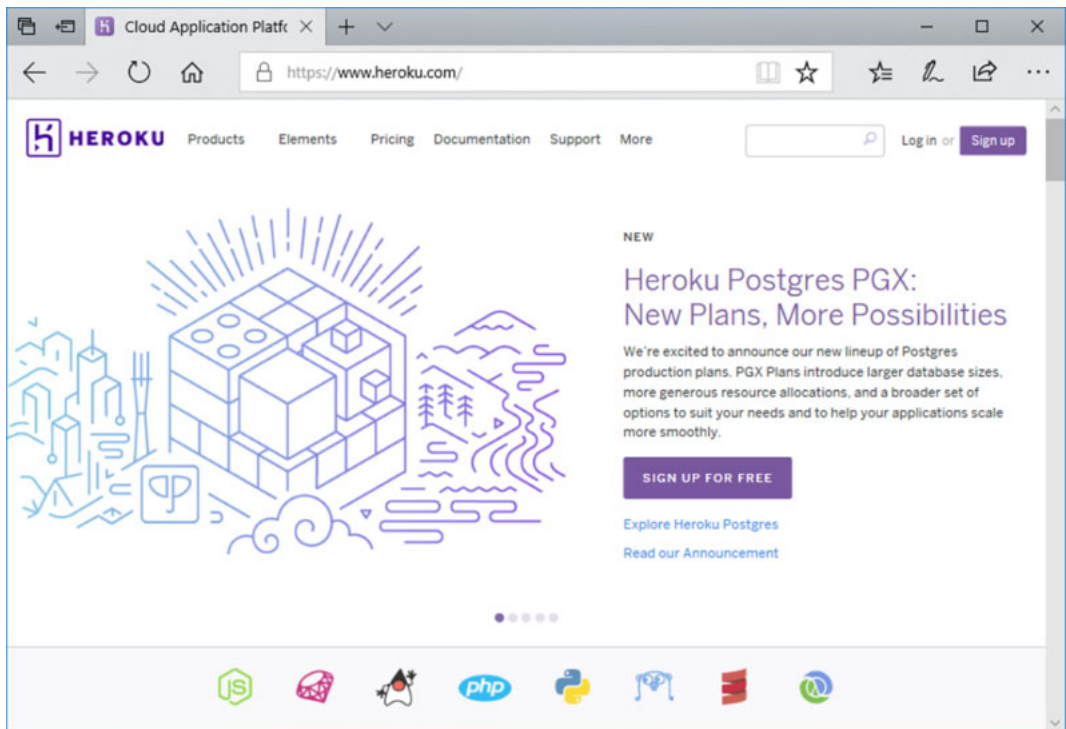


Abb. 2.1: Heroku-Anmeldung – Schritt 1

Öffnen Sie hierzu bitte die Webseite <https://heroku.com> und klicken Sie auf die Schaltfläche **SIGN UP FOR FREE**.

Heroku | Sign up

https://signup.heroku.com/?c=7013A000001yL5XQAU

HEROKU

Already have an account? Log in

Sign up for free and experience Heroku today

Free account

Create apps, connect databases and add-on services, and collaborate on your apps, for free.

Your app platform

A platform for apps, with app management & instant scaling, for development and production.

Deploy now

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name *

Last name *

Email Address *

Company name

Role *

Country *

Abb. 2.2: Heroku-Anmeldung – Schritt 2

Geben Sie bitte Ihre Daten in die mit einem roten Sternchen gekennzeichneten Pflichtfelder ein. Markieren Sie dann unten auf der Seite die Checkbox vor **Ich bin kein Roboter** und klicken Sie anschließend auf die Schaltfläche **CREATE FREE ACCOUNT**.

Hinweis:

Unter Umständen müssen Sie auch noch eine kleine Aufgabe lösen, um nachzuweisen, dass das Konto nicht maschinell angelegt wurde.

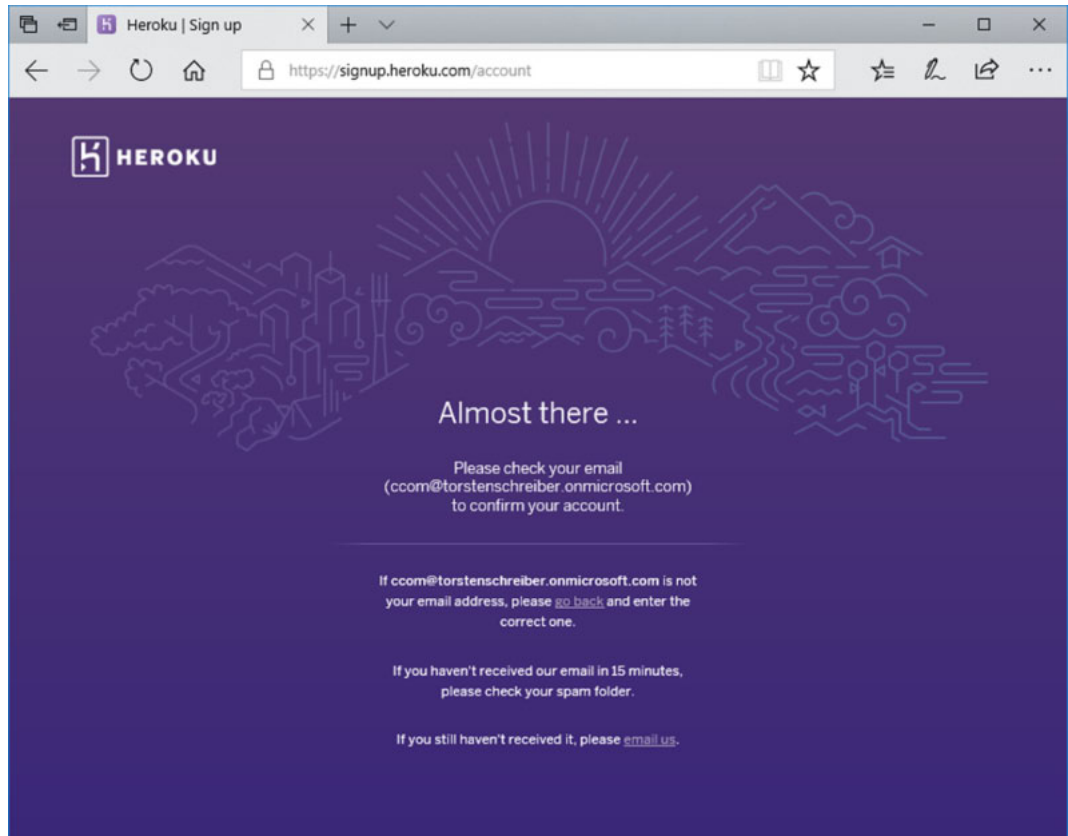


Abb. 2.3: Heroku-Anmeldung – Schritt 3

Sie erhalten nun die Nachricht, dass eine Bestätigungs-E-Mail an Sie versandt worden ist. Sollten Sie die E-Mail nicht erhalten, so überprüfen Sie bitte den Spam-Ordner Ihres E-Mail-Postfachs. Die E-Mail wird Ihnen von einer E-Mail-Adresse der Domäne heroku.com gesendet.

Der erste Hyperlink in der E-Mail mit dem Betreff „Confirm your account on Heroku“ stellt den nächsten Schritt der Anmeldung dar. Bitte klicken Sie nun auf diesen Link.

Abb. 2.4: Heroku-Anmeldung – Schritt 4

Geben Sie nun Ihr gewünschtes Kennwort⁸ in das Feld **Password** ein und bestätigen Sie es im Feld **Password Confirmation**. Klicken Sie dann auf **SET PASSWORD AND LOG IN**.

Abb. 2.5: Heroku-Willkommensseite

Klicken Sie auf der Heroku-Willkommensseite auf den Link **CLICK HERE TO PROCEED**

8. Das Kennwort muss mindestens acht Zeichen lang sein und aus Buchstaben, Ziffern und/oder Sonderzeichen bestehen.

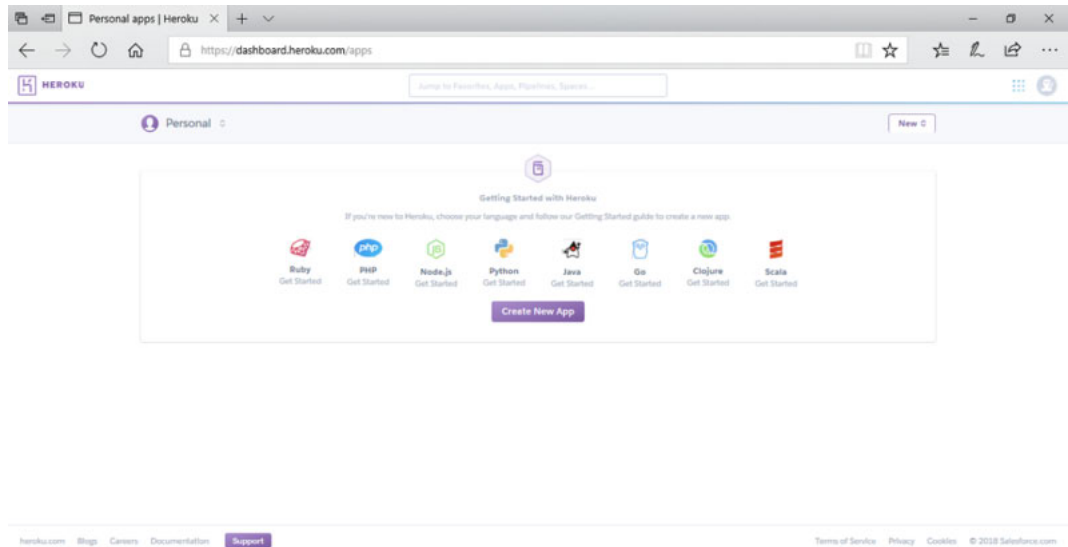


Abb. 2.6: Heroku Personal apps

Es erscheint die Seite „Personal apps“, auf der Sie später Ihre auf Heroku laufenden Anwendungen sehen können.

Schauen wir uns jetzt an, wie Sie ein Nutzerkonto für das Quellcode-Versionierungssystem GitHub.com erstellen.

Öffnen Sie hierzu bitte die Webseite <https://github.com>.

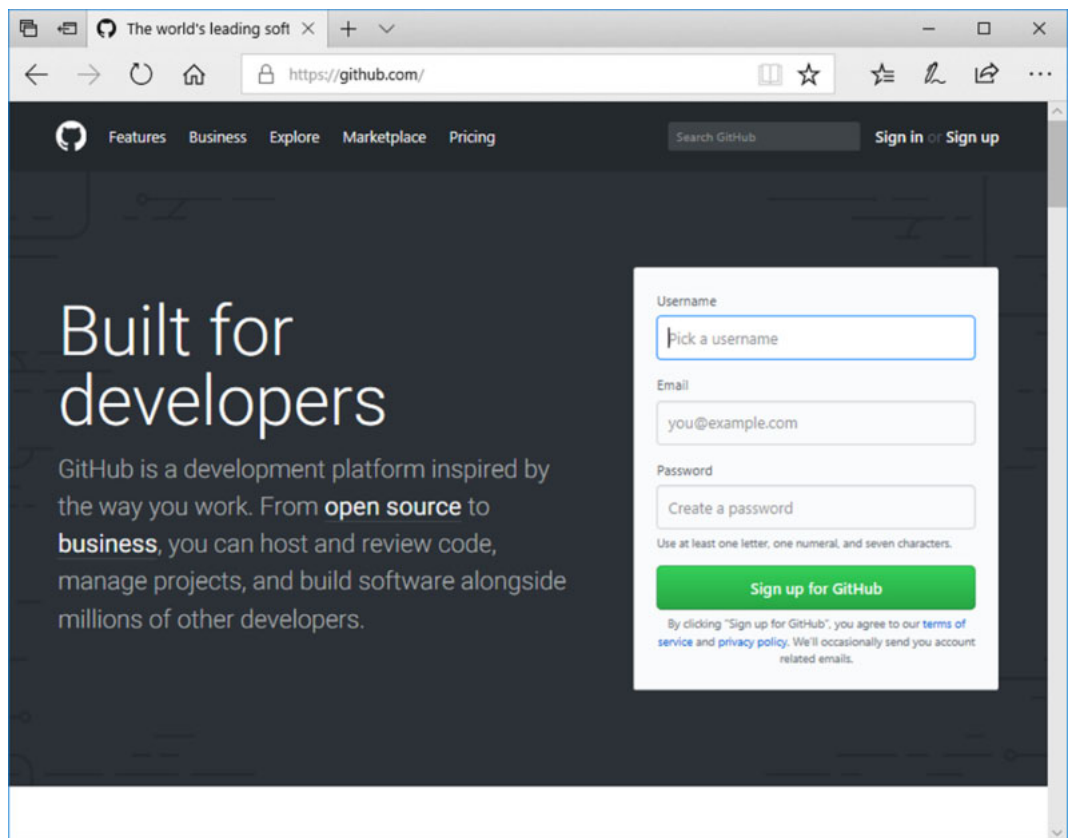


Abb. 2.7: GitHub-Anmeldung – Schritt 1

Geben Sie in die Felder bitte Ihren Nutzernamen, Ihre E-Mail-Adresse sowie Ihr gewünschtes Kennwort ein und klicken Sie auf die Schaltfläche **Sign up for GitHub**. Das Kennwort muss mindestens 7 Zeichen lang sein und mindestens einen Buchstaben und eine Ziffer enthalten.

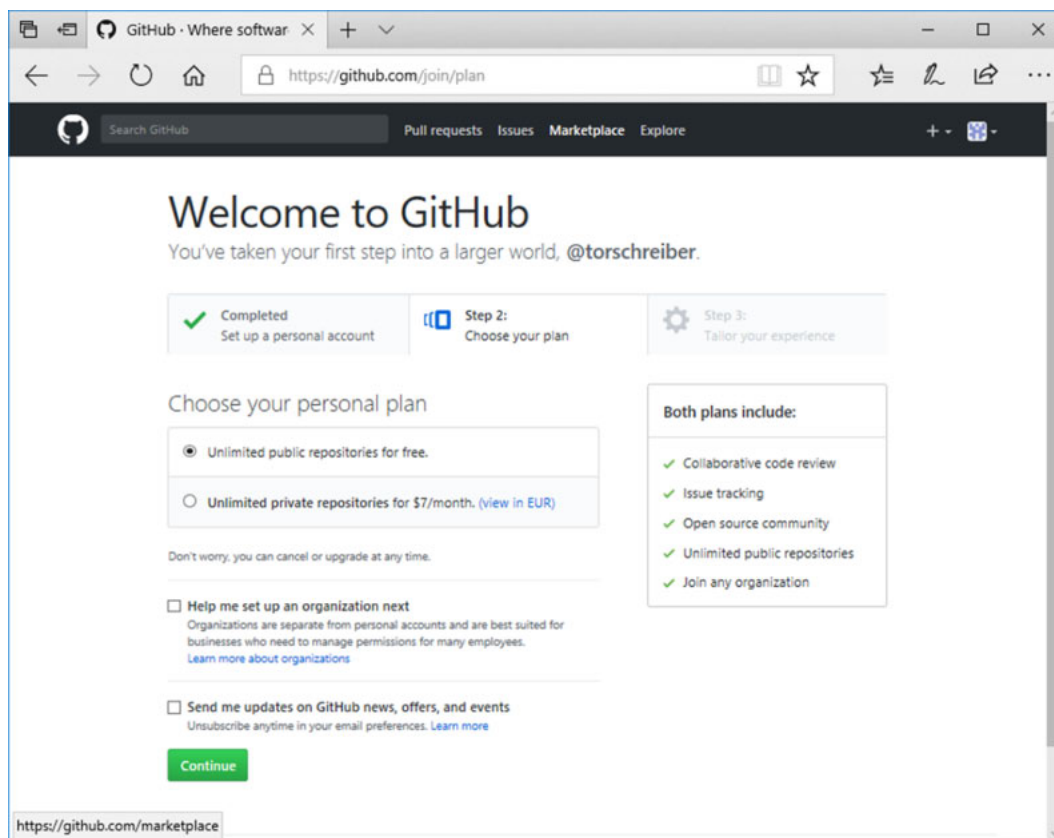


Abb. 2.8: GitHub-Anmeldung – Schritt 2

In diesem Schritt der Anmeldung legen Sie die Art beziehungsweise das Modell Ihres GitHub-Kontos fest. Für unbegrenzt öffentliche Repositories⁹ ist das Konto kostenlos. Das ist die Standardeinstellung, die Sie unverändert übernehmen können.

