4 Infrastruktur für den Betrieb

Der stetige Wandel von Technologien im IT-Umfeld stellt Firmen und Hochschulen vor immer neue Herausforderungen. Projekte können auf Grund der großen Auswahl von Technologien und Zielplattformen auf viele verschiedene weisen umgesetzt werden. Hierbei haben die beteiligten Unternehmen, die Lehrstühle als auch die Studenten Ideen, Wünsche und ggf. Vorgaben, mit welchen Technologien ein Problem zu lösen ist. Erst durch die freie Auswahl von Frameworks, Programmiersprachen und Zielplattformen, können sich sowohl Unternehmen als auch Studenten in den Projekten voll einbringen, entfalten und innovative Lösungen für Probleme finden.

Die große Herausforderung ist nun, eine Infrastruktur für Projekt zur Verfügung zu stellen, welche flexibel genug ist um die Beteiligten nicht einschränkt, aber trotzdem einen strukturierten Rahmen für die Projektabwicklung bietet.

4.1 Anforderungen und Rahmenbedingungen

Die Studenten befinden sich in der Rolle eines kleinen Software-Dienstleisters. Etablierte Firmen dieser Branche nutzen Prozesse und Werkzeuge für ihr Tagesgeschäft:

- Planung,
- Entwicklung,
- Qualitätssicherung,
- Dokumentation und Überführung in den Produktivbetrieb.

Aufgrund der Kürze der Projekte, fehlen den Studenten die zeitlichen Ressourcen, diesen Grad der Professionalisierung innerhalb eines Projektes selbst zu erarbeiten. Sie sollen sich vorrangig auf die Kommunikation mit dem Kunden, Projektmanagement und, das Entwickeln von lauffähiger Software konzentrieren.

Generelle Anforderungen

Zu den Zielplattformen bei der Softwareentwicklung, welche im Rahmen der Projekte von Bedeutung sind gehören:

- Desktop- und Server-Lösungen,
- Smartphone-Apps (Android, iOS, Windows Phone),
- Embedded Systems,
- Webanwendungen und Webservices.

Zusätzlich müssen auch die Bedingungen für Produkt- und Dienstleistungsentwicklung erfüllt sein.

Die Projektinfrastruktur sollte es nicht nur als Speicherort für Daten dienen, sondern auch als Plattform zur Koordination von Teams dienen und aktiv den Entwicklungsprozess unterstütz. Dokumentation und Nachvollziehbarkeit sind hierbei ebenso wichtig, wie ein möglichst intuitive Bedienung.

Für die Professoren und Mitarbeiter ist vor allem die Startphase der Projekte üblicherweise mit viel händischer Arbeit und Coaching verbunden. Das Ziel ist, administrative Tätigkeiten wie das Anlegen von virtuellen Servern, Projekten, Benutzerkonten, Repositories und ggf. anfallender Konfigurationsaufwand für die Projekte zu automatisieren und die Teamadministration in die Zuständigkeit der Studenten zu übergeben.

Des Weiteren wurde an der Hochschule Rosenheim aus Gründen des Datenschutzes und zur Vermeidung von Lizenzkosten und Hosting-Gebühren, eine nicht kommerzielle On-Premise Lösung bevorzugt.

Anforderungen im Projektkontext

Im nachfolgenden Abschnitt wird von einer erfolgten Planungsphase im Projekt ausgegangen. Hierzu gehören die Formulierung der Ziele, Evaluation von Ideen zur Umsetzung und der Planung des Projektablaufs in Meilensteinen. Die Einteilung der Aufgabenpakete erfolgt in Tickets, und sollte entsprechenden Meilensteinen zugeordnet sein. Der Aufwand für die Abarbeitung eines Tickets bzw. eine Aufgabe umfasst, in der Regel, maximal einen Arbeitstag.

4.2 Projektinfrastruktur

Die Wahl der richtigen Projektmanagement- und Systeminfrastruktur hilft Herausforderungen wie heterogene Technologien, multiple Zielplattformen und im Allgemeinen die Unterstützung für professionelles Projektmanagement und Softwareentwicklung, zu bewältigen.

