Fachhochschule
Dortmund
University of Applied Sciences and Arts

Prof. Dr. Dirk Wiesmann
Softwaretechnik D
Praktikum 11

1. Sie werden heute verschiedene statische Prüfungen des Projekts `BuchRedesign` durchführen. Schauen Sie sich vorher den Code an! Wieviele Regelverstöße erwarten Sie bei einer statischen Prüfung?

   Führen Sie zunächst die statischen Prüfungen lokal aus (siehe Aufgaben 2 bis 4).

2. Verwenden Sie das Werkzeug *PMD*. Ein Report kann mit dem *Goal* `pmd:pmd` erstellt werden. Verschaffen Sie sich einen Überlick über die Meldungen.

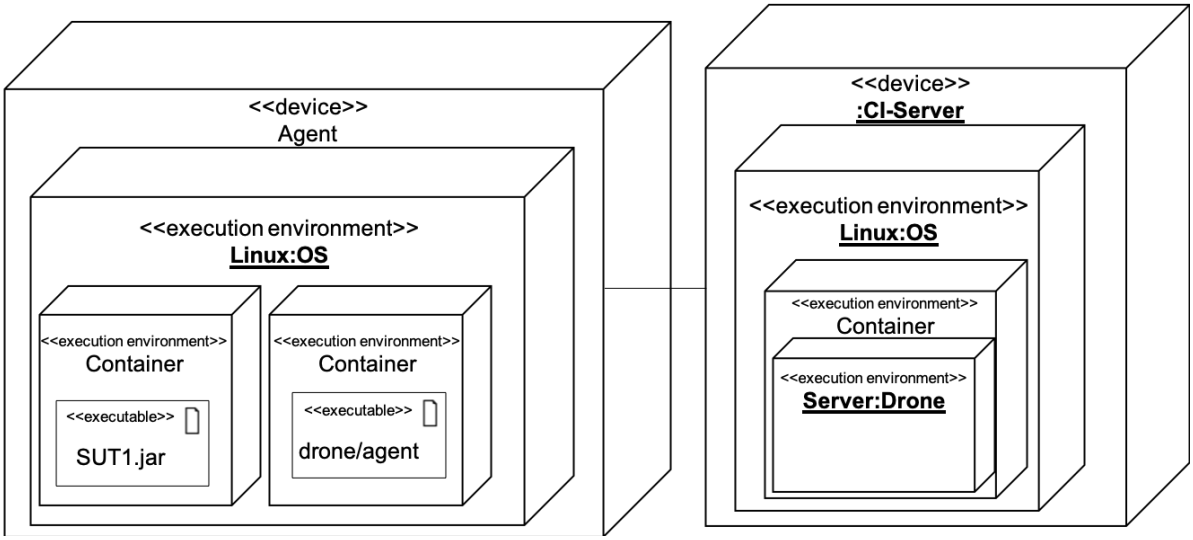
3. Verwenden Sie nun *Checkstyle*. Eine Analyse mit einer Report-Erstellung kann mit dem *Goal* `checkstyle:checkstyle` gestartet werden. Verschaffen Sie sich einen Überblick über die Meldungen.

4. Wählen Sie aus allen Regelsätzen (*Checkstyle*, *Spotbugs*, *PMD*) ingesamt 5 Regeln aus, die Sie in einem professionellen Projekt verwenden würden! Begründen Sie Ihre Wahl!

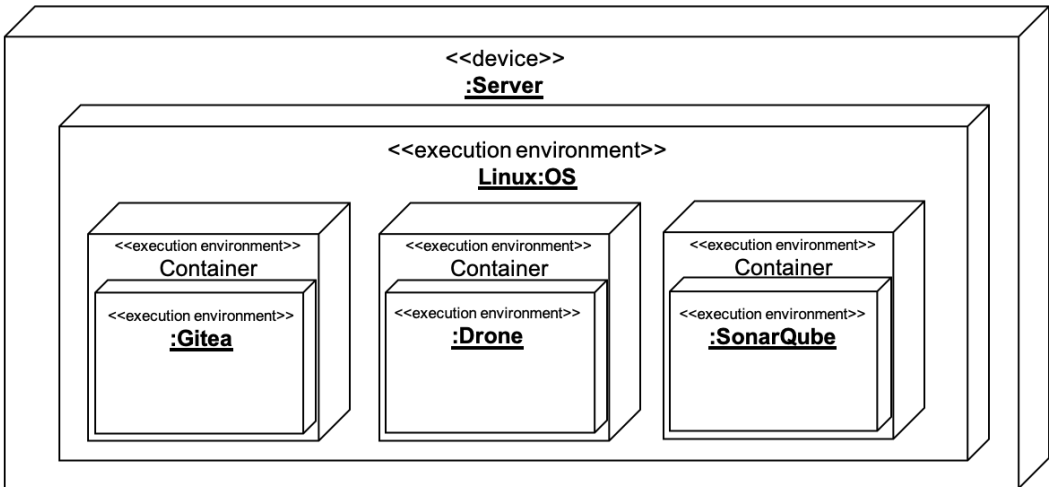
5. Nutzen Sie den Integrationsserver *Drone* (`http://drone.pipeline`), um den *Build*-Prozess automatisiert ablaufen zu lassen! Der Integrationsserver *Drone* arbeitet auf der Basis von Containern[1]. Jeder Integrationslauf wird in einem eigenen Container gestartet. Die automatische Erstellung von einheitlichen Integrationsumgebungen wird dadurch erzwungen und erleichtert. Die einzelnen Intergrationsläufe finden vollständig isoliert voneinander statt. Um eine Lastverteilung zu erreichen, können die Integrationsläufe auf entfernten Servern gestartet werden. Diese Server müssen mit *Agenten* (*agents*) ausgestattet sein[2]. Diese erhalten vom zentralen *Drone*-Server Anweisungen zur Ausführung des *Build*-Prozesses.