Klicken Sie auf **Continue**.

9. Ein Repository ist ein verwaltetes Verzeichnis von Objekten – zum Beispiel Quellcodedateien.

The screenshot shows the GitHub 'Welcome to GitHub' page in a web browser. The browser's address bar shows 'https://github.com/join/customize'. The page has a dark header with the GitHub logo, a search bar, and navigation links: 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is white and features the heading 'Welcome to GitHub' with the subtitle 'You'll find endless opportunities to learn, code, and create, @torschreiber.' Below this is a progress bar with three steps: 'Completed Set up a personal account' (with a green checkmark), 'Step 2: Choose your plan' (with a green icon), and 'Step 3: Tailor your experience' (with a blue gear icon). The form for Step 3 includes several sections: 'How would you describe your level of programming experience?' with radio buttons for 'Totally new to programming', 'Somewhat experienced', and 'Very experienced'; 'What do you plan to use GitHub for? (check all that apply)' with checkboxes for 'Research', 'School projects', 'Project Management', 'Development', 'Design', and 'Other (please specify)'; 'Which is closest to how you would describe yourself?' with radio buttons for 'I'm a hobbyist', 'I'm a student', 'I'm a professional', and 'Other (please specify)'; and 'What are you interested in?' with a text input field. At the bottom of the form, there is a small text example: 'e.g. tutorials, android, ruby, web-development, machine-learning, open-source'.

Abb. 2.9: GitHub-Anmeldung – Schritt 3

Im letzten Schritt der Anmeldung können Sie noch einige zusätzliche Angaben machen. Markieren Sie bei Bedarf die passenden Felder und klicken Sie unten auf **Submit**. Falls Sie hier keine Angaben machen möchten, klicken Sie auf den Link **skip this step**.

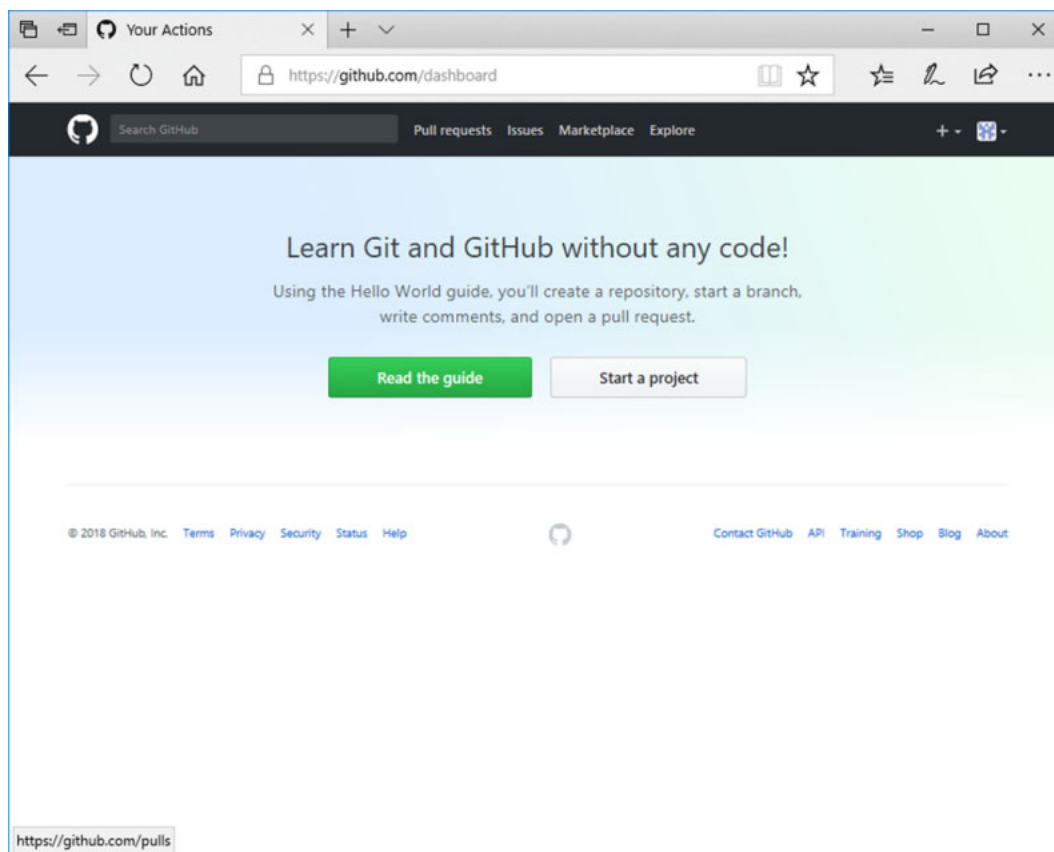


Abb. 2.10: GitHub-Anmeldung – Schritt 4

Genau wie bei Heroku ist auch bei GitHub eine Bestätigung Ihrer E-Mail-Adresse erforderlich. Ihnen wurde zu diesem Zweck eine Bestätigungsmail zugesandt. Bitte öffnen Sie diese E-Mail in Ihrem E-Mail-Programm und klicken Sie dort auf den Bestätigungslink **Verify email address**. Nun ist die Anmeldung bei GitHub abgeschlossen.

2.3 Entwicklungsumgebung einrichten

Aufgrund der Fokussierung auf die Zielgruppe der Anwendungsentwickler gehen die meisten PaaS-Anbieter davon aus, dass der Entwickler eine lokale Entwicklungsumgebung hat und den darin entwickelten Quellcode gern mit entsprechenden Werkzeugen direkt aus dieser lokalen Entwicklungsumgebung in die Laufzeitumgebung des PaaS-Angebots hochladen möchte. Deshalb benötigen wir nun eine lokale Entwicklungsumgebung, die uns dazu befähigt, Quellcode in die Laufzeitumgebung im PaaS-Angebot Heroku hochzuladen.

Um diese Entwicklungsumgebung einrichten zu können, haben Sie grundsätzlich verschiedene Möglichkeiten. Wir installieren und verwenden im Folgenden Heroku CLI¹⁰ und Git. Beginnen wir mit Heroku CLI.

10. CLI steht für Command Line Interface (dt. Kommandozeilen-Schnittstelle).

2.3.1 Heroku CLI installieren

Sie finden die Installationsprogramme für Windows in der 32-bit- und 64-bit-Version im Begleitmaterial zu diesem Heft im Ordner HerokuCLI. Weitere Versionen für andere Plattformen können Sie im Heroku Dev Center unter <https://devcenter.heroku.com/articles/heroku-cli> herunterladen.

Kopieren Sie das zu Ihrem Betriebssystem passende Heroku-CLI-Installationsprogramm aus dem Begleitmaterial auf die Festplatte Ihres Computers. Führen Sie die Datei dann aus.

Bestätigen Sie gegebenenfalls anschließend die Nachfrage der Benutzerkontensteuerung mit **Ja**.

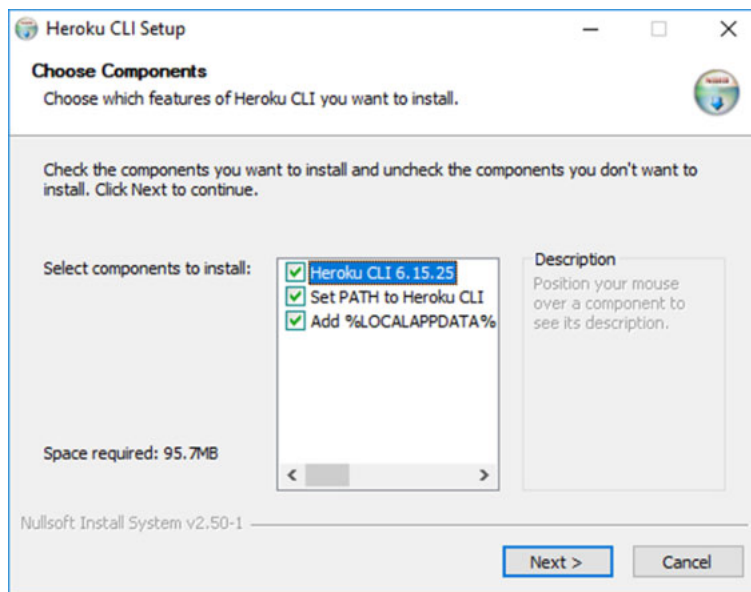


Abb. 2.11: Installation Heroku CLI – Schritt 1

Im ersten Schritt der Installation wählen Sie die zu installierenden Komponenten aus. Für unseren Zweck können wir alles unverändert lassen. Klicken Sie auf die Schaltfläche **Next >**.

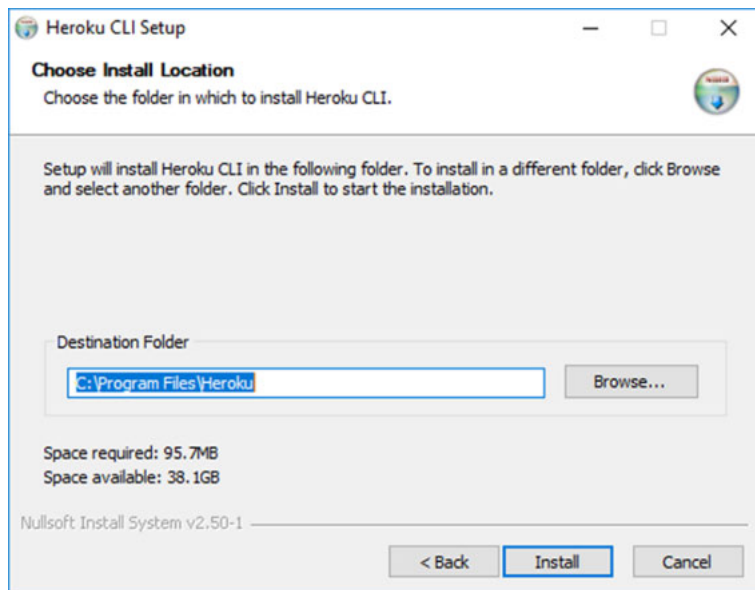


Abb. 2.12: Installation Heroku CLI – Schritt 2

In diesem Schritt müssen Sie lediglich das Installationsverzeichnis bestätigen. Klicken Sie auf die Schaltfläche **Install**, um die Installation zu starten.

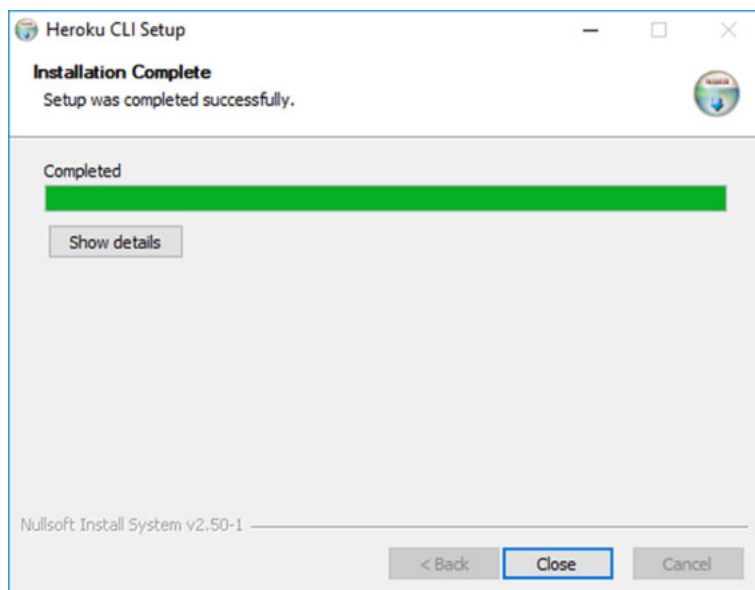


Abb. 2.13: Installation Heroku CLI abgeschlossen

Nach erfolgreicher Installation klicken Sie bitte auf die Schaltfläche **Close**.

2.3.2 Git installieren und konfigurieren

Git steht wie Heroku CLI für verschiedene Plattformen zur Verfügung. Die Installer für Windows finden Sie im Begleitmaterial dieses Studienhefts im Ordner Git. Versionen für andere Plattformen können Sie unter <https://git-scm.com/> herunterladen.

Kopieren Sie das zu Ihrem Betriebssystem passende Git-Installationsprogramm aus dem Begleitmaterial auf die Festplatte Ihres Computers. Führen Sie die Datei dann aus. Bestätigen Sie die gegebenenfalls erscheinende Sicherheitswarnung mit **Ausführen** und die Nachfrage der Benutzerkontensteuerung mit **Ja**.



Abb. 2.14: Git-Installation – Schritt 1

Im ersten Schritt des Installationsprogramms werden Ihnen Lizenzinformationen angezeigt. Lesen Sie diese durch und klicken Sie dann auf **Next >**.

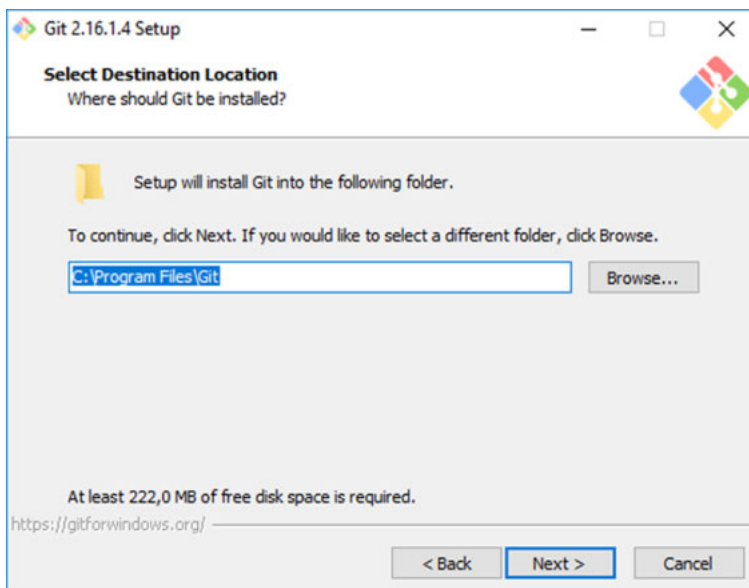


Abb. 2.15: Git-Installation – Schritt 2

Im zweiten Schritt der Installation müssen Sie das Installationsverzeichnis festlegen. In der Regel können Sie das vorgeschlagene Verzeichnis übernehmen. Klicken Sie auf **Next >**, um die Installation fortzusetzen.