Unsere langjährigen Erfahrungen an der Hochschule Rosenheim haben gezeigt, dass quelloffene Lösungen, vor allem durch die speziellen Anforderungen im akademischen Umfeld, viele Vorteile gegenüber proprietärer Software haben. Zusätzlich zur Anpassbarkeit und Kostenneutralität, ist eine Integration verschiedener Werkzeuge in die eigene IT-Infrastruktur leicht möglich.

Felix Hummel und Martin Kucich - die Mitarbeiter des Innovationslabors an der Hochschule Rosenheim - unterstützen andere Hochschulen gerne bei der Inbetriebnahme der Komponenten und bieten den Austausch von Erfahrungen und Know-how im Rahmen einer entsprechenden Einführung an.

4.2.1 Grundinfrastruktur (1. Ausbaustufe)

Um strukturiertes Arbeiten und Nachverfolgbarkeit auch nach Abschluss des Projekts zu ermöglichen, muss der Entwicklungsprozess durch Werkzeuge unterstützen werden. Die erste Ausbaustufe umfasst die Komponente GitLab, welche die für die Softwareentwicklung in Teams nötige Grundfunktion zur Verfügung stellt.

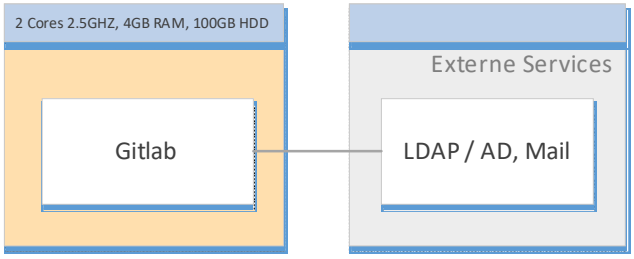

Abbildung 7: Systemumgebung der Infrastruktur bei Ausbaustufe 1.

Nutzerverwaltung und E-Mail

Eine dieser Grundfunktionen ist die Verwaltung von Nutzern, welche durch die Anbindung an das hochschuleigene LDAP / AD unterstützt wird. Durch das erste Anmelden an GitLab mit der Hochschulkennung wird ein lokaler Benutzer angelegt, wobei externe Stakeholder auch von Administratoren angelegt werden können. Die Integration eines Mailservers ermöglicht das

automatisierte Versenden von Emails – welche Emails hierbei vom Benutzer empfangen werden, kann dieser selbst in seinen Profileinstellungen festlegen.

Projekte und Gruppen

Angemeldete Benutzer können, je nach Ermessen der Administratoren, eigene Projekte und Gruppen anlegen. In Gruppen lassen sich zusammengehörige Projekte wie beispielsweise Projektsammlungen einer Lehrveranstaltung übersichtlich verwalten.

Versionsverwaltung mit Git

Die Versionsverwaltung mit Git ermöglicht das Arbeiten im Team an gemeinsamen Code, siehe 3.6. Zudem werden auch alle projektinternen Wikis über Git verwaltet, wodurch auch hier die Versionierung von Änderungen und dem entsprechend Nachverfolgbarkeit gewährleistet wird. Dieses Feature ist besonders wichtig, um gewonnenes Wissen und auch dessen Werdegang für Teams, welche das Projekt weiter betreuen, zu erhalten.

Wiki

Die Verwendung eines Wikis ermöglicht komfortabel, ein versioniertes und vollständiges Projekthandbuch zu erstellen – vergleiche Abbildung 13 im Anhang. Inhalt eines Solchen Projekthandbuchs sind üblicherweise die zu erreichenden Projektziele, organisatorische Informationen wie die Verantwortlichkeiten der Teammitglieder und der beteiligten Stakeholder. Ebenfalls wichtig sind beispielsweise die allgemeinen Rahmenbedingungen und einzuhaltenden Normen oder Richtlinien, sowie die Projektdokumentation.

