Continuous Delivery am Beispiel eines Build Files

Im Zuge der Einsendeaufgabe **Continious Delivery** wurde das build.xml File aus der vergangenen Einsendeaufgabe **Buildmanagement** mittels eines eingerichteten **Github Workflows** überprüft.

```
25 lines (20 sloc) | 613 Bytes
 <target name="clean">
          <delete dir="build"/>
 5
6
      <target name="compile">
8
          <mkdir dir="build/classes"/>
           <javac srcdir="src" destdir="build/classes"/>
9
      </target>
10
      <target name="jar">
           <mkdir dir="build/jar"/>
           <jar destfile="build/jar/Raytracer.jar" basedir="build/classes">
16
                   <attribute name="Main-Class" value="Output"/>
               </manifest>
18
           </jar>
      </target>
19
20
      <target name="run">
           <java jar="build/jar/Raytracer.jar" fork="true"/>
       </target>
24
25 </project>
```

Abbildung 1: build.xml Datei mit den Funktionalitäten Clean, Compile, Jar und Run

Der Workflow Ordner enthält ein **ant.yml** File mit einer **Java CI-Pipeline**, da die build.xml in IntelliJ ebenfalls mit dem **command-line tool Ant** entworfen wurde.

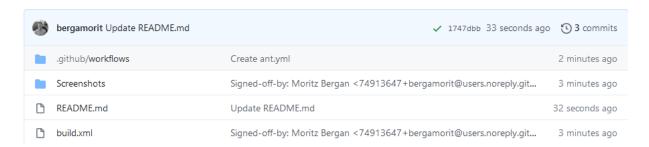


Abbildung 2: Github Workflow und build.xml im Repository

Nachdem ein neuer Push erfolgt, wird das Prüfverfahren angestoßen und die neuen Dateien/Dateiänderungen werden auf ihre korrekte Funktion überprüft.

All workflows

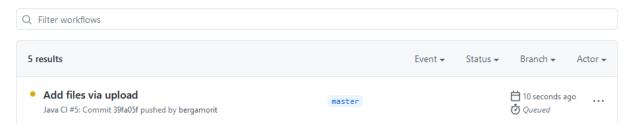


Abbildung 3: Durch neuen Push angestoßener Check der Java CI

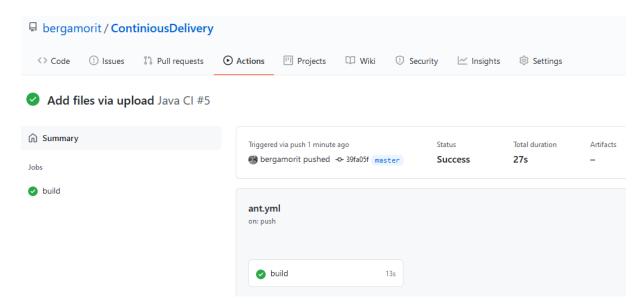


Abbildung 4: Zusammenfassung des erfolgreichen Checks inklusive Dauer der Überprüfung

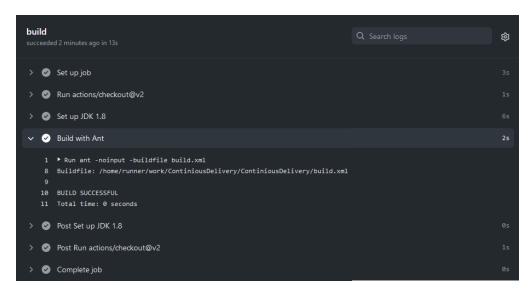


Abbildung 5: Detailansicht der überprüften build.xml