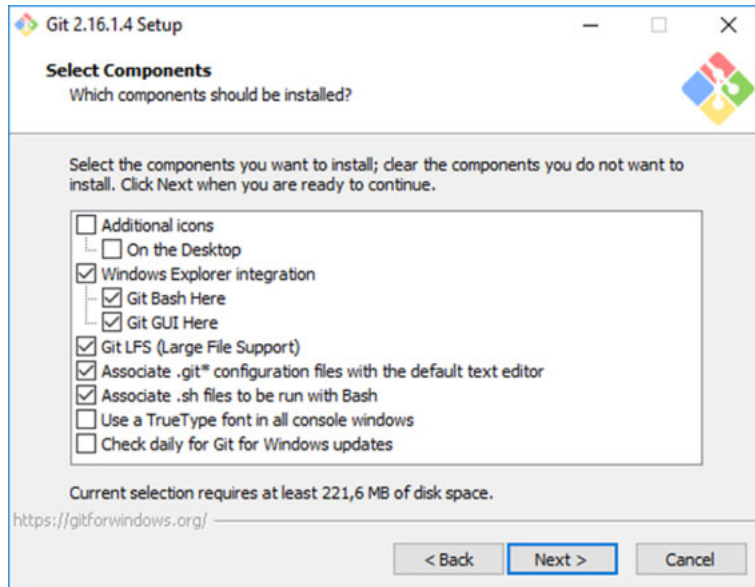


Abb. 2.16: Git-Installation – Schritt 3

Die zu installierenden Komponenten wählen Sie im dritten Schritt des Installationsprogramms aus. Auch hier müssen Sie nichts ändern. Klicken Sie auf **Next >**.

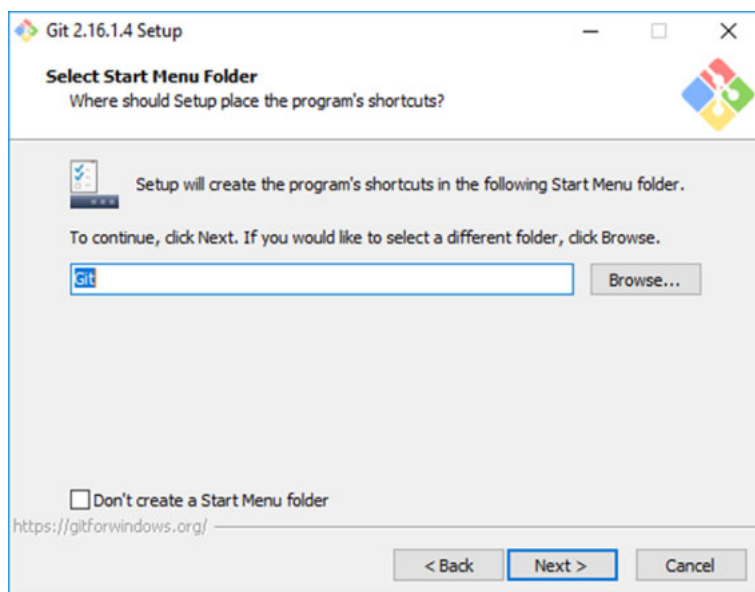


Abb. 2.17: Git-Installation – Schritt 4

Im vierten Schritt können Sie den Namen des Ordners im Startmenü angeben, in dem die Programmverknüpfungen angelegt werden. Übernehmen Sie den Vorschlagswert mit **Next >**.

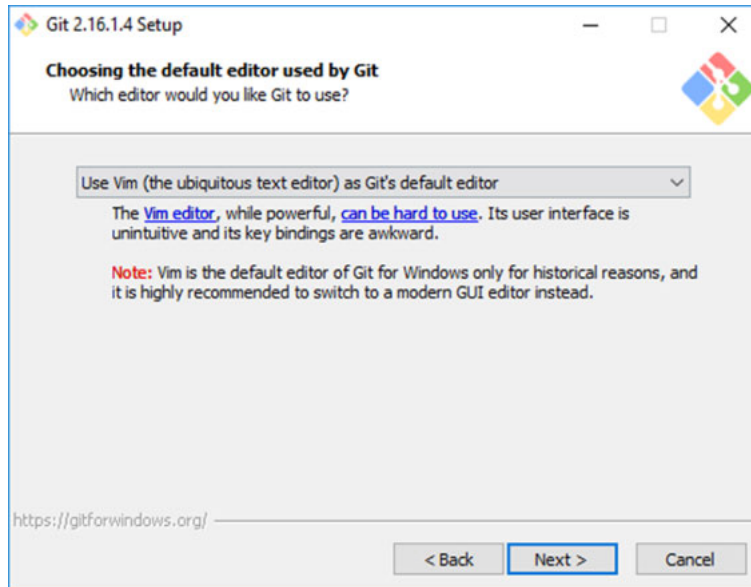


Abb. 2.18: Git-Installation – Schritt 5

Den Standardeditor können Sie in diesem Schritt auswählen. Für unsere Zwecke ist das ohne Bedeutung. Klicken Sie auf **Next >**, um den Standardeditor zu übernehmen.

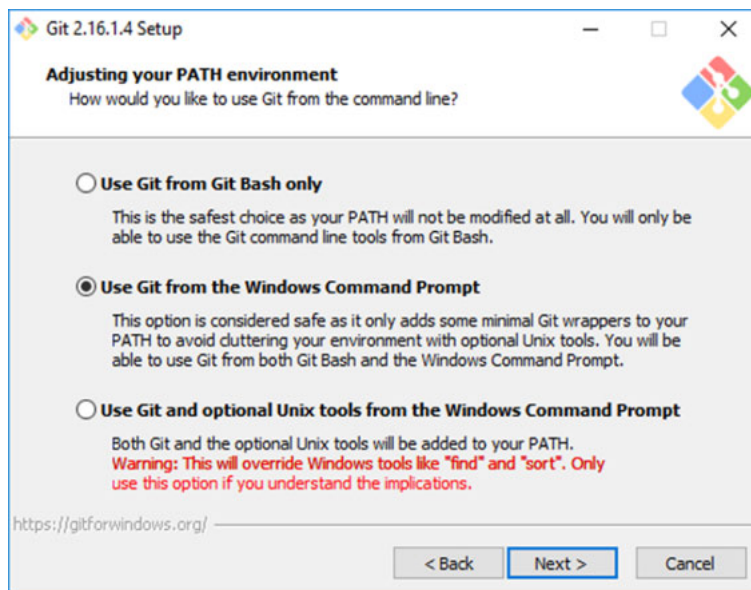


Abb. 2.19: Git-Installation – Schritt 6

Da Git ein Programm ist, das ursprünglich aus der Unix-Welt stammt, wird es über die Kommandozeile bedient, das heißt in der Windows-Eingabeaufforderung oder der Git Bash. In diesem Schritt der Installation können Sie festlegen, ob Sie Git nur über die mit der Anwendung zur Verfügung gestellten Git Bash – einer speziellen Konsole – oder auch über die Windows-Kommandozeile benutzen möchten.

Übernehmen Sie auch hier die Vorschlagsoption **Use Git from the Windows Command Prompt**. Klicken Sie auf **Next >**.

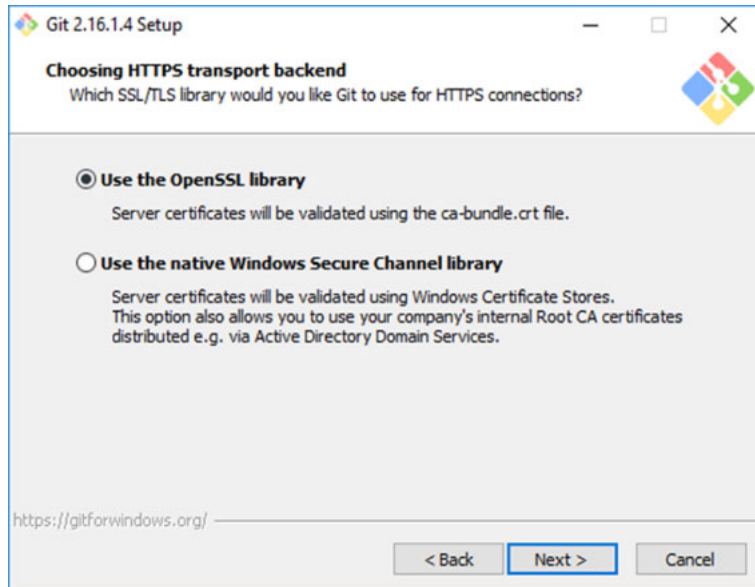


Abb. 2.20: Git-Installation – Schritt 7

Übernehmen Sie den Vorschlagswert **Use the OpenSSL library** für sichere Verbindungen und klicken Sie auf **Next >**.

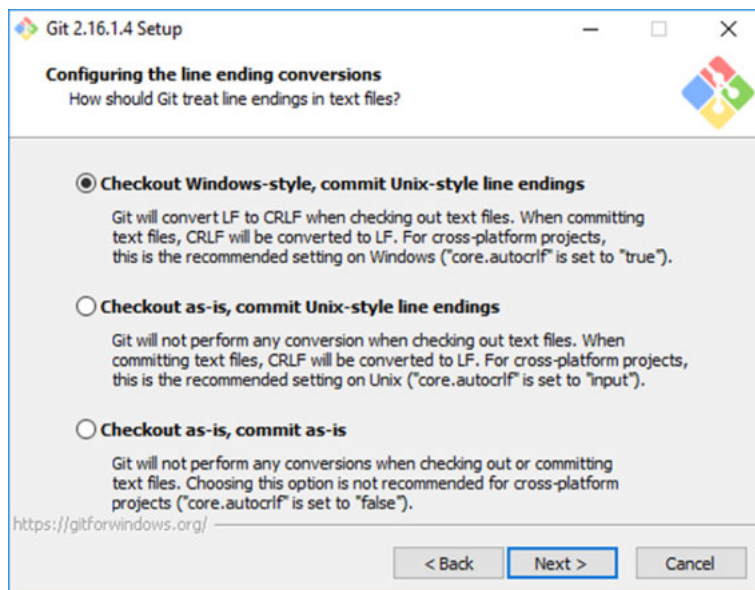


Abb. 2.21: Git-Installation – Schritt 8

Auch im achten Schritt müssen Sie nichts verändern. Klicken Sie auf **Next >**.

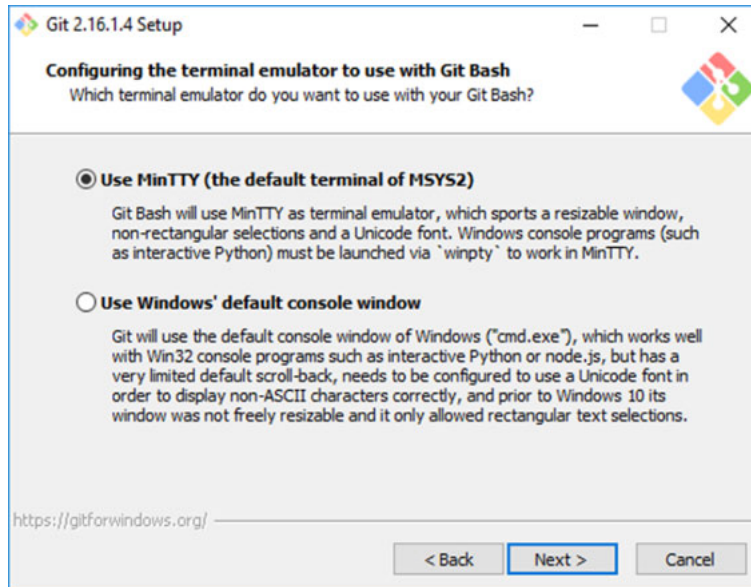


Abb. 2.22: Git-Installation – Schritt 9

Hier können Sie festlegen, welchen Terminal Emulator Sie verwenden möchten. Übernehmen Sie die Standardoption mit **Next >**.

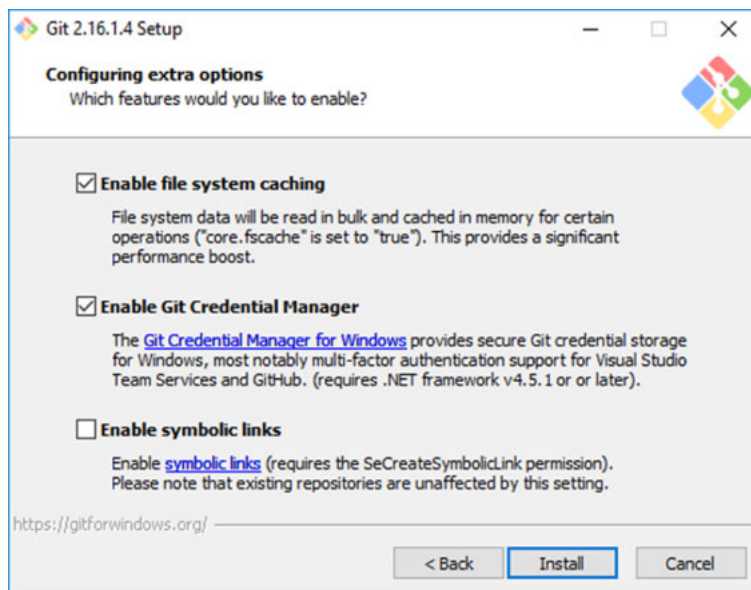


Abb. 2.23: Git-Installation – Schritt 10

Im letzten Schritt vor der eigentlichen Installation können Sie nun einige zusätzliche Programmfunktionen ein- beziehungsweise ausschalten. Wir lassen die Vorschlagswerte unverändert. Klicken Sie auf **Install**.

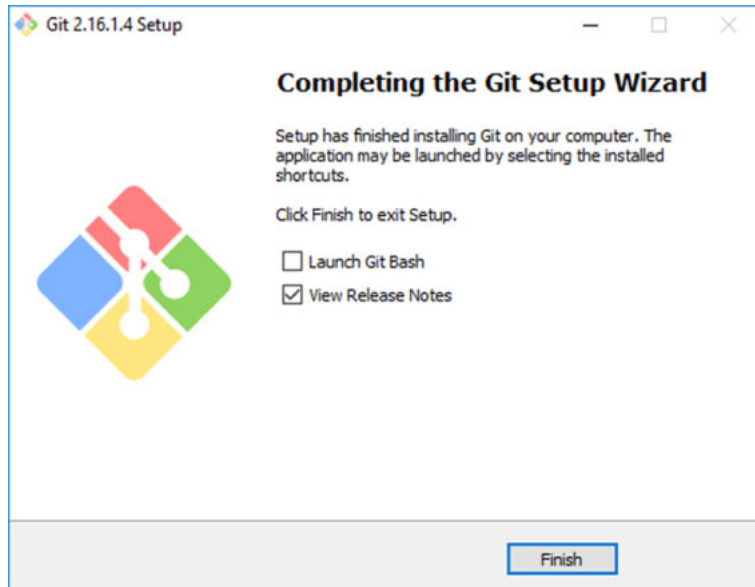


Abb. 2.24: Git-Installation abgeschlossen

Nach Abschluss der Installation erhalten Sie einen entsprechenden Hinweis.

