Step 1: Gitea installieren

1. Chocolaty ist eine Windows Umgebung um Anwendungen über Kommandozeile zu installieren:
    1. https://chocolatey.org/install
2. GO installieren
    1. https://golang.org/dl/
3. cmd öffnen und echo %GOPATH% eingeben. Wenn als Ergebnis wieder %GOPATH% steht, Windows neustarten und nochmal prüfen. Andernfalls unter Systemsteurung -> System -> Erweitert -> Umgebungsvariablen die GOPATH Variable setzen
4. cmd als Admin starten und folgende Befehle eingeben:
5. [crayon-67fe40cb2f97d168848237/]

6. Nun ist der Service unter http://localhost:3000 erreichbar, zeigt aber Fehler an. Daher muss man das Package herunterladen:
    1. https://dl.gitea.io/gitea/1.11.0/gitea-1.11.0-windows-4.0-amd64.exe
    2. Diese direkt in das Verzeichnis c:\gitea kopieren
7. [crayon-67fe40cb2f983390453212/]

MS SQL Server vorbereiten

8. Im SQL Management Studio nun eine Neue Datenbank mit dem Namen „Gitea“ erstellen
9. Unter Security -> Logins einen Neuen SQL User „gitea-user“ erstellen.
10. R. Maustaste auf den User -> Properties -> Server Roles und nur die neue Gitea Datenbank zuweisen
11. Mindestens einmal als dieser User einlogen und das neue Passwort setzen
12. Wieder mit Properties -> Global die Haken bei Enforce Password expiration raus nehmen

SSH Server einrichten

13. PowerShell als Admin öffnen
    [crayon-67fe40cb2f985165560857/]
14. Zum SSH Test einloggen eingeben. Standard Port ist 22 und sollte in der Firewall freigeschaltet sein
    [crayon-67fe40cb2f987836688220/]

Microservices mit .Net Framework und Core ohne Docker

Momentan liest man überall von Microservices und welche Vorteile diese Architektur mit sich bringt. Zwar wird immer wieder aufgeführt, dass Microservices Polyglot (Eine Architektur mit unterschiedlichen Programmiersprachen) unterstützen können. Tatsächlich findet man aber nur Anleitungen (In der Microsoft Welt) Zu .Net Core, in Verbindung mit Docker und Azure.

Mein Ziel ist es eine Architektur ohne Docker (Da auf VMs nicht unterstützt wird) und eine Verbindung aus Core und .Net Framework herzustellen. Aus einer Reihe von nachhaltigen Frameworks soll die Entwicklung in CD (Continues Delivery) gestaltet werden. TFS und GitLab scheiden wegen ihren Lizenzmodell aus.

- CodeRepository: Git -> https://git-scm.com/download/win
- Build Server: Jenkins -> https://jenkins.io/download/
    - https://www.guru99.com/jenkin-continuous-integrati

on.html

- Api Gateway (Kommunikation ClientApi zu Microservices) Ocelot : https://github.com/ThreeMammals/Ocelot
- Kommunikationsprotokoll zwischen Microservices: gRPC (Da schneller als Http)
- Sicherheit unter Microservices: JWT Token
- Authentifizierung Microservice mit OAuth 2.0 und OWIN Middleware: https://oauth.net/code/dotnet/
- Repository: Implementierung mit Dapper: https://github.com/StackExchange/Dapper
- Repository Cache: https://github.com/MichaCo/CacheManager
- Datenbank Code Migration SSDT: SQL Server Data Tools
    - https://www.youtube.com/watch?v=6ass_PYECmM&t
    - https://arapaima.uk/post/2017-04-04-jenkins-windows-git-ssdt-profit/
- (Optional) Search Engine: https://www.elastic.co/guide/en/elasticsearch/client/net-api/current/introduction.html
    - https://www.red-gate.com/simple-talk/dotnet/net-development/how-to-build-a-search-page-with-elasticsearch-and-net/