[1] Ein Container stellt eine abgegrenzte Laufzeitumgebung dar. Auf einem Unix/Linux Betriebssystem kann eine "beliebige" Anzahl von Containern gestartet werden. Die Prozesse, die innerhalb eines Containers laufen, sind vollständig von den Prozessen außerhalb des Containers abgegrenzt. Ein Programm kann mit all seinen Abhängigkeiten in einem Container bereitgestellt werden. Dadurch können unter bestimmten Randbedingungen die Verteilung, die Konfiguration und der Betrieb von Softwaresystemen vereinfacht werden.

[2] In der aktuellen Installation werden keine verteilten Agenten genutzt.



Auch die CI-Server-Anwendung *Drone* läuft selber in einem Container. In der vorliegenden Installation laufen alle Server-Anwendungen der Werkzeugkette in eigenen Containern[3]. Aktuell laufen alle Container auf einer Server-Instanz.



[3]Die vollständige Werkzeugette kann automatisiert über *Ansible* auf einem entfernten Server installiert werden.

Der Container für den Integrationslauf muss geeignet konfiguriert werden. Dazu muss das Projekt um eine Konfigurationsdatei `.drone.yml` ergänzt werden.

```
pipeline:
  build:
    image: maven:3.5-alpine
    commands:
      - mvn --settings ./settings.xml clean test sonar:sonar
```

Der *Drone*-Server analysiert diese Datei und erstellt einen entsprechenden Container. Der Container enthält eine Maven-Installation und ein Plugin für *Sonarqube*. Innerhalb des Containers wird dann der vorgegebene *Maven*-Befehl (siehe `commands`) ausgeführt.

Bisher haben wir die Werkzeuge *Checkstyle*, *Spotbugs* und *PMD* für die statische Code-Analyse verwendet. Die entsprechenden Reports wurden im Unterverzeichnis `target` des Projekts gespeichert. Dies ist gerade bei der Verwendung von Containern unpraktisch, da nach dem Integrationslauf der Container automatisch gelöscht wird.

Wir nutzen daher nun *Sonarqube* (`http://sonarqube.pipeline`), um die Ergebnisse des Integrationslaufs an einer zentralen Stelle verwalten zu können. Zudem wird der komplette historische Integrationsverlauf gespeichert.

Die Datei `pom.xml` muss daher entsprechend angepasst werden. Entfernen Sie alle Abhängigkeiten zu *Checkstyle*, *Spotbugs* und *PMD*. Die statische Analyse erfolgt nun über die *Sonarqube*-Scanner. Fügen Sie daher den folgenden Eintrag dem `reporting`-Element hinzu.

```
<plugins>
   <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>sonar-maven-plugin</artifactId>
      <version>3.6.0.1398</version>
   </plugin>
</plugins>
```

Das Projekt muss um eine Datei `sonar-project.properties` ergänzt werden, um entsprechende Pfade für die *Sonarqube*-Scanner zu setzen.

```
sonar.java.binaries=./target/classes
sonar.sources=src/main/java
```

Für die Integration von *Sonarqube* in *Maven* wird das Projekt um eine Datei `settings.xml` ergänzt[4].

```
<settings>
   <pluginGroups>
      <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
   </pluginGroups>
   <profiles>
      <profile>
         <id>sonar</id>
         <activation>
            <activeByDefault>true</activeByDefault>
         </activation>
         <properties>
            <sonar.host.url>
               http://sonarqube.pipeline
            </sonar.host.url>
            <sonar.junit.reportPaths>
               target/surefire-reports
            </sonar.junit.reportPaths>
            <sonar.coverage.jacoco.xmlReportPaths>
               target/site/jacoco
            </sonar.coverage.jacoco.xmlReportPaths>
         </properties>
      </profile>
   </profiles>
</settings>
```

Nun muss *Drone* mit dem *Repository* verbunden werden. Melden Sie sich unter `http://drone.pipeline` an. Es gelten dabei die Anmeldedaten von *Gitea*. Selektieren Sie den Menüeintrag *Repositories* und aktivieren Sie dort das relevante *Gitea-Repository* mit dem Schiebeschalter. Danach wechseln Sie in die *Build*-Ansicht. Dort wählen Sie im Menü den Eintrag *Secrets*. Dort legen Sie die folgenden Einträge an, damit `Drone` mit den anderen Komponenten der *Build-Pipeline* kommunizieren kann.

- Name: `sonar_host` Value: `http://sonarqube.pipeline`
- Name: `sonar_token` Value: `ce92e014b149f9a911b9dfd78042451a4357b4ee`
- Name: `gitea_token` Value: `drone`

[4]Normalerweise befindet sich diese Datei in einem *Maven*-Verzeichnis. Da wir nun in einem temporären Container arbeiten, wird die Datei stattdessen im Projektverzeichnis gespeichert. Beim Start von *Maven* muss der Pfad auf die Datei entsprechend gesetzt werden.

6. Bitte nehmen Sie in den Maven-Projektnamen Ihren Gruppennamen auf. Der Projektname wird in *Sonarqube* angezeigt. Lösen Sie einen Integrationslauf durch eine Code-Änderung im *Repository* (*Gitea*) aus. Kontrollieren Sie die Prüfergebnisse in *Sonarqube*.