Im nächsten Schritt werden wir Git konfigurieren. Entfernen Sie die Markierung im Feld **View Release Notes** und markieren Sie das Feld **Launch Git Bash**. Klicken Sie dann auf die Schaltfläche **Finish**.

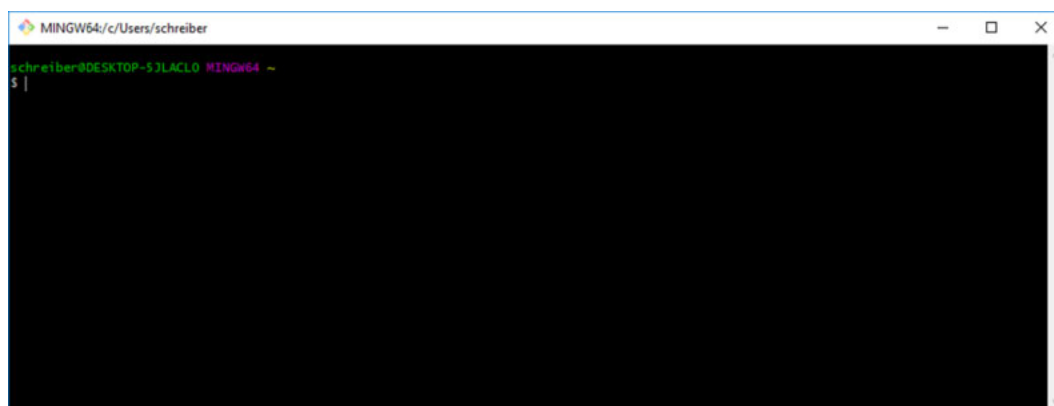



Abb. 2.25: Die Git Bash

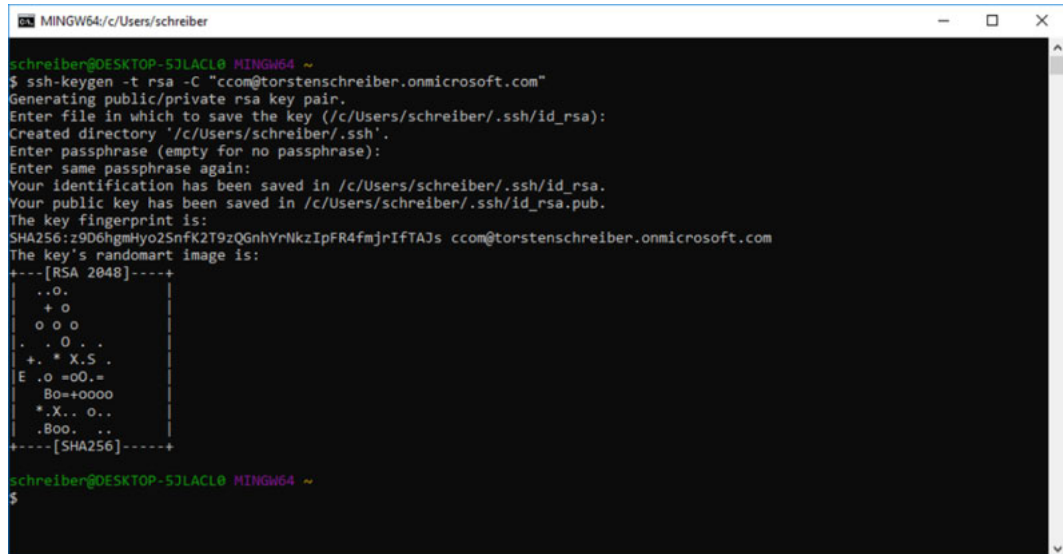
Nachdem sich die Git Bash geöffnet hat, werden wir sie nutzen, um SSH-Schlüssel zu generieren. Git verwendet SSH-Schlüssel, um die Kommunikation mit den einzelnen Repositories zu verschlüsseln.

Um neue SSH-Schlüssel generieren zu lassen, geben Sie bitte den folgenden Befehl in die Kommandozeile ein und drücken Sie die Eingabetaste .

```
ssh-keygen -t rsa -C "email@email.de"
```

Ersetzen Sie dabei bitte *email@email.de* durch Ihre eigene E-Mail-Adresse. Nachdem Sie den Befehl ausgeführt haben, wird das Skript nach dem Speicherort für die Schlüsseldatei fragen. Drücken Sie an dieser Stelle einfach die Eingabetaste , um die Vorauswahl zu bestätigen. Danach wird Sie das Skript zweimal (einmal zur Eingabe und einmal

zur Bestätigung) nach einer **passphrase** fragen. Geben Sie hierfür bitte ein längeres Kennwort ein, das Sie sich gut merken können. Bitte beachten Sie, dass das von Ihnen eingegebene Kennwort nicht angezeigt wird. Dieses Kennwort müssen Sie später immer eingeben, wenn Sie über Git mit einem Repository kommunizieren wollen.



```

MINGW64/c/Users/schreiber
schreiber@DESKTOP-53LACL0 MINGW64 ~
$ ssh-keygen -t rsa -C "ccom@torstensschreiber.onmicrosoft.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/schreiber/.ssh/id_rsa):
Created directory '/c/Users/schreiber/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/schreiber/.ssh/id_rsa.
Your public key has been saved in /c/Users/schreiber/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:z9D6hgmHyo2Snfk2T9zQGnhYrNkzIpFR4fmjrIFTAJs ccom@torstensschreiber.onmicrosoft.com
The key's randomart image is:
+---[RSA 2048]---+
  ..O.
   + o
  o o o
  . . O . .
 +. * X.S .
E .o =oO.=
  Bo=+oooo
  *.X.. o..
  .Boo. ..
+---[SHA256]---+
schreiber@DESKTOP-53LACL0 MINGW64 ~
$

```

Abb. 2.26: SSH-Schlüssel generiert

Nachdem die SSH-Schlüssel (bestehend aus einem privaten und einem öffentlichen Schlüssel) generiert worden sind, müssen wir den öffentlichen Schlüssel dem Quellcode-Speicherdienst GitHub.com bekannt machen. Dazu werden wir den Schlüssel im ersten Schritt aus einer Textdatei kopieren.

Starten Sie den Windows-Editor Notepad. Geben Sie dazu beispielsweise `Edit` in das **Suchfeld** in der Taskleiste ein und klicken Sie dann auf den Eintrag **Editor**. Nach dem Starten des Editors wählen Sie bitte **Datei -> Öffnen**.

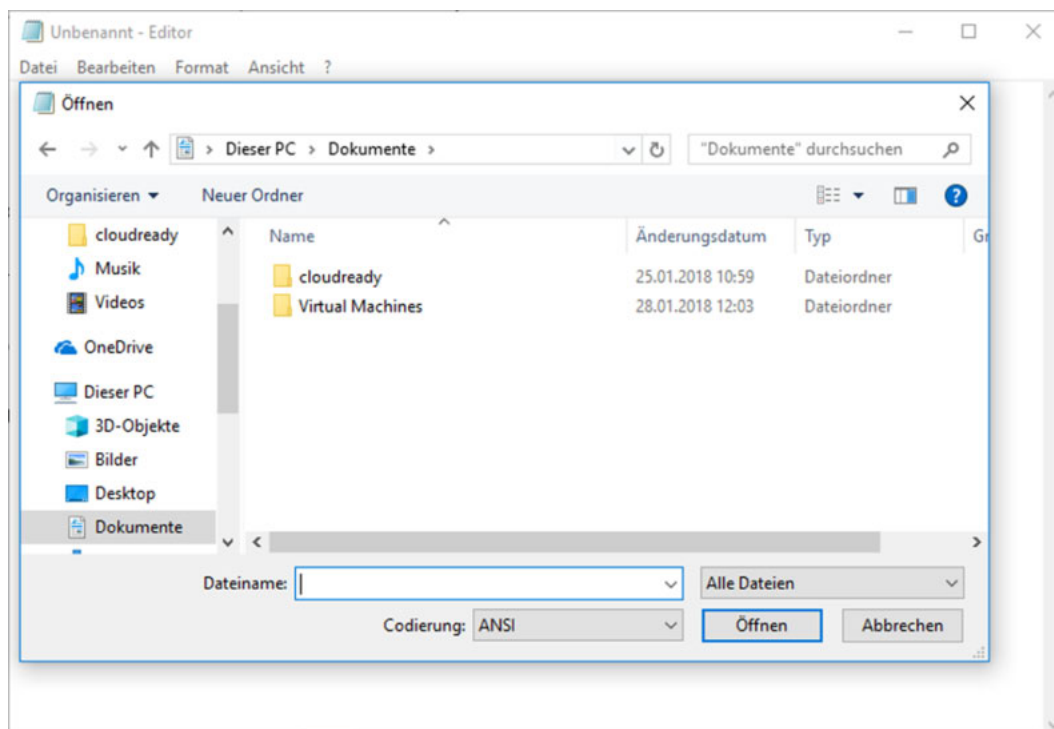


Abb. 2.27: Geöffneter Windows-Editor

Der öffentliche Schlüssel befindet sich in einer Textdatei mit dem Namen `id_rsa.pub`.

Geben Sie in das Feld **Dateiname** `%UserProfile%\.ssh\id_rsa.pub` ein und klicken Sie dann auf die Schaltfläche **Öffnen**.

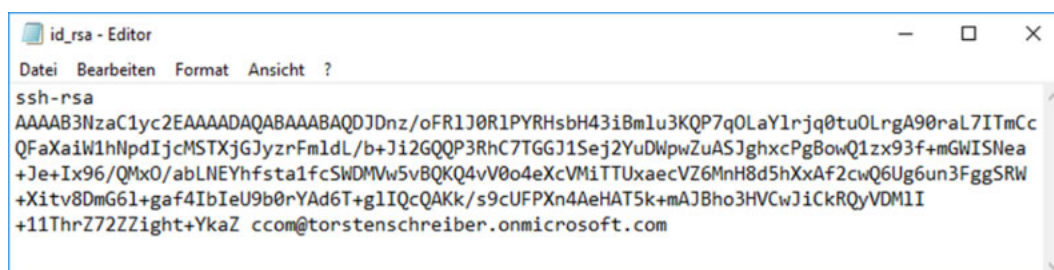


Abb. 2.28: Der öffentliche Schlüssel im Editor

Markieren Sie nun bitte den gesamten öffentlichen Schlüssel – beispielsweise mit der Tastenkombination **Strg** + **A** – und wählen Sie **Kopieren** aus dem Menü **Bearbeiten** oder aus dem Kontextmenü. Wenn Sie die GitHub-Webseite nicht mehr geöffnet haben, öffnen Sie bitte nun die Webseite <https://github.com> und loggen Sie sich mit Ihrem vorher erstellten Nutzerkonto ein.

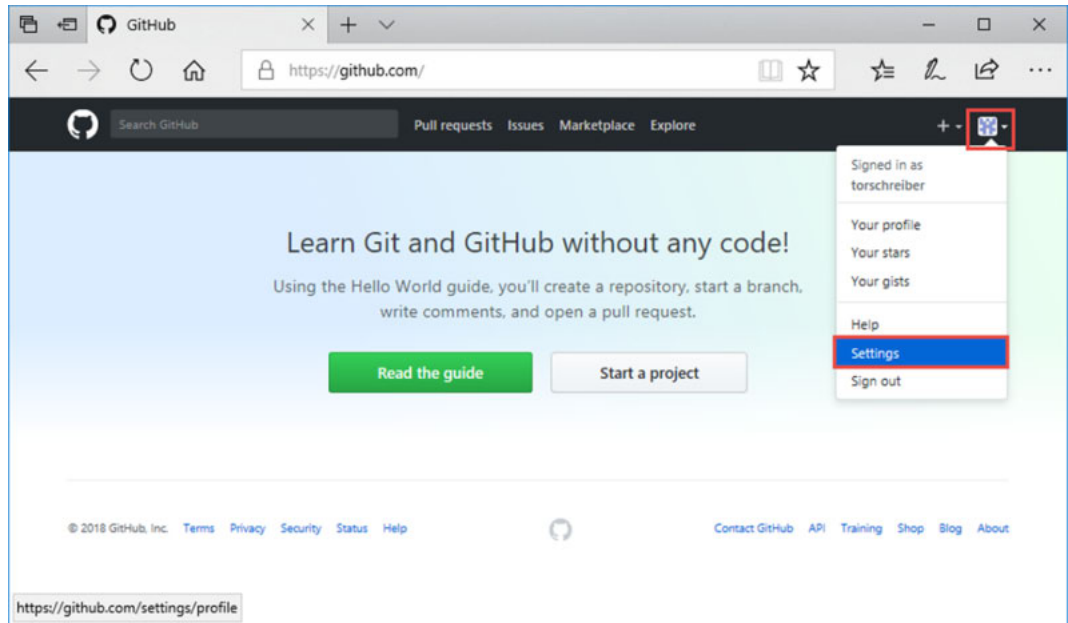


Abb. 2.29: GitHub Settings

Auf der GitHub-Webseite öffnen Sie dann bitte oben rechts das Menü und klicken dort auf **Settings**.

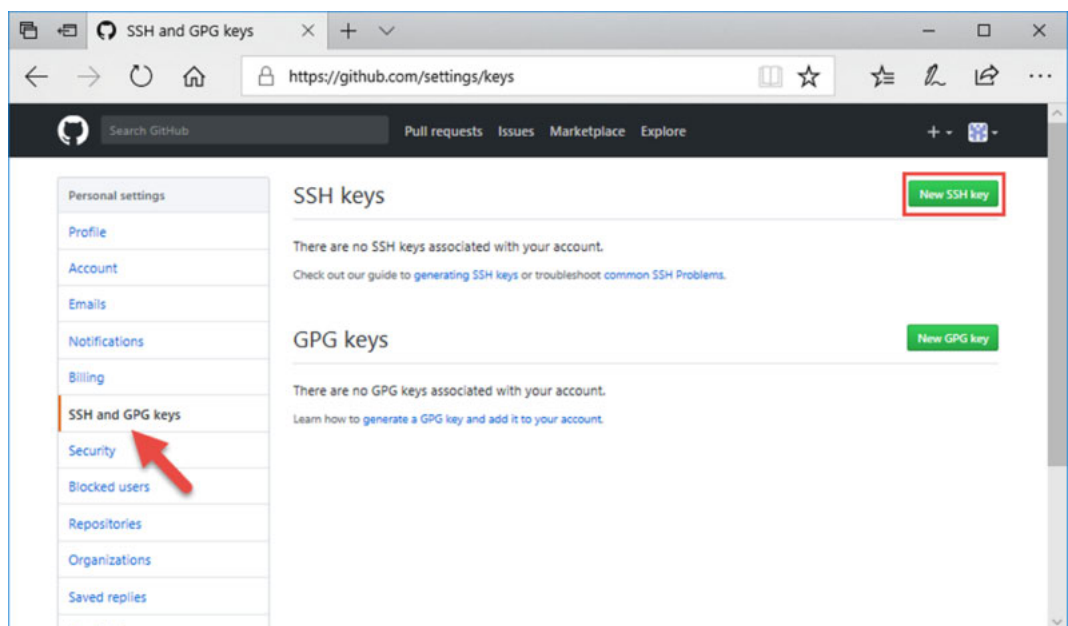


Abb. 2.30: Neuen SSH Key einrichten

Links im Bereich **Personal settings** klicken Sie dann auf den Punkt **SSH and GPG keys**. Klicken Sie dann oben rechts auf **New SSH key**.

