Assessment Senior / Lead Developer

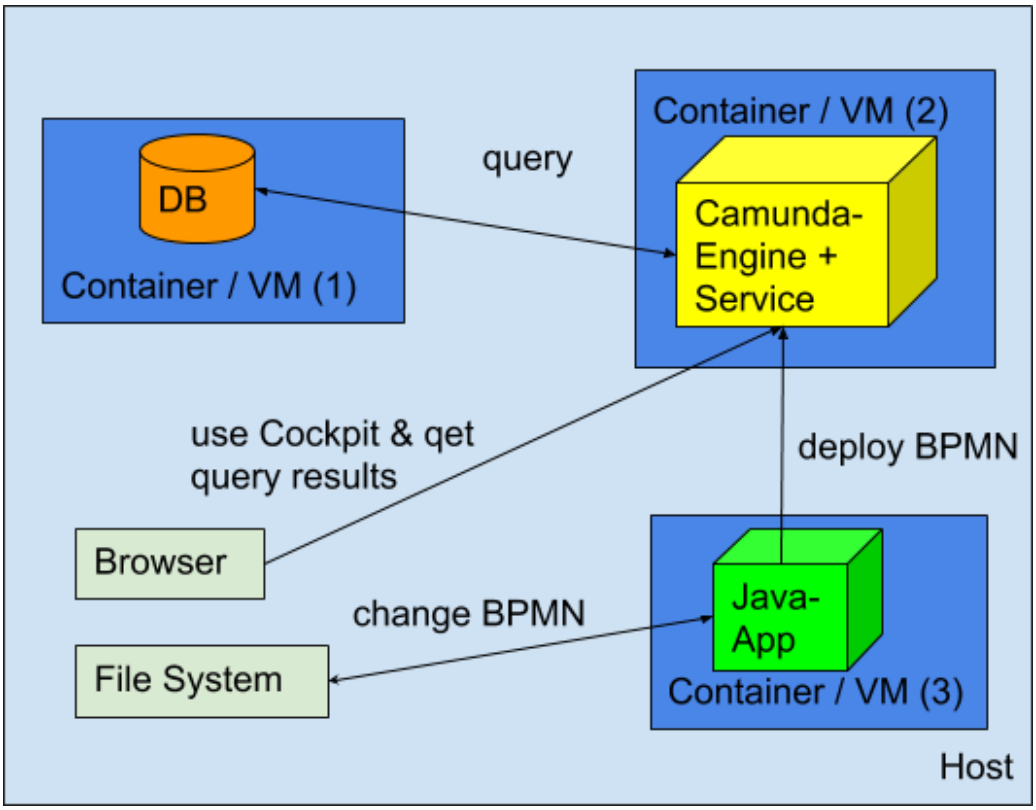
Ziel dieser Übung ist es, dass Du als Interessent für die Position als Lead Developer uns demonstrierst wie Du mit den von uns eingesetzten Basistechnologien umgehst. Wenn Du noch keine Erfahrungen mit den verwendeten Technologien hast, wollen wir herausfinden wie schnell Du Dich darin zurecht findest.

Einstiegstest / "Hausaufgabe"

Als Java/Techstack-Einstiegstest soll eine Demo-Applikation mit Camunda erstellt werden.

Bitte eine grobe Auflistung der tatsaechlichen Aufwaende pro Punkt mitprotokollieren (als CSV, XLS) im Repo.
Das Gesamtprojekt via Github- oder Bitbucket-Repository bis spätestens am Vorabend des Termins teilen (falls private, bitte zumindest flwal-alanda einladen).

Bei Rueckfragen / Unklarheiten bitte Mail an florian.waltersdorfer@alanda.io (mit marco.dworschak@alanda.io im CC).



Setup der Umgebung

Das Ziel ist einen Install Guide und die benötigten Dateien für ein Setup auf Linux/Ubuntu zu erstellen. Das Setup soll dabei in <1h hochgezogen werden können.

Mittels reproduzierbarem Setup (z.B. Docker Swarm / compose, Kubernetes, Vagrant) sollen automatisiert 3 Container / VMs erzeugt werden, die untereinander via Netzwerk kommunizieren können, unter bestimmten Ports "außerhalb" (vom Host) erreichbar sind und shares/mounts auf das Hostsystem erlauben.

Die 3 Container / VMS sind für:
1) Eine persistente DB (empfohlen Postgres), die von Camunda verwendet wird.
2) Eine Camunda-Instanz, welche die DB initialisiert und einen Backend-Service (s.u.) enthält)
3) Eine Java-Applikation, die in der Lage ist, via REST ein bpmn nach (2) zu deployen.

Best Practices zum Technologie-Stack sind unter https://camunda.com/best-practices/deciding-about-your-stack/ zu finden.

Camunda-Instanz

Ziel ist es eine Camunda-Instanz hochzufahren. Das Camunda Cockpit soll vom Host erreichbar sein, der Service soll laufen, der REST-Endpoint der Camunda-Instanz soll erreichbar sein und der DB-Container soll von der Camunda-Instanz erfolgreich initialisiert werden.

Ob Spring Boot oder Camunda Run bzw. Wildfly, Tomcat oder eine anderere Java EE Umgebung verwendet wird, liegt im eigenen Ermessen.

Innerhalb dieser Instanz soll
a) der/die DB-Container/VM initialisiert werden (Camunda-seitig)
b) ein Service erstellt werden, welcher von Camunda verwendet werden kann.
c) ein REST-Endpoint existieren, der vergangene Aufrufe des Service (b) ausgibt (in-memory store, nicht persistent).

Das Service soll eine DB-Query absetzen (z.B. Zähle alle bisher durchlaufenen Camunda Prozesse) & das Ergebnis inkl. Timestamp in-memory (Liste) speichern. Der Aufruf des REST-Endpoints (c) soll den Inhalt des Caches als JSON zurückliefern.

Material ist u.a hier zu finden: https://camunda.com/best-practices/invoking-services-from-the-process/

Java-Applikation

Ziel ist, dass nach Start der Applikation in der "leeren" Camunda-Instanz ein neuer Prozess angelegt ist. Wenn dieser nun via Camunda Cockpit gestartet wird, ist nach Durchlaufen des Prozesses durch den REST-Endpoint (c) vom Container "Camunda-Instanz" das Ergebnis des Service-Calls nachlesbar.

Es soll via Camunda-REST-API ein Demo-Prozess / ein BPMN von einem fixen Pfad in die Camunda-Instanz deployed werden, in welchem ein einzelner Service-Task aufgerufen wird.
Dieser Service-Task ruft den Service (b) vom Punkt "Camunda-Instanz" auf.
Die Java-Applikation läuft anschließend weiter und kann das Deployen bei User-Input (Konsole oder REST, nach eigenem Ermessen) wiederholen (Austauch des BPMN sollte mittels mount/share erfolgen).

Coding Kata

(to be extended)

https://github.com/emilybache/GildedRose-Refactoring-Kata