Des Weiteren können Wikis verwendet werden, um gewonnenes Wissen in einer Knowledge Base zu aggregiert, was eine erneute Verwendung von erprobte Lösungen ermöglicht. Teams können so Zeit sparen, wenn es z.B. darum geht, eine Datenbank über JPA in Java anzubinden, oder LDAP als Authentifikation zu verwenden. Denkbar wäre auch, eine alle Innovationslabore übergreifende Knowledge Base einzurichten, um den gegenseitigen Wissensaustausch noch stärker zu unterstützen.

Tickets und Meilensteine

Das im Webportal umgesetzte Ticketing-System ist hierbei die zentrale Anlaufstelle zum Verwalten von Aufgaben. Die Darstellung ist als flache Liste aller Aufgaben oder als Task-Board möglich, welches besonders bei agilen Methoden Verwendung findet – siehe Abbildung 14 im Anhang. Die Zuordnung der Tickets kann auch an Meilensteine geknüpft werden und dient so als leichtgewichtiger Projektplan – siehe Abbildung 15 im Anhang.

Workflow im Projekt

Für die nachfolgende Beschreibung des Workflows gehen wir von einem erfolgten Projektstart, und spezifizierten Tickets aus. Es wird jeweils die Abarbeitung eines Tickets mit Programmieraufwand im Zusammenspiel mit der jeweiligen Ausbaustufe der Infrastruktur vorgestellt.

In dieser Ausbaustufe unterschieden wird zwei Rollen: dem Developer und dem Code Quality Manager. Letzterer ist ein anderes Teammitglied als der Entwickler. Zum einen, um durch das Vier-Augen-Prinzip Fehler zu vermeiden und Qualität zu gewährleisten, und zum anderen, dass immer mehrere Mitglieder eines Teams die Funktion des produzierten Programmcodes kennen.

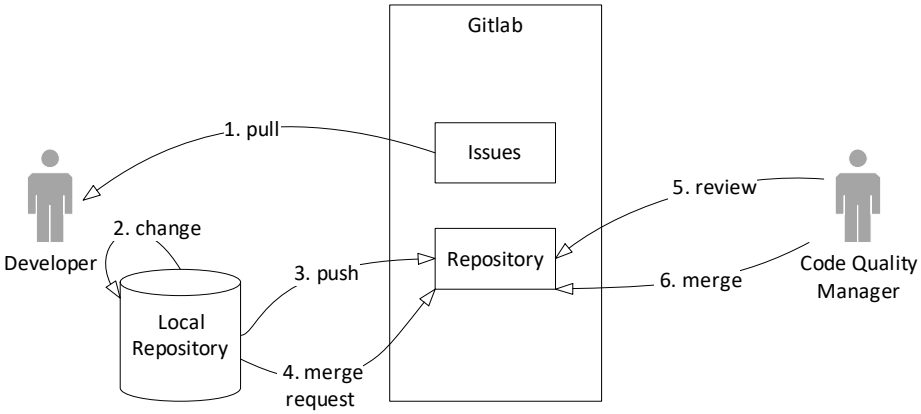

Abbildung 8: Workflow im Projekt bei Ausbaustufe 1.

1. pull: Der Entwickler holt sich einen Ticket, das ihm zugewiesen ist.
2. change: Der Entwickler programmiert und committet seine Änderungen in seine lokale Feature Branch.
3. push: Ist ein guter Zwischenstand erreicht (oft am Ende des Arbeitstages), dann "pusht" der Entwickler die Änderungen aus seinem lokalen Repository in das zentrale Repository.
4. merge request: Sobald das Feature umgesetzt ist, erstellt der Entwickler ein *Merge Request.*
5. review: Der Reviewer überprüft die Arbeit. Üblich ist, ein Code-Review durchzuführen. Fragen und Feedback können direkt am Merge-Request diskutiert werden.
6. merge: Der Reviewer akzeptiert das Merge Request, wodurch die Änderungen im "master" landen.