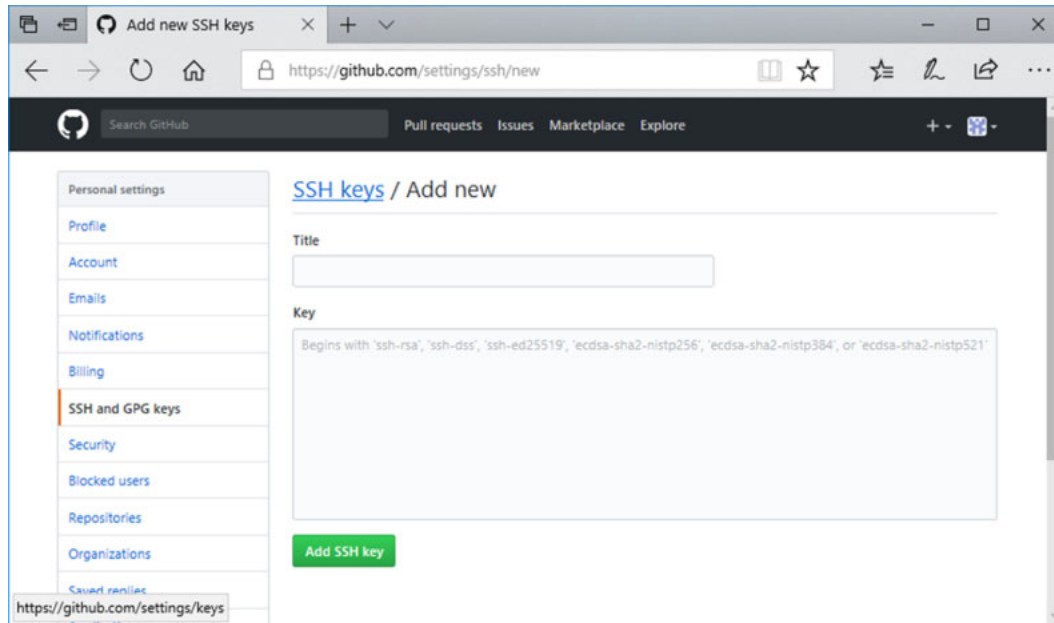


Abb. 2.31: SSH Key hinzufügen

Geben Sie in das Feld **Titel** einen Namen ein und fügen Sie den zuvor im Notepad kopierten öffentlichen Schlüssel in das Feld **Key** ein. Klicken Sie dann auf **Add SSH key**.

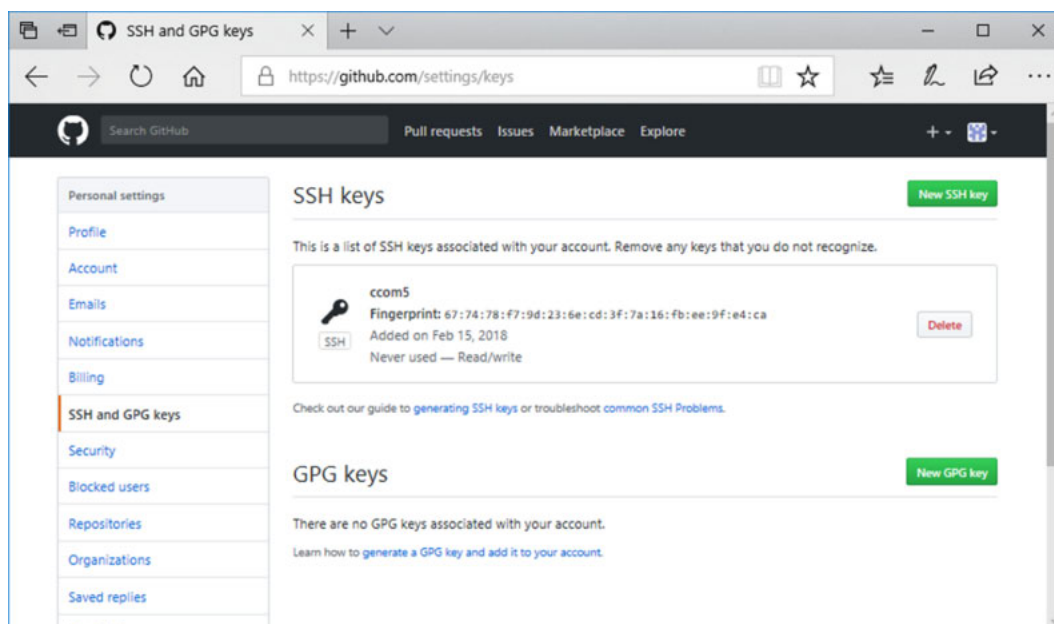


Abb. 2.32: SSH Key hinzugefügt

Nachdem wir die Kommunikation mit den GitHub Servern sichergestellt haben, müssen wir nun noch die Kommunikation mit den Heroku Servern herstellen. Hierzu rufen Sie bitte die heroku.com-Webseite auf und melden sich an.

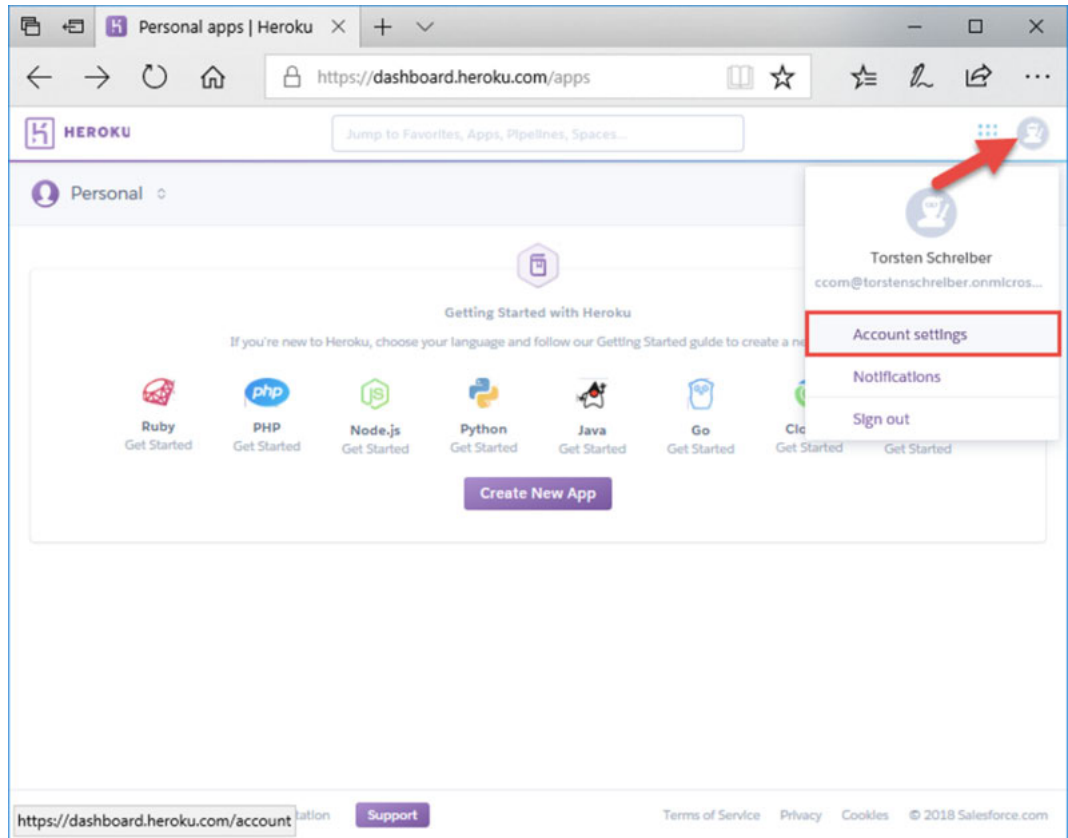


Abb. 2.33: Heroku-Kontoeinstellungen öffnen

Öffnen Sie anschließend ganz rechts oben das **Menü** und klicken dort auf **Account settings**.

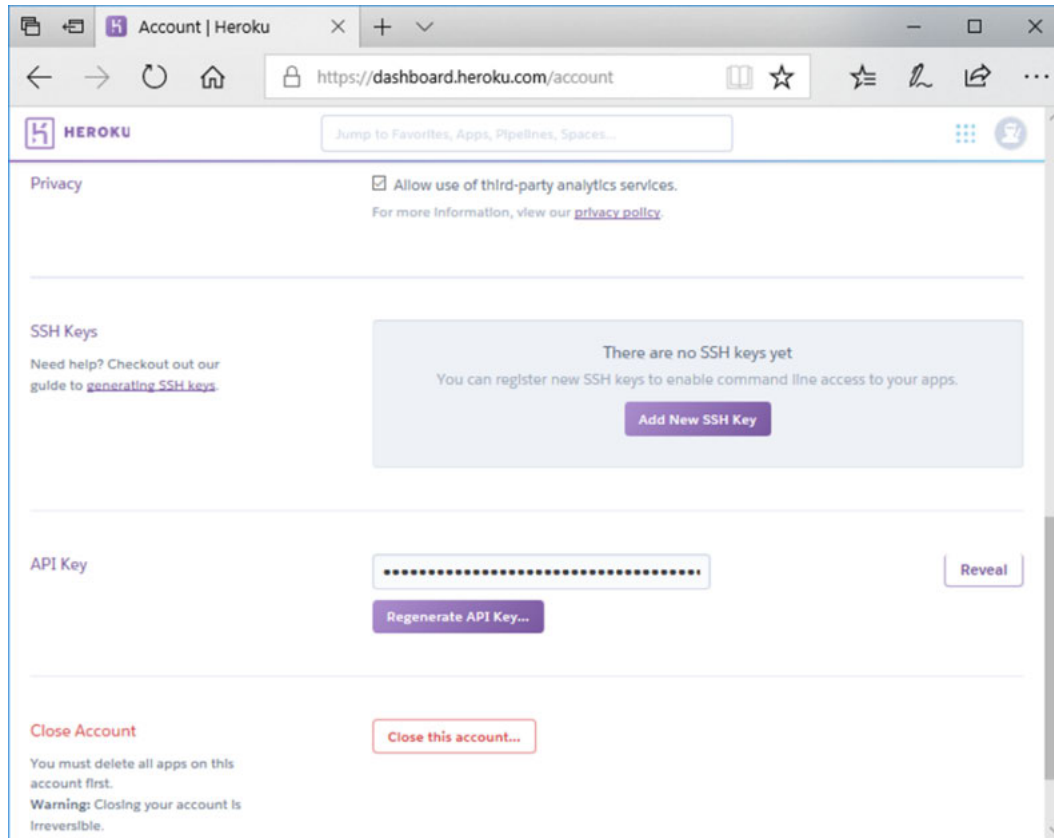


Abb. 2.34: SSH Key hinzufügen

Auf der Heroku-Account-Seite scrollen Sie bitte zum Punkt „SSH Keys“ und klicken rechts daneben auf die Schaltfläche **Add New SSH Key**.

The screenshot shows a modal window titled 'New SSH Key'. It contains a text input field labeled 'SSH Key' with a placeholder 'Your SSH Key'. Below the input field is a red 'Required' label. At the bottom of the modal is a purple button labeled 'Save changes'.

Abb. 2.35: SSH Key speichern

Jetzt fügen Sie denselben Public-Key in das Feld **SSH Key** ein, den Sie auch in GitHub verwendet haben. Klicken Sie abschließend auf die Schaltfläche **Save changes**.

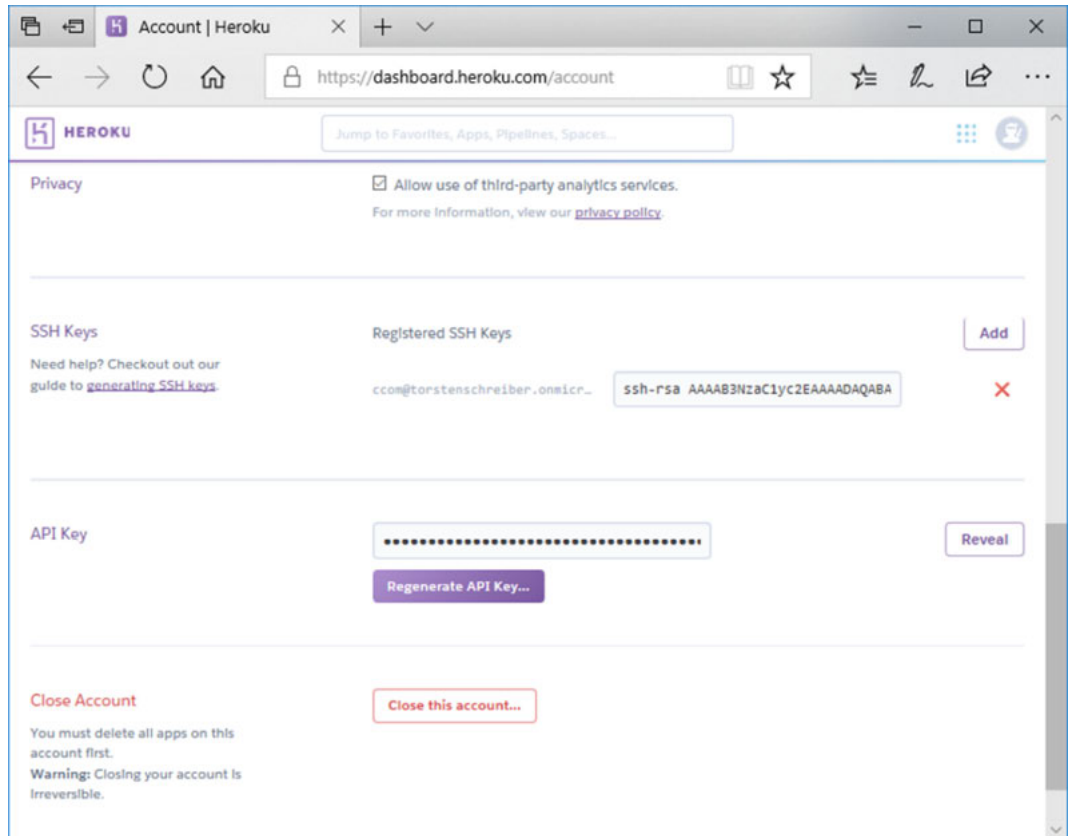


Abb. 2.36: Registrierter SSH-Key

Damit haben Sie nun die Verbindung zwischen der Entwicklungsumgebung und den Heroku Servern abgesichert und können Code in die Laufzeitumgebung hochladen.

Nun werden wir mit der Konfiguration von Git fortfahren und weitere Informationen eingeben, die es GitHub.com ermöglichen, die Kommunikation mit Ihrem Rechner, Ihrem Nutzerkonto zuzuordnen. Wechseln Sie dazu wieder in die Git Bash. Geben Sie dann nacheinander bitte die folgenden zwei Befehle ein:

```
git config --global user.name "Vorname Nachname"
```

```
git config --global user.email "email@email.de"
```

Ersetzen Sie dabei bitte *Vorname Nachname* und *email@email.de* mit Ihren eigenen Werten.

