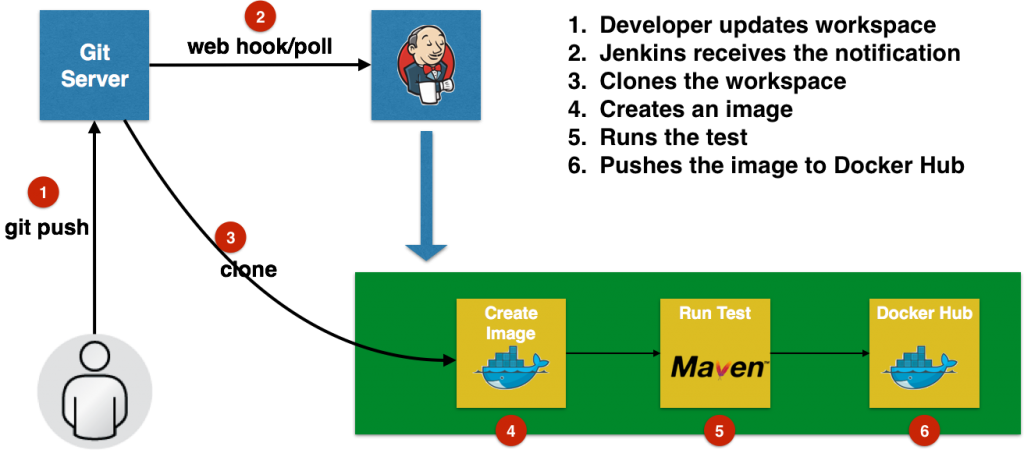
**Mandatory II – Jenkins**

**Jenkins pipeline architecture:**



**Instance setup:**

Vi opretter en EC2-instans med Amazon AWS. Det skal lige huskes at sige at man inde på AWS, skal vælge en ubuntu-instans med minimum 8GB ram(t.2large), så både Docker og Jenkins kan køre uden problemer.   
Når man opretter en EC2-instans, skal man oprette en nøglefil, som så giver dig adgang til at køre denne EC2-instans gennem terminalen  
Når EC2-instansen er blevet oprettet, så skal man ændre i *Inbound-rule*s, og tilføje en ny regel for hver port, hvilket er 8080, samt 80, så programmet kører i porten 8080.

For at komme ind på instancen fra serveren så skal man indtaste en linje under instancen på AWS.

* $ sudo ssh -i keyfile.pem ubuntu@18.185.82.12

Når vi er inde på selve instancen i vores terminal, så er starter vi med at installerer Docker ved anvende disse kommand line:

* sudo apt update // Used to re-synchronize the package index files from their sources.
* wget -O - https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
  + sudo add-apt-repository \

   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

   $(lsb\_release -cs) \

   stable"

* sudo apt update
* sudo apt install docker-ce
* sudo docker run hello-world
* sudo usermod -aG docker $USER
* sudo reboot

Nu er vi klar til at installerer Jenkins på vores instance, dette gør vi på følgende måde ved først at installerer jdk verion 8:

* sudo apt install openjdk-8-jdk
* wget -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add
* sudo apt update
* sudo apt install jenkins
* systemctl status jenkins

Nu er Jenkins installeret og klar til at bruge. For at starte med at bruge jenkins, så skal vi på browseren og indtaste vores IP-adresse og efterfølgende /8080

Når vi er kommet ind i Jenkins, så skal vi oprette et brugernavn og installerer de nødtvendige plugins.

Setup Jenkins:

Vi er nået til den fase hvor vi skal oprette en API key som skal hjælpe os med at forbinde til GitHub, dette sker i github under personal access tokens. Her får vi udleveret en kode som vi skal til at genererer en credentials.

Install Maven:

* 1. In your AWS Ubuntu terminal install Maven
     1. $ sudo apt install maven
     2. $ ls /usr/share/mav\*
     3. Now you should see some maven libraries
  2. In Jenkins dashboard, select Manage Jenkins, Global Tool Configuration
     1. Scroll down to Maven, and click Add Maven
     2. Uncheck the “Install Automatically” checkbox
     3. In textbox MAVEN\_HOME enter:   /usr/share/maven   And give it a name in the Name textbox
     4. click Save

Når vi har installeret maven, så er det tid til at lave et Webhook. Webhooket laver man inde på Github inde på det Repository man ønsker Jenkins kører. Dette gør man inde på repositoriets indstilliner, hvor efter man indsætter sit EC2 IP-adresse samt /github-webhook/.

