Workshop 2

Jenkins pipeline

Gemaakt door:

Tobias Eckhart

Stijn Nooteboom

Eelke Hoffman

Wesley Ackah

Inhoud

[Voorbereiding 3](#_Toc2254393)

[Downloaden en installeren vooraf 3](#_Toc2254394)

[VMware Workstation Player: 3](#