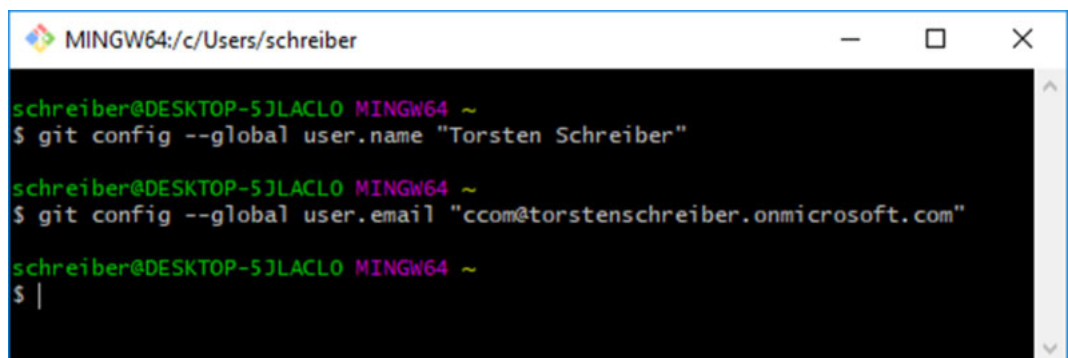


Abb. 2.37: Git-Konfiguration

Damit ist die Konfiguration der lokalen Git-Installation abgeschlossen und wir können gemeinsam im nächsten Schritt die Anwendung bereitstellen.

Schließen Sie jetzt die Git Bash. Die folgenden Schritte werden wir mit der Windows Eingabeaufforderung durchführen, da es bei den aktuellen Versionen von Heroku und Git Probleme mit der Git Bash geben kann.

Öffnen Sie die Windows-Eingabeaufforderung. Sehr schnell geht das, wenn Sie `cmd` in das **Suchfeld** in der Taskleiste eingeben und dann auf den Eintrag **Eingabeaufforderung** klicken. Alternativ öffnen Sie das **Startmenü** und wählen dort **Windows-System** -> **Eingabeaufforderung**.

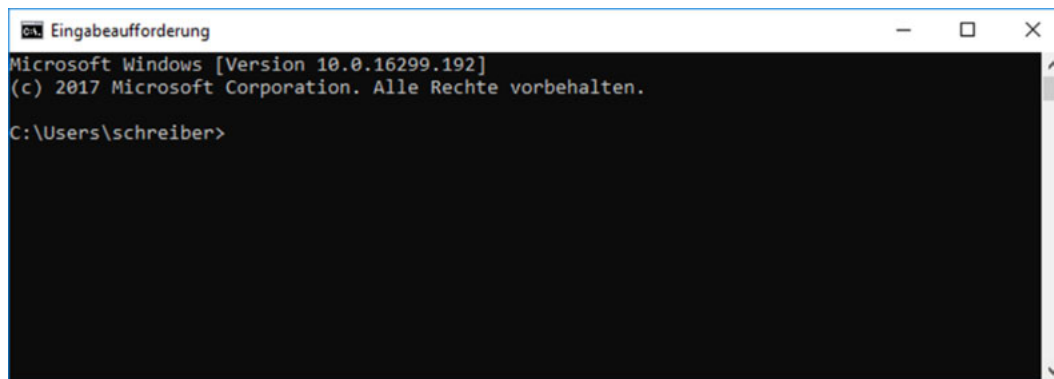


Abb. 2.38: Die Windows-Eingabeaufforderung

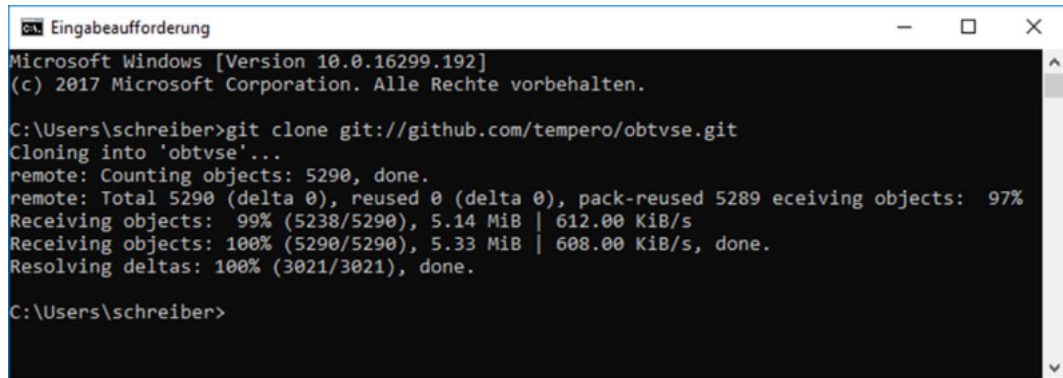
2.4 Anwendung bereitstellen

Nachdem wir unsere Entwicklungsumgebung vollständig eingerichtet haben, können wir nun unsere Testanwendung auf dem PaaS-Angebot Heroku bereitstellen. Hierzu werden wir die folgenden Schritte nacheinander ausführen.

1. Quellcode der Testanwendung herunterladen.
2. Einen Anwendungscontainer auf Heroku erzeugen.
3. Den Quellcode der Testanwendung in die Laufzeitumgebung hochladen.
4. Die Anwendungsdatenbank einrichten.
5. Die fertig bereitgestellte Anwendung ausprobieren.

Lassen Sie uns nun mit dem ersten Schritt, dem Herunterladen des Quellcodes beginnen. Geben Sie hierzu die folgende Befehlszeile gefolgt durch ein  ein:

```
git clone git://github.com/tempero/obtvse.git
```



```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\schreiber>git clone git://github.com/tempero/obtvse.git
Cloning into 'obtvse'...
remote: Counting objects: 5290, done.
remote: Total 5290 (delta 0), reused 0 (delta 0), pack-reused 5289 eceiving objects: 97%
Receiving objects: 99% (5238/5290), 5.14 MiB | 612.00 KiB/s
Receiving objects: 100% (5290/5290), 5.33 MiB | 608.00 KiB/s, done.
Resolving deltas: 100% (3021/3021), done.

C:\Users\schreiber>
```

Abb. 2.39: Herunterladen des Quellcodes

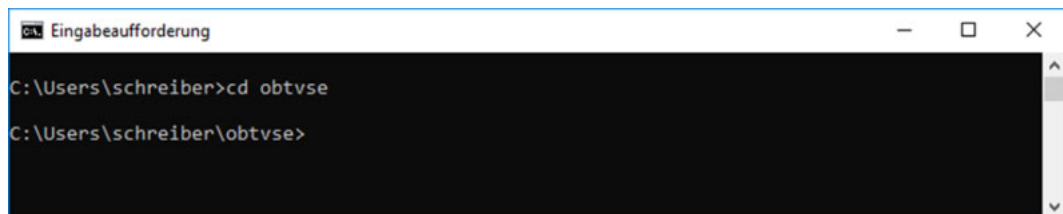
Sieht die darauffolgende Ausgabe wie die auf dem Screenshot dargestellte Ausgabe aus, so befindet sich der Quellcode der Testanwendung nun im Unterordner obtvse in Ihrem Windows-Nutzerprofil.



Anmerkung: Der Quellcode unserer Testanwendung stammt ursprünglich von `git://github.com/tempero/obtvse.git`. Um sicherzustellen, dass der Quellcode bis zu Ihrem Aufruf nicht verändert wird, wurde der Quellcode-Speicher auf einen anderen Quellcode-Speicher „geforkt“. Das heißt, es wurde eine Kopie erstellt, die nicht durch die ursprünglichen Autoren verändert werden kann.

Wechseln Sie nun bitte in das Verzeichnis obtvse mit der folgenden Befehlszeile:

```
cd obtvse
```



```
C:\Users\schreiber>cd obtvse
C:\Users\schreiber\obtvse>
```

Abb. 2.40: Verzeichnis wechseln

Damit sind wir nun bereit, um die Anwendung auf Heroku zu erzeugen und den Quellcode in die Laufzeitumgebung hochzuladen.



Anmerkung: Alle folgenden Befehle, die in der Git Bash ausgeführt werden und das Wort „heroku“ enthalten, sollten stets in dem Quellcode-Verzeichnis der Testanwendung ausgeführt werden. Dies ist für die Zuordnung der Anwendung notwendig, die mit dem entsprechenden Befehl durch die Heroku-Tools konfiguriert werden soll.

Führen Sie zum Erstellen einer leeren Laufzeitumgebung bitte den folgenden Befehl aus:

```
heroku create ccom-Namenskuerzel --stack cedar
```

Ersetzen Sie dabei bitte *Namenskuerzel* mit Ihrem eigenen Namenskürzel. Die Heroku-Tools werden Sie dann nach Ihren Anmeldeinformationen für das PaaS-Angebot Heroku fragen. Geben Sie bitte die E-Mail-Adresse und das dazugehörige Kennwort an, die Sie bei der Anmeldung auf heroku.com verwendet haben. Bitte beachten Sie, dass Ihr Kennwort bei der Eingabe nicht angezeigt wird.

```

C:\Users\schreiber>cd obtvse
C:\Users\schreiber\obtvse>heroku create ccom-toschr --stack cedar
Creating ☐ ccom-toschr... !
! Invalid credentials provided.
Enter your Heroku credentials:
Email: ccom@torstenschreiber.onmicrosoft.com
Password: *****
Creating ☐ ccom-toschr... done, stack is heroku-16
https://ccom-toschr.herokuapp.com/ | https://git.heroku.com/ccom-toschr.git
C:\Users\schreiber\obtvse>

```

Abb. 2.41: Erstellung eines Heroku-Anwendungscontainers

Anmerkung: Wenn Sie den Befehl "heroku create" ohne einen darauffolgenden Parameter ausführen, so wird der Heroku-Anwendungscontainer mit einem zufälligen Namen erstellt. Von diesem Namen hängt es ab, wie Sie später auf Ihre Anwendung zugreifen können. Mit der von uns im Beispiel gewählten Notation erhalten Sie eine Anwendung, die Sie über den Namen ansprechen können – für das Beispiel aus der Abbildung 2.41 unter <https://ccom-toschr.herokuapp.com>.

Um nun den Quellcode unserer Testanwendung in die soeben erstellte Laufzeitumgebung hochzuladen, geben Sie bitte den folgenden Befehl in die Eingabeaufforderung ein:

```
git push heroku master
```

```

remote:      No Procfile detected, using the default web server.
remote:      We recommend explicitly declaring how to boot your server process via a Procfile.
remote:      https://devcenter.heroku.com/articles/ruby-default-web-server
remote:      -----> Discovering process types
remote:      Procfile declares types      -> (none)
remote:      Default types for buildpack -> console, rake, web, worker
remote:      -----> Compressing...
remote:      Done: 39.7M
remote:      -----> Launching...
remote:      Released v5
remote:      https://ccom-toschr.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy.... done.
To https://git.heroku.com/ccom-toschr.git
 * [new branch]      master -> master
C:\Users\schreiber\obtvse>

```

Abb. 2.42: Quellcode in Laufzeitumgebung hochladen

Nun wird es Zeit, die Anwendungsdatenbank einzurichten. Hierfür geben Sie bitte den folgenden Befehl in das GIT-Kommandozeilen-Fenster ein:

```
heroku run rake db:migrate
```

```

C:\Users\schreiber\obtvse>heroku run rake db:migrate
Running rake db:migrate on ccom-toschr... up, run.7145 (Free)
DEPRECATION WARNING: You have Rails 2.3-style plugins in vendor/plugins! Support for these plugins will be removed in Rails 4.0. Move them out and bundle the
m in your Gemfile, or fold them in to your app as lib/myplugin/* and config/initializers/myplugin.rb. See the release notes for more on this: http://weblog.r
ubyonrails.org/2012/1/4/rails-3-2-0-rc2-has-been-released. (called from <top (required)> at /app/Rakefile:7)
DEPRECATION WARNING: You have Rails 2.3-style plugins in vendor/plugins! Support for these plugins will be removed in Rails 4.0. Move them out and bundle the
m in your Gemfile, or fold them in to your app as lib/myplugin/* and config/initializers/myplugin.rb. See the release notes for more on this: http://weblog.r
ubyonrails.org/2012/1/4/rails-3-2-0-rc2-has-been-released. (called from <top (required)> at /app/Rakefile:7)
-- CreatePosts: migrating =====
-- create_table(:posts)
--> 0.0136s
-- CreatePosts: migrated (0.0137s) =====
-- AddSessionsTable: migrating =====
-- create_table(:sessions)
--> 0.0090s
-- add_index(:sessions, :session_id)
--> 0.0114s
-- add_index(:sessions, :updated_at)
--> 0.0100s
-- AddSessionsTable: migrated (0.0309s) =====
-- SetPostDefaultToDraft: migrating =====
-- change_column_default(:posts, :draft, true)
--> 0.0040s
-- SetPostDefaultToDraft: migrated (0.0041s) =====
-- AddAsideToPosts: migrating =====
-- add_column(:posts, :aside, :boolean, {:default=>false})
--> 0.0066s
-- AddAsideToPosts: migrated (0.0067s) =====
-- AddUrlToPosts: migrating =====
-- add_column(:posts, :url, :string)
--> 0.0025s
-- AddUrlToPosts: migrated (0.0026s) =====
-- AddParentToPosts: migrating =====
-- add_column(:posts, :parent, :integer)
--> 0.0016s
-- AddParentToPosts: migrated (0.0017s) =====
C:\Users\schreiber\obtvse>

```

Abb. 2.43: Einrichtung Anwendungsdatenbank

Durch diesen Befehl wird das Datenbankschema der Testanwendung in die Anwendungsdatenbank geladen.

Damit haben wir unsere Testanwendung auf dem PaaS-Angebot Heroku bereitgestellt. Um die Anwendung aufzurufen, geben Sie einfach den folgenden Befehl in die Eingabeaufforderung ein:

```
heroku open
```

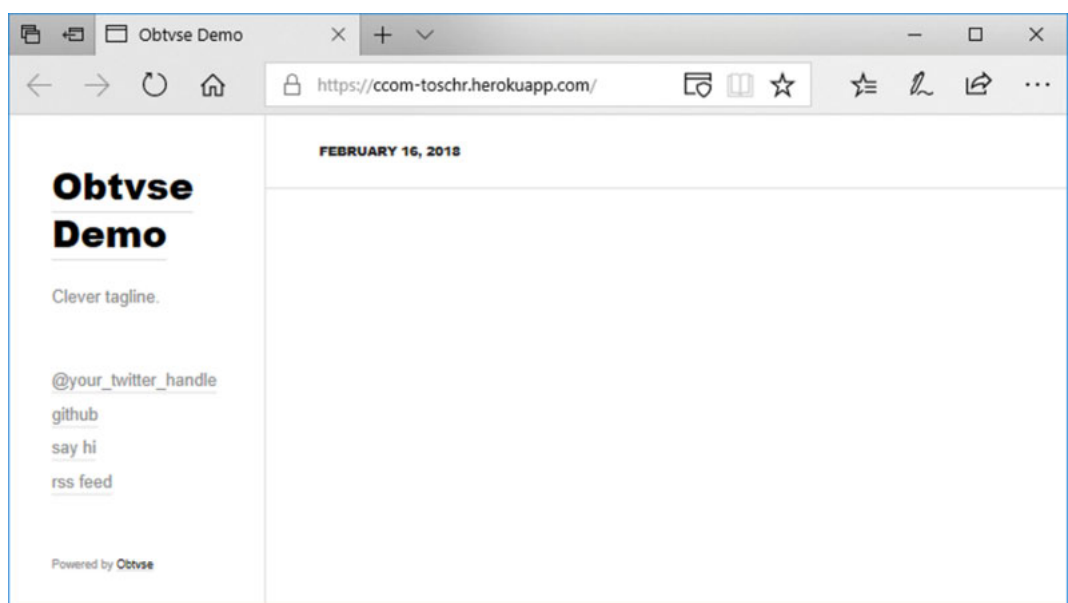


Abb. 2.44: Startseite der Testanwendung

Daraufhin sollte sich nach einer kurzen Zeit ein Browser-Fenster öffnen, das den oben angezeigten Inhalt hat.