Nu er det endelig tid til at lave Jenkins-projektet, så vi kan oprette en pipeline. Dette gør man inde på forsiden, hvor man trykker på **”New Item”**, i dette tilfælde kalder vi det ’myproject’. 🡪 Derefter trykker vi på Freestyle Project. Vi indsætter vores Github-repository link, siden vores projekt er åbent, så er der ikke behov for vores credentials.  
  
Derefter går vi ind på **Build Triggers** 🡪 **Configure 🡪 ’ Add build Step ’ 🡪 Sørg for Maven er konfigureret. 🡪 Tid til test.**

Vi afprøver vores kode 🡪 Returnerer succesful 🡪 Ændre i koden 🡪 Jenkins begynder med at builde igen. 🡪 Jenkins virker

For at se om koden virker, så henter vi Docker, ved at indtaste kommandoen Docker-kommandoen inde på kommandoprompten. Efter dette tester vi Docker for at se om det virker.

Derefter tilføjer vi Build Steps til Docker, og gemmer dette valg, så burde dette køre automatisk fremover i stedet for man selv skal pushe hver gang.  
  
Derefter er vi færdige. Vi prøver på at se om vores jenkins køre optimalt ved at ændre noget kode i IntelliJ og pushe efter. Hvis koden kører optimalt, bygger den programmet for os, uden at vi skal gøre noget andet.

**Udfordringer**:

Vi havde flere problemer med at køre Jenkins, samt finde nogle af de løsninger på vores problemer vi havde. Nogle af de problemer vi stødte på, var at første gang vi kom ind på Jenkins havde vi brug for en adgangskode, dette løste vi ved at få koden fra kommandoprompten.

Det andet problem vi havde, var at oprette en token gennem Github. Dette viste sig, at man skulle oprette en Secret Server, og få et ID derfra, modsat guiden der henviste til at man skulle oprette token igennem Github-login samt kode, det virkede derefter. Vi brugte ikke rigtigt dette, da det mere skulle bruges hvis Github-repository var privat.

Den største udfordring vi har haft er nok at vi ikke kunne hoste vores hjemmeside via Jenkins, det har vi brugt en del tid på. Men vi fandt heldigvis ud af at man kan bruge Docker til denne opgave.

**Refleksion:**

De valg vi traf ved at lave dette projekt var ikke kommet let. Da vi første gang begyndte i Jenkins, så i stedet for et freestyle-projekt, valgte vi pipeline, som vi troede ville gøre det samme som vores freestyle-projekt gjorde i denne omgang, derimod stødte vi på en masse fejl. Derefter spurgte vi nogle gamle klassekammerater om hjælp, og fandt ud af at vi måske skulle prøve en anden strategi, og det førte os frem til at gøre det på denne måde.

Men før vi spurgte vores klassekammerater, så lavede vi en del research på internettet, hvor vi fandt en del tutorials på at lave en pipeline. Alle tutorials viste hver deres måde at stille en pipeline op på, hvilket gjorde os en del forvirret omkring hvordan vi skulle stille vores pipeline. Vi gik i gang med at følge 2-3 tutorials som ikke hjælp os min vores projekt, da de brugte andre ressourcer end dem vores projekt stillede krav til.

Vi valgte også at lave vores pipeline som jenkins pipeline-projekt, hvor man deler CI og CD i forskellige stages. Problemet her var at vi ikke vidste hvordan vi skulle hoste vores hjemmesiden, derfor fik vi den ide at lave en sh commad, hvor vi kører jar filen manuelt med Java. Det hele fungeret perfekt, alle test kørte som de skulle, projektet blev deployet ad sig selv. Men da vi pushed koden til git og webhook auto kørte jenkins, så fejlmeldte den at port 9090 var optaget. Vi fandt efter noget tid frem til en løsning. Men jenkins fejlmeldte stadig, da hver gang der blev pushed til git, så afbryd jenkins det kørende program. Efter at vi havde rodet med det en del, så kom vi frem til at der måtte være en bedre løsning, da den løsning vi har udarbejdet, er for rodet og kompleks. Derfor valgte vi at søge nye muligheder, hvor på at vi fik hjælp fra nogle klassekammerater som hjælp os på rette vej.