4.2.2 Professionalisiert Version (2. Ausbaustufe)

Die zweite Ausbaustufe erweitert die in 5.2.1 beschriebene Infrastruktur um einen Build-Server, die Container-Registry und einen Team-Chat (Mattermost). Diese Infrastruktur ist aktuell an der Hochschule Rosenheim im Produktivbetrieb und beherbergt aktuell 322 Projekte mit 730 Benutzern (Stand 20.12.2016) – siehe Abbildung 16 im Anhang für den Überblick auf der Administratorenseite.

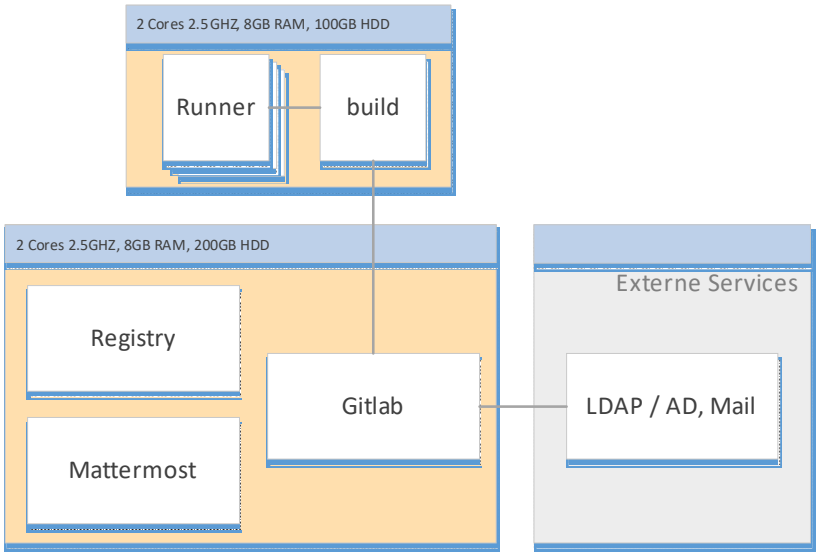

Abbildung 9: Systemumgebung der Infrastruktur bei Ausbaustufe 2.

Kontinuierliche Integration

Neben der bestehenden Infrastruktur von Wiki, Git-Repository und Aufgabenverwaltung, wurden das Erstellen (Build) und das Testen der Software automatisiert. Hierdurch können Qualitätsdefizite durch statische Codeanalyse und Ereignisprotokolle frühzeitig erkannt werden. Auf dieser Basis ist es von Beginn an möglich, geeignete Maßnahmen zur kontinuierlichen Qualitätsverbesserung zu ergreifen. Dies geschieht durch die kurzen Testzyklen automatisch, und ist in Verbindung mit dem Codereview eines Teammitglieds ein erstes Quality Gate vor dem Push auf Master. Die durch den Build-Vorgang entstandene Ausgabe lässt sich im GitLab einsehen, oder auch mit Hilfe von automatischem Reporting über Mattermost ausgeben.

Mattermost

Da häufig ein großer Teil der Kommunikation im Team über Facebook, WhatsApp und per E-Mail abgewickelt wird, ist eine spätere Aggregation der Informationen schwer. Viele dieser Informationen sind allerdings für Folgeprojekte oder andere Teammitglieder von entscheidender Bedeutung. Der webbasierte Instant-Messanger-Dienst Mattermost bietet einen Chat für alle Projektbeteiligten. Des Weiteren können Nachrichten auch von und zu Infrastrukturkomponenten abgesetzt werden. Authentifizierung erfolgt über GitLab und damit indirekt über LDAP / AD. Zusätzlich gibt es Clients für mobile Endgeräte (iOS, Android) und alle gängigen Betriebssysteme.