Herzlichen Glückwunsch! Sie haben soeben Ihre erste Anwendung auf einem PaaS-Angebot bereitgestellt.

Hinweis: Sollte beim ersten Aufruf der Anwendung ein „internal server error“ erscheinen, so kann es sein, dass die Anwendung noch nicht vollständig bereitgestellt ist. In diesem Fall warten Sie bitte einfach einige Minuten, bevor Sie die Anwendung noch einmal aufrufen.



2.5 Anwendung ausprobieren

Lassen Sie uns nun die Testanwendung zusammen ausprobieren! Die Testanwendung ist ein einfaches Blog. Um neue Blog-Einträge hinzufügen zu können, hat es eine Administrationsoberfläche, die Sie durch das Hinzufügen von **/admin** zur URL öffnen können.

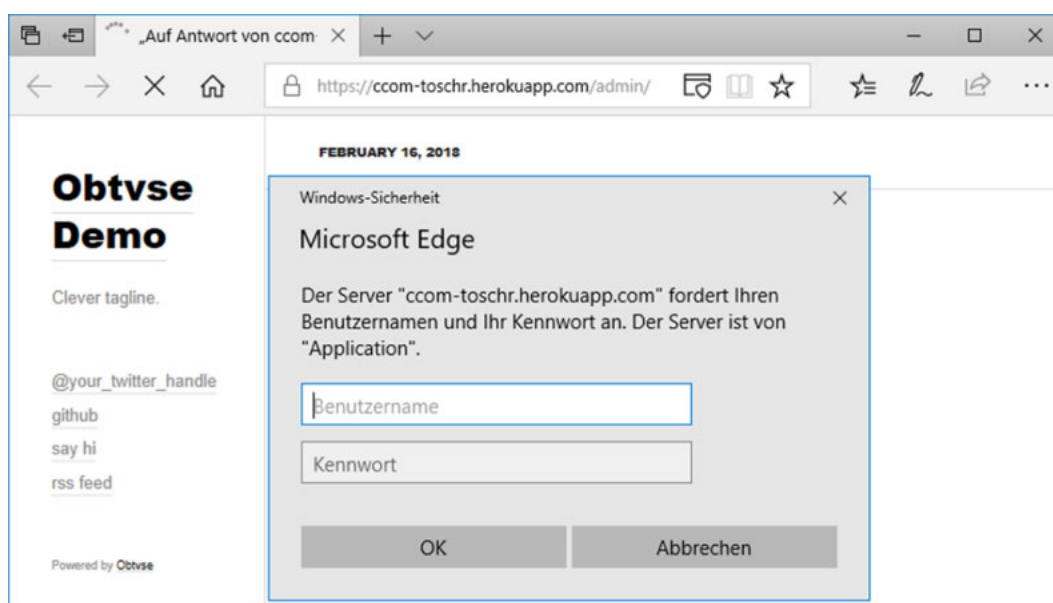


Abb. 2.45: Anmelden an der Testanwendung

Die Administrationsoberfläche wird Sie zunächst nach einem Benutzernamen und Passwort fragen. Geben Sie als Benutzernamen bitte **username** und als Passwort geben Sie bitte **password** ein und klicken Sie auf **OK**.

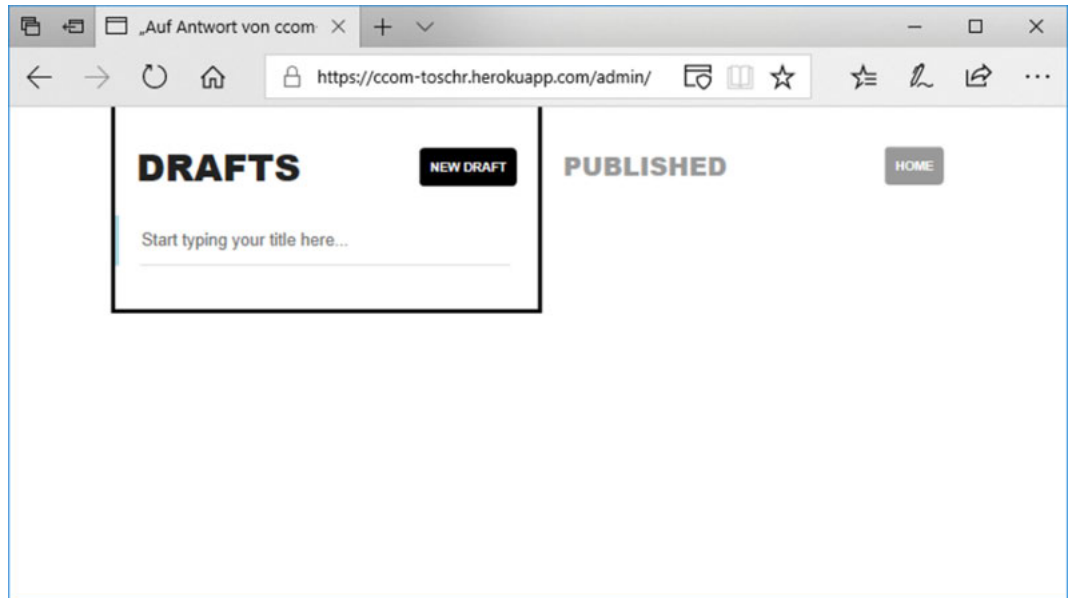


Abb. 2.46: Die Testanwendung

Um einen neuen Beitrag zu erstellen, klicken Sie nach der Anmeldung bitte auf die Schaltfläche **NEW DRAFT**.

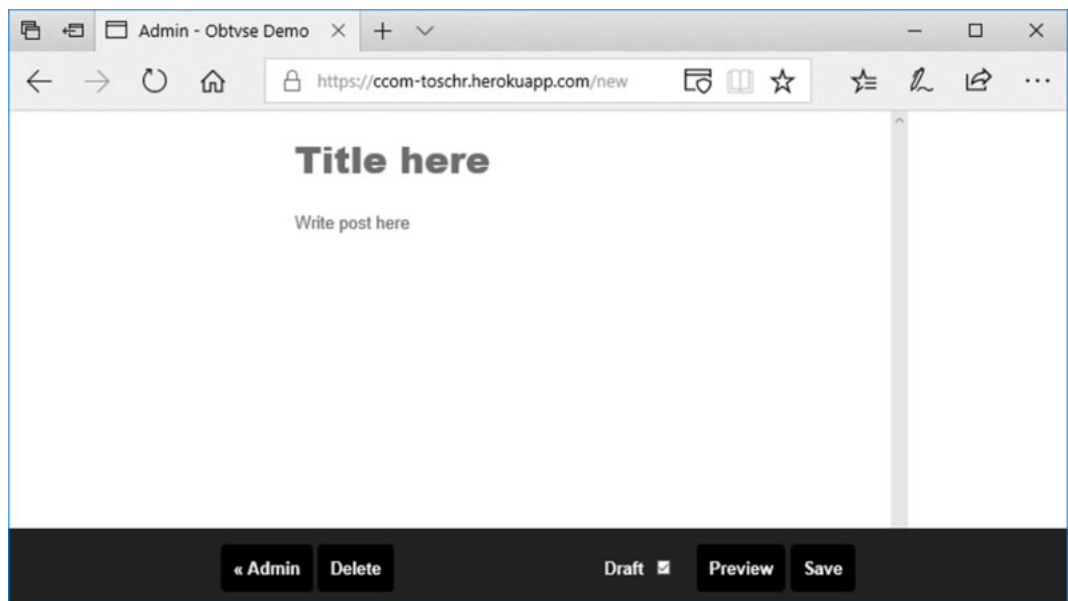


Abb. 2.47: Neuen Beitrag erstellen

In der Ansicht zur Erstellung eines Beitrags geben Sie nun bitte eine Überschrift und einen Text ein.

Um den Beitrag dann zu veröffentlichen, entfernen Sie bitte den Haken bei **Draft** und klicken Sie auf die Schaltfläche **Save**.

Nachdem der Beitrag gespeichert worden ist, klicken Sie bitte auf die Schaltfläche **Admin**, um zur Administrationsoberfläche zurückzukehren. Klicken Sie nun in der Administrationsoberfläche auf die Schaltfläche **HOME**, um das Blog anzuzeigen.

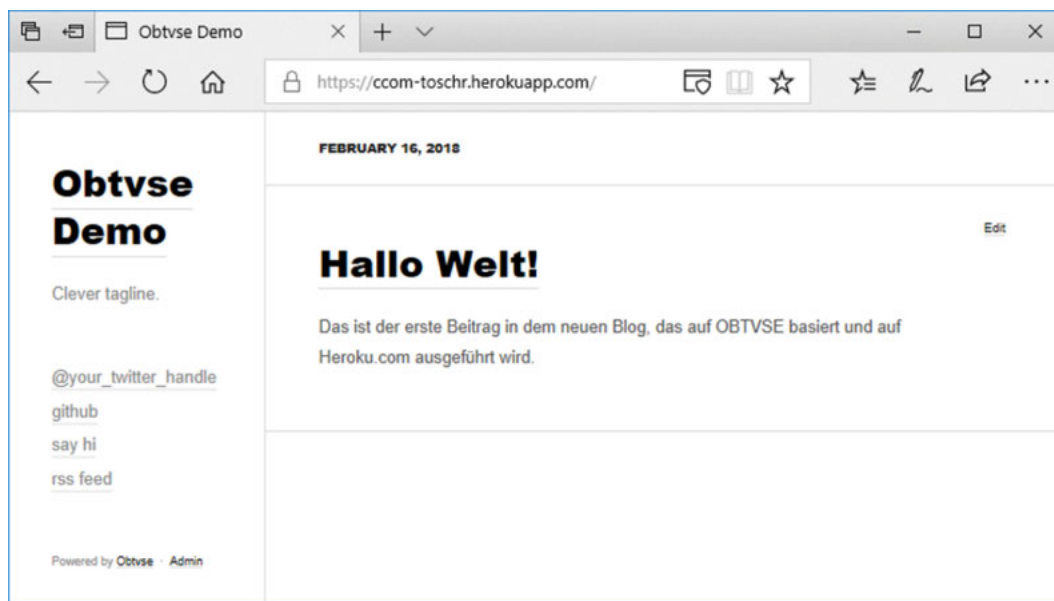


Abb. 2.48: Das Frontend der Testanwendung

Falls Sie das Blog weiterverwenden wollen, sollten Sie den Benutzernamen und das Passwort in der Datei `config/config.yml` ändern und den Code neu hochladen. Um geänderten Code in Ihre Heroku-App hochzuladen, nutzen Sie bitte die folgenden drei Zeilen:

```
git add .  
git commit -m "Benutzername geaendert"  
git push heroku master
```

Zusammenfassung

Heroku, GitHub und Amazon S3 sind sehr bekannte PaaS-Angebote. Heroku stellt Laufzeitumgebungen und Datenbanken für „Ruby on Rails“-Anwendungen bereit. GitHub stellt einen Quellcode-Speicherdienst bereit, der es verschiedenen Entwicklern erlaubt, gemeinsam über das Internet an einem Anwendungs-Entwicklungs-Projekt zu arbeiten. GitHub basiert auf dem Versions-Verwaltungssystem Git. Amazon S3 stellt PaaS-Speicherdienste bereit und wird von vielen Entwicklern zum Abspeichern von Dateien innerhalb der eigenen Anwendungen genutzt.

Für die Nutzung von Heroku und GitHub.com benötigen Sie ein Nutzerkonto, das Sie kostenlos online anlegen können. Um Quellcode in eine Laufzeitumgebung auf Heroku hochladen zu können, benötigen Sie eine Entwicklungsumgebung, die sowohl Ruby on Rails als auch Git unterstützt. Diese kann mithilfe von kostenfreien Programmen erstellt werden.

Das Bereitstellen von Anwendungen auf Heroku besteht aus den Schritten „Erstellen einer leeren Laufzeitumgebung“, „Hochladen des Quellcodes“ und „Konfiguration der Anwendungsdatenbank“. Zusätzlich werden Anwendungen, die auf Heroku betrieben werden, häufig an den PaaS-Speicherdienst Amazon S3 angebunden.

Die Testanwendung, die wir auf Heroku bereitgestellt haben, stellt ein einfaches Content Management System dar.

Aufgaben zur Selbstüberprüfung

- 2.1 Welche Informationen werden bei der Anmeldung eines Nutzerkontos bei Git benötigt?
- 2.2 Was können Sie mit dem Befehl „heroku create“ im Kommandozeilen-Fenster auslösen?
- 2.3 Was erscheint im Webbrowser, wenn Sie in der Testanwendung die Seite „/admin“ aufrufen?

3 Schlussbetrachtung

In diesem Heft haben Sie gelernt, was Platform as a Service ist und welche Dienste durch PaaS-Angebote bereitgestellt werden. Sie haben gelernt, dass sich PaaS-Angebote hauptsächlich an Entwickler richten und diesen die Arbeit für die Bereitstellung einer skalierbaren, sicheren und hoch verfügbaren Laufzeitumgebung abnehmen.

Sie haben sich mit dem Aufbau von Anwendungen, die auf PaaS-Angeboten entwickelt und betrieben werden sollen, auseinandergesetzt. Dadurch wissen Sie jetzt, dass es in der Regel notwendig wird, mehrere PaaS-Angebote miteinander zu verbinden. Sie wissen nun auch, dass webbasierte Anwendungen häufig durch zwei verschiedene Entwickler (-Teams) bereitgestellt werden. Das sind zum einen Designer für grafische Benutzeroberflächen und zum anderen Anwendungsentwickler, die die Programmlogik in sogenannten Web-Services bereitstellen.

Im praktischen Teil dieses Heftes haben Sie eine „Ruby on Rails“-Entwicklungsumgebung mit dem Versions-Verwaltungssystem GIT installiert und für die Verwendung mit GitHub.com und Heroku konfiguriert. Zum Schluss haben Sie dann Ihre erste Anwendung auf dem PaaS-Angebot Heroku bereitgestellt und diese ausprobiert.