Registry

Die Registry ist die Ablage für die im Build-Schritt entstandenen Artefakte. Übliche Formate sind zum Beispiel:

- Docker-Images für Server-Komponenten
- Pakete diverser Programmiersprachen wie Python, NodeJS, Java oder C für Bibliotheken
- Installer für Betriebssysteme (setup.exe für Windows, deb/rpm/tar.gz für Linux, dmg für MacOS)
- Pakete für Mobile Geräte (apk für Android, ipa für iOS)

Momentan ist eine Docker-Registry mit dem Gitlab der Hochschule Rosenheim integriert – Sonatype Nexus[1] wird gerade evaluiert.

Build-Server und (GitLab-)Runner

Runner-Prozesse laufen auf den Build-Servern und führen Build-Tasks für das GitLab aus. Durch Tags können solche Prozesse für alle Projekte oder für spezifische Projekte konfiguriert werden. Die Anweisungen, wie ein Projekt gebaut wird, liegen im Code-Repository des Projektes und sind von Studenten zu pflegen. Die Runner verwenden ein Docker-Image, laden sich den Code aus dem Repository und führen dann den Build und die Tests aus. Es können aber auch VirtualBox-Images verwendet werden.

Workflow im Projekt

Zusätzlich zu dem in 5.2.1 beschriebenen Workflows, wurden die Komponenten Mattermost, Build und die Registry den Ablauf integriert. Diese Komponenten sind unerlässlich für den weiteren Ausbau hin zur kontinuierliche Lieferung und einer vollständigen Software-Development-Infrastruktur. Integrationsprobleme werden durch automatisierte Builds frühzeitig aufgedeckt, und mit dem durch Mattermost zur Verfügung gestellten Team-Chat sind die so gewonnenen Informationen aggregiert.

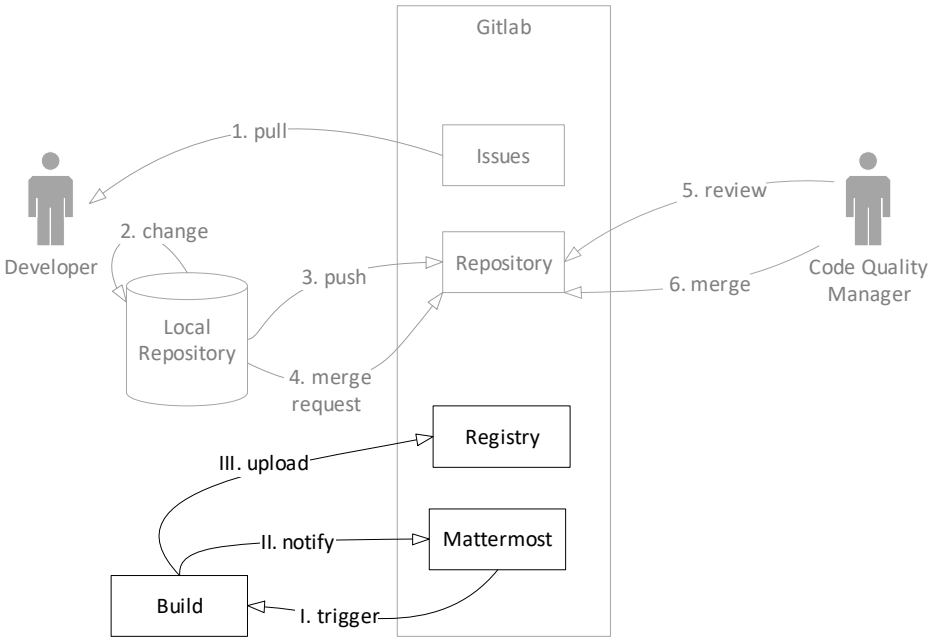

Abbildung 10: Workflow im Projekt bei Ausbaustufe 2.

I. trigger: Ein automatisierter Buildprozess (Branch) wird angestoßen.
II. notify: Benachrichtigung des Teams über Mattermost erfolgt.
Zusätzlich wird der Build vom Master angestoßen,
und der Deploy-Status wird über Mattermost veröffentlicht.
III. upload: Abschließend erfolgt der Upload in die Registry bzw. das Artefaktrepository.

[1] https://www.sonatype.com/download-oss-sonatype

4.2.3 State of the Art (3. Ausbaustufe)

Durch Hosting der gesamten Infrastruktur auf einer freien Architektur für Cloud-Computing wie OpenStack, können viele Hürden der kontinuierlichen Lieferung überwunden werden. Die nächste Entwicklungsstufe der Infrastruktur benötigt nicht nur Build-Prozesse auf einem Server, sondern zusätzlich Staging- und Produktivinstanzen. Der Creator wird neben einem Webportal für das einfache Anlegen von Projekten mit Projektskeletons und virtuellen Entwicklungsmaschinen, auch für das automatisierte Erstellen von Staging- und Produktivinstanzen zuständig sein.

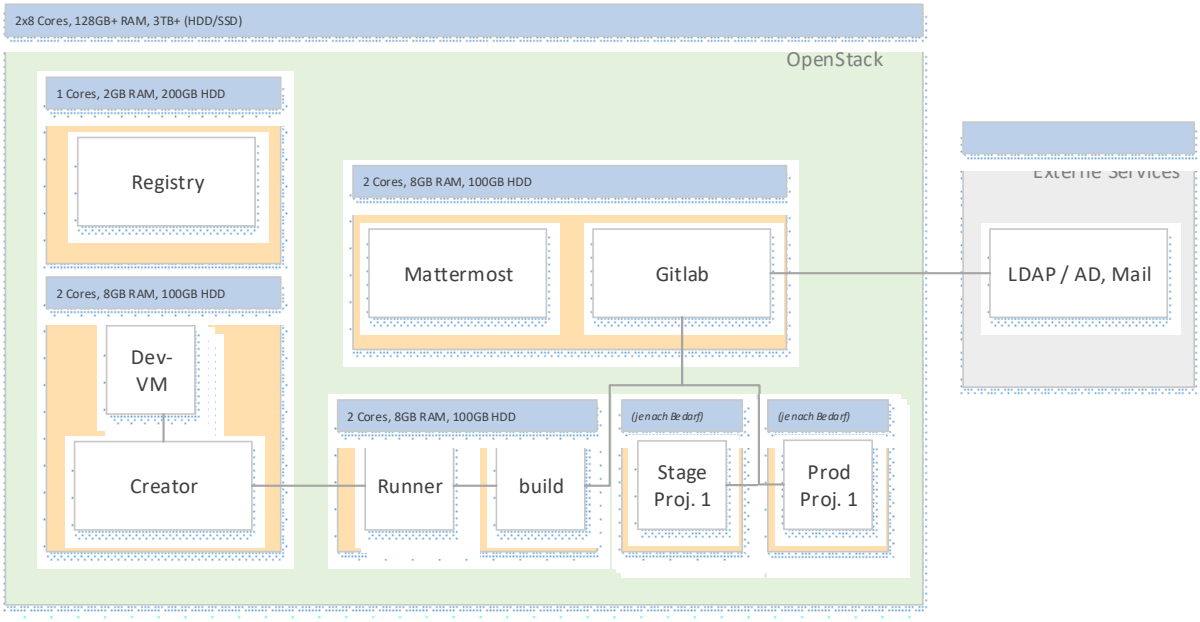

Abbildung 11: Systemumgebung der Infrastruktur bei Ausbaustufe 3.

Cloud-Infrastruktur

Der Ausbau und Skalierung der Infrastruktur um Stage und Prod, kann händisch kaum bewältigt werden. Bei beispielsweise 10 Projekten mit Serverkomponente oder Desktopapplikation müssten somit mindestens 20 Server händisch angelegt werden – bei jeweils einer Produktiv- und Staging-Instanz. Für Zielplattformen wie Mobile- oder Embedded Bereich können die Instanzen geteilt verwendet werden, allerdings werden für viele Mobile Applikationen auch Serverkomponenten benötigt, die für die Entwicklung wieder Staging und Produktivinstanzen benötigen. Die zu erwartende Last auf diesen virtuellen Servern ist verhältnismäßig gering, und die Hardware kann durch die Virtualisierung optimal genutzt werden.