Sebastian Stein und Torsten Schreiber

A. Lösungen der Aufgaben zur Selbstüberprüfung

Hier finden Sie die Lösungen zu den Aufgaben zur Selbstüberprüfung in den einzelnen Kapiteln. Bei offenen Aufgaben mit freien Formulierungen kommt es nicht auf eine wörtliche Übereinstimmung an, sondern auf den Inhalt. Entsprechen Ihre Ergebnisse nicht den Lösungen, wiederholen Sie bitte das entsprechende Kapitel und bearbeiten Sie die zugehörigen Aufgaben zur Selbstüberprüfung nach einer Pause erneut.

Kapitel 1

- 1.1 Richtig
- 1.2 Der Entwickler muss sich nicht mehr um die IT-Infrastruktur und den benötigten Software-Stack kümmern.
- 1.3 aPaaS – Application PaaS
- 1.4 Webbasierte Benutzeroberflächen und mobile Anwendungen für Smartphones
- 1.5 $\text{Gesamtausfallzeit} = \text{Gesamtzeit} - \text{Gesamtzeit} \times \text{Verfügbarkeit}$
- 1.6 Falsch

Kapitel 2

- 2.1 Username, E-Mail-Adresse und Kennwort
- 2.2 Die Erstellung einer leeren Laufzeitumgebung
- 2.3 Die Administrationsoberfläche der Testanwendung

B. Glossar

ACID-Prinzip	ACID steht für Atomarität (engl. Atomicity), Konsistenz (engl. Consistency), Isolation und Dauerhaftigkeit (engl. Durability) und beschreibt gewünschte Eigenschaften von Verarbeitungsschritten in Datenbanksystemen.
aPaaS – Application PaaS	aPaaS-Angebote sind der Standardtyp für PaaS-Angebote und bringen Werkzeuge und Dienste zur Entwicklung und den Betrieb von Geschäftsanwendungen mit.
API	Die Abkürzung API steht für Application Programming Interface und wird mit Programmierschnittstelle übersetzt. Die Programmierschnittstelle ist ein Teil eines Computerprogramms, das die Anbindung von weiteren externen Computerprogrammen erlaubt.
BASE-Prinzip	Ist ein Prinzip, nach dem Datenbankänderungen nicht sofort, sondern nachgelagert repliziert werden, was in einer nachgelagerten, aber trotzdem garantierten Konsistenz mündet.
BLOB-Speicher	BLOB ist eine Abkürzung für Binary Large Object und bezeichnet nicht weiter spezifizierte (aber große) Datenblöcke.
CAP-Theorem	<p>CAP ist eine Abkürzung der englischen Begriffe für Konsistenz, Verfügbarkeit und Partitionstoleranz. Das CAP-Theorem besagt, dass es für ein verteilt rechnendes System unmöglich ist, gleichzeitig die drei Eigenschaften Konsistenz, Verfügbarkeit und Partitionstoleranz zu erfüllen.</p> <p>Konsistenz (engl. Consistency) bedeutet hier: Alle Knoten sehen zur selben Zeit alle Daten.</p> <p>Verfügbarkeit (engl. Availability) bedeutet hier: Alle Anfragen an das System werden stets beantwortet.</p> <p>Partitionstoleranz (engl. Partition tolerance) bedeutet hier: Das System arbeitet trotz willkürlicher Verluste von Nachrichten weiter.</p>
Independent Software Vendor (ISV)	Independent Software Vendor (engl. für unabhängige Softwareanbieter) sind Unternehmen, deren Kerngeschäft die Produktion und der Verkauf von Software ist.
Infrastructure as a Service (IaaS)	Cloud-Computing-Angebot, das den Betrieb von virtuellen Maschinen in einer Cloud (außerhalb des eigenen Rechenzentrums) ermöglicht.

iPaaS – Integration and Governance PaaS	PaaS-Angebote, die Middleware-Dienste anbieten, um On-Premise- und Off-Premise-Dienste miteinander zu verbinden.
Laufzeitumgebung	Eine Laufzeitumgebung ist ein Computerprogramm, das die Ausführung von Quellcode ermöglicht und die Kommunikation mit dem Betriebssystem übernimmt.
Lebenszyklus von Software	Der Lebenszyklus von Software umfasst die Konfiguration, Bereitstellung, Pflege und Ablösung von Softwaresystemen.
MemCached	Ist ein Cache-Server (Software), der unter der BSD-Lizenz (Open-Source-Lizenz) steht und deshalb von vielen Webseiten und Datenbanksystemen verwendet wird.
NoSQL-Datenbanken	NoSQL-Datenbanken sind im Gegensatz zu SQL-Datenbanken nicht relationale Datenbanken und sind (im Gegensatz zu SQL-Datenbanken) horizontal skalierbar.
Permanenter Datenspeicher	Permanente Datenspeicher sind Datenspeicher, die durch Speicherung auf magnetischen oder optischen Speichermedien, Daten auch ohne den Verbrauch von Strom speichern können.
Platform as a Service (PaaS)	Cloud-Computing-Angebot, das eine Entwicklungsplattform für Web-Applikationen inkl. skalierbarer Datenbanken bereitstellt.
Serviceorientierte Architektur (SOA)	SOA ist ein Architekturmuster, das den Ansatz verfolgt, Anwendungen in wiederverwendbare Dienste aufzuteilen.
SOAP	SOAP ist eine Abkürzung für Simple Object Access Protocol und ist ein Netzwerkprotokoll, das den Austausch von Daten zwischen verschiedenen Systemen und den Aufruf von Funktionen auf entfernten Systemen erlaubt. Zur Repräsentation der Daten wird dabei XML verwendet.
Software as a Service (SaaS)	Software, die in einer Cloud betrieben wird und als Dienst genutzt und abgerechnet werden kann.
SQL-Datenbanken	SQL-Datenbanken sind relationale Datenbanken. Diese basieren auf einer Menge von Tabellen und dazugehöriger Relationen. Aufgrund des sogenannten CAP-Theorems können SQL-Datenbanken nur begrenzt horizontal skaliert werden.
UDDI	UDDI ist eine Abkürzung für Universal Description, Discovery und Integration und bezeichnet einen standardisierten Verzeichnisdienst, der Web-Services auflistet.

webbasierte Anwendung	Eine webbasierte Anwendung ist eine Anwendung, die über einen Webbrowser aufgerufen und bedient werden kann.
Web-Service	Ein Web-Service ist ein webbasierter Dienst, mit dem über XML-basierte Nachrichten Daten ausgetauscht werden können. Ein Web-Service stellt in der Regel eine API zur Verfügung, über die man die Dienste des Web-Service in Anspruch nehmen und Daten austauschen kann.
XML	XML ist eine Abkürzung für Extensible Markup Language und wird mit „erweiterbare Auszeichnungssprache“ übersetzt. Sie dient der Darstellung hierarchisch strukturierter Daten.
XML-RPC	XML-RPC ist eine Abkürzung für Extensible Markup Language Remote Procedure Call und ist eine Definition zum Funktionsaufruf auf entfernten Systemen.

C. Weiterführende Webseiten

Offizielle Webseite des Speicherdienstes Amazon S3

<https://aws.amazon.com/de/s3/>

Offizielle Webseite des „Ruby on Rails Web“-Frameworks

<http://rubyonrails.org>

Verzeichnis von „Ruby on Rails Open Source“-Projekten

<http://www.opensourcerails.com/>

Offizielle Webseite der PaaS-Sparte von salesforce.com

<http://www.salesforce.com/de/platform/>

D. Abbildungsverzeichnis

Abb. 0.1	Platform as a Service im Cloud Computing Stack.....	1
Abb. 1.1	SOA-Prinzip	3
Abb. 1.2	PaaS-Modell	5
Abb. 1.3	Architektur von Anwendungen.....	6
Abb. 1.4	Funktionsweise BLOB-Speicher	8
Abb. 1.5	Lastverteilung für Webserver	10
Abb. 2.1	Heroku-Anmeldung – Schritt 1	14
Abb. 2.2	Heroku-Anmeldung – Schritt 2	15
Abb. 2.3	Heroku-Anmeldung – Schritt 3	16
Abb. 2.4	Heroku-Anmeldung – Schritt 4	17
Abb. 2.5	Heroku-Willkommenseite	17
Abb. 2.6	Heroku Personal apps	18
Abb. 2.7	GitHub-Anmeldung – Schritt 1	18
Abb. 2.8	GitHub-Anmeldung – Schritt 2	19
Abb. 2.9	GitHub-Anmeldung – Schritt 3	20
Abb. 2.10	GitHub-Anmeldung – Schritt 4	21
Abb. 2.11	Installation Heroku CLI – Schritt 1	22
Abb. 2.12	Installation Heroku CLI – Schritt 2	23
Abb. 2.13	Installation Heroku CLI abgeschlossen	23
Abb. 2.14	Git-Installation – Schritt 1	24
Abb. 2.15	Git-Installation – Schritt 2	24
Abb. 2.16	Git-Installation – Schritt 3	25
Abb. 2.17	Git-Installation – Schritt 4	25
Abb. 2.18	Git-Installation – Schritt 5	26
Abb. 2.19	Git-Installation – Schritt 6	26
Abb. 2.20	Git-Installation – Schritt 7	27
Abb. 2.21	Git-Installation – Schritt 8	27
Abb. 2.22	Git-Installation – Schritt 9	28
Abb. 2.23	Git-Installation – Schritt 10	28
Abb. 2.24	Git-Installation abgeschlossen	29
Abb. 2.25	Die Git Bash	29
Abb. 2.26	SSH-Schlüssel generiert	30
Abb. 2.27	Geöffneter Windows-Editor.....	31
Abb. 2.28	Der öffentliche Schlüssel im Editor	31

Abb. 2.29	GitHub Settings	32
Abb. 2.30	Neuen SSH Key einrichten	32
Abb. 2.31	SSH Key hinzufügen	33
Abb. 2.32	SSH Key hinzugefügt	33
Abb. 2.33	Heroku-Kontoeinstellungen öffnen	34
Abb. 2.34	SSH Key hinzufügen	35
Abb. 2.35	SSH Key speichern	35
Abb. 2.36	Registrierter SSH-Key	36
Abb. 2.37	Git-Konfiguration	36
Abb. 2.38	Die Windows-Eingabeaufforderung	37
Abb. 2.39	Herunterladen des Quellcodes	38
Abb. 2.40	Verzeichnis wechseln	38
Abb. 2.41	Erstellung eines Heroku-Anwendungscontainers	39
Abb. 2.42	Quellcode in Laufzeitumgebung hochladen	39
Abb. 2.43	Einrichtung Anwendungsdatenbank	40
Abb. 2.44	Startseite der Testanwendung	40
Abb. 2.45	Anmelden an der Testanwendung	41
Abb. 2.46	Die Testanwendung	42
Abb. 2.47	Neuen Beitrag erstellen	42
Abb. 2.48	Das Frontend der Testanwendung	43

E. Sachwortverzeichnis

A

Abstraktion	4
ACID-Prinzip	7
Amazon S3	7, 13
aPaaS	5, 11
API.....	5, 7, 11, 13
Application PaaS	5, 11
Applikationsserver	9, 11
Architektur	6

B

BASE-Prinzip	7
BLOB	7
BLOB-Schlüssel	7
BLOB-Speicher	7
BLOB-Speicherdienst	7
Bucket	13

C

Cache	8
Cache-Server	8
CAP-Theorem.....	7
Cloud Computing Stack	1
Cloud-Computing-Art	1
Cluster	11
Clusterbildung	9
Confirm your account on Heroku.....	16

D

Dateisystem	13
Datenbank	3

E

Entwicklungsumgebung	21
Entwicklungswerkzeuge	11

F

force.com.....	6
----------------	---

G

Gesamtausfallzeit	9
Gesamtverfügbarkeit.....	9
GIT	13

Git	26
git config	36
git push	39
GitHub	13
Google Apps	6

H

Heroku	6, 13
heroku create	38
heroku open.....	40
HTML-Dateien	9

I

IaaS	1
Independent Software Vendor	3
Infrastructure as a Service	1
Integration and Governance PaaS.....	5
internal server error	41
iPaaS	5
ISV	3

K

Kompatibilität.....	10
---------------------	----

L

Lastverteiler	11
Lastverteilung	9
Laufzeitumgebung	3, 11
Lebenszyklus von Software	3
Linux-Kernel	13

M

MemCached-Dienst	8
Microsoft Azure	6
Middleware-Dienst	6
Mule iON	6

N

Net-Anwendung	11
NoSQL-Cluster	7
NoSQL-Datenbanken	7
Nutzerkonten	14

O

Open-Source-Projekt 13

P

PaaS 1

PaaS-Typen 5

Personal apps 18

Platform as a Service 1, 3

Programmiersprache 10

Q

Quellcode 38

Quellcode-Hosting 13

Quellcode-Speicherdienst 13

R

Rechenzentrum 5, 10

Redundanz 9

Ruby on Rails 10, 13

S

SaaS 1

Serviceorientierte Architektur 3

Skalierbarkeit 11

SOA 3, 5

SOA-Hype 4

SOAP 4

SOA-Prinzip 4

Software as a Service 1

Software-Stack 11

Speicherdienst 11

SQL-Datenbanken 7

SSH Key 32

SSH-Schlüssel 29

Standardisierung 4

T

Testanwendung 38

U

UDDI 4

UX 8

V

Verfügbarkeit 9, 11

W

Webframework 13

Web-Service 6, 7, 9

X

XML 4

XML-RPC 4

F. Einsendeaufgabe

Platform as a Service

Code:

CCOM05B-XX1-N01

Name:	Vorname:
Postleitzahl und Ort:	Straße:
Studien- bzw. Vertrags-Nr.:	Lehrgangs-Nr.:

Fernlehrer/in:

Datum:

Note:

Unterschrift Fernlehrer/in:

Bitte reichen Sie Ihre Lösungen über die Online-Lernplattform ein oder schicken Sie uns diese per Post. Geben Sie bitte immer den Code zum Studienheft an (siehe oben rechts).

1. An welcher Stelle des Cloud Computing Stacks wird Platform as a Service eingeordnet?
10 Pkt.
2. Was bedeutet die Abkürzung ISV?
10 Pkt.
3. Was bedeutet die Abkürzung SOA und welches Prinzip steckt dahinter?
10 Pkt.
4. Welche Aufgaben sollen Anwendungsentwicklern von den PaaS-Angeboten abgenommen werden?
15 Pkt.
5. Beschreiben Sie in eigenen Worten, was sich hinter der Abkürzung aPaaS verbirgt.
10 Pkt.
6. Nennen Sie die Teile, aus denen eine moderne webbasierte Anwendung, die auf einem PaaS-Angebot entwickelt und betrieben werden soll, prinzipiell besteht.
15 Pkt.
7. Welchen Nachteil haben SQL-Datenbanken gegenüber NoSQL-Datenbanken?
5 Pkt.
8. Welche Art von Speicherdienst sollte man in PaaS-Anwendungen für die Speicherung von großen Datenmengen nutzen? Nennen Sie diesen.
5 Pkt.
9. Wie funktioniert der sogenannte MemCached-Dienst?
10 Pkt.

10. Berechnen Sie die Gesamtausfallzeit bei einer Verfügbarkeit von 99,9 % über ein Jahr und geben Sie das Ergebnis in Stunden an.

5 Pkt.

11. Warum ist es wichtig, bei der Auswahl eines PaaS-Angebots auf die Kompatibilität zu anderen PaaS-Angeboten zu achten?

5 Pkt.

Gesamt: 100 Pkt.