Creator

Der Creator soll ein Portal zum Anlegen von Projekten für unterschiedliche Stacks bieten. Die Stacks umfassen neben der Zielplattform auch beispielsweise Frameworks und Datenbank, welche zur Verfügung gestellt werden. Auf dieser Basis können die virtuellen Server für Stage und Prod ohne großen Aufwand erzeugt und ab Projektstart vollständig in die CI / CD Pipeline integriert werden.

Auch die Provisionierung von Entwicklungs-VMs wird durch diese Komponente erledigt. So wird der Einarbeitungsaufwand durch vorgefertigte virtuelle Maschinen (VirtualBox, Vagrant, Salt), bestückt mit erprobten Entwicklerwerkzeugen drastisch reduziert.

Geplant ist die Umsetzung mit Linux und entsprechenden IDEs für die Entwicklung, zusätzlich automatisch erzeugte SSH Keys für Kommunikation mit GitLab. Ein Skeleton kann beispielsweise

folgendes enthalten: ein Hello-World-Modul, eine Lizenz-Datei, Paketinformationen, eine Beschreibung der Abhängigkeiten, Dokumentation und Tests[2].

Workflow im Projekt

In der letzten Ausbaustufe sind zwei weitere Rollen beteiligt: der Tester und Release Manager. Der Informationsfluss geht hierbei immer über Mattermost, so dass für das gesamte Team ersichtlich ist, was gerade im Projekt passiert. Im Hintergrund wird GitLab benachrichtigt und führt Aufgaben, wie den Deploy des letzten Standes auf die Staging-Umgebung oder auf das Produktiv-System, durch.

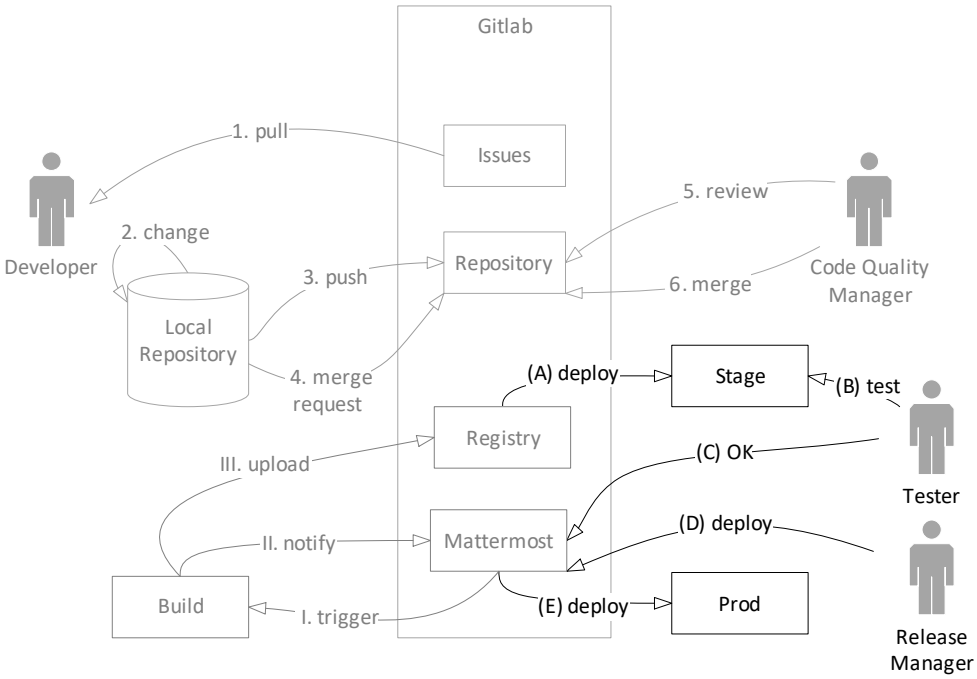

Abbildung 12: Workflow im Projekt bei Ausbaustufe 3.

A. Deploy: Automatischer Deploy auf Stage, nachdem der Build auf Master erfolgreich durchgelaufen ist.
B. Test: Der Tester testet auf Stage,
C. Ok: und informiert den Projektleiter über Mattermost über deren Ausgang.
D. Deploy: Release Manager schreibt im Mattermost beispielsweise „Deploy Prod“, worauf hin Mattermost das GitLab anweist, den Deploy einzuleiten.
E. Deploy: Deployment wird durchgeführt, und eine entsprechende Benachrichtigung über Status wird im Projektchat ausgegeben.

4.3 Ausblick

Neben den in den verschiedenen Ausbaustufen beschriebenen Werkzeugen für die Projektabwicklung, müssen zusätzliche unterstützende Komponenten in Betrieb genommen werden. Diese unterstützen

[2] http://docs.python-guide.org/en/latest/writing/structure/#sample-repository

nicht nur einen reibungslosen Betrieb der Infrastruktur, sondern sollen auch wichtige Kennzahlen visualisieren und über den gesamten Zeitraum des Betriebs aggregieren.

4.3.1 Monitoring und Logging der Anwendungen

Mit der dritten Ausbaustufe haben Teams installierbare Software, die qualitätsgesichert ist. Sollen diese Produkte in den Produktivbetrieb übergehen, dann muss auch dort die Qualität sichergestellt werden.

Hier helfen Monitoring und Logging: Monitoring zeigt die Auslastung der Maschinen, während man Logging verwendet um die Dienstgüte (Quality of Service - QoS) zu erhöhen und Nutzungsstatistiken zu erhalten. Das Innovationslabor kann diese Komponenten für alle Teams als Plattform anbieten. Momentan setzen wir Prometheus[3] mit Grafana[4] für das Monitoring sowie Elasticsearch, Logstash und Kibana (ELK)[5] für Logging ein.

Werden diese Dienste bereits von Anfang an in die Produkte integriert, dann ist die Grundlage für horizontale Skalierbarkeit geschaffen. Egal ob in der Cloud oder on-premise gehostet - das Produkt und die Teams sind darauf vorbereitet.

4.3.2 Projekt-Monitoring und Data-Warehousing

Um den Verlauf der Projekte im Blick zu behalten, bietet es sich an Statistiken zu erheben, wie zum Beispiel die Aktivität der Teammitglieder, die Anzahl der fehlgeschlagenen Builds oder die Testabdeckung. Diese Daten können auf einem Information Radiator dargestellt werden.

Sinnvoll ist es, diese Daten über den Zeitraum eines Semesters vorzuhalten. Doch was geschieht dann?

Die schiere Anzahl an Daten, die feine Granularität, die langen Zeiträume und die vielen Projekte machen es schwer, alles zu speichern. Es wäre jedoch wünschenswert, historische Daten aus allen Projekten analysieren zu können, um den Fortschritt des Innovationslabors über die Jahre darzustellen und daraus Schlüsse ziehen zu können. Dazu bietet es sich an, nach Projektabschluss aggregierte Daten in eine Datenbank zu überführen, auf der Analysen angestellt werden können.

4.3.3 Inbetriebnahme und Hilfestellungen

An der Hochschule Rosenheim wurde die unter 4.2.2 vorgestellte Infrastruktur erfolgreich in Betrieb genommen und in Verbindung mit Studentenprojekten konfiguriert, getestet und stetig weiterentwickelt. Diese so gewonnene Expertise und zusätzliche Hilfestellungen bei der Inbetriebnahme, stellen wird gerne anderen Hochschulen zur Verfügung.

Eine enge Kooperation der Hochschulen würde im Bereich der Projektbewertung durch Monitoring von Code-Qualität, durch die Entwicklung von anderen Qualitätskriterien für Projekte und dem Austausch von Workshops und Tutorials, zum Nutzen aller sein.

---

[3] https://prometheus.io/

[4] http://grafana.org/

[5] https://www.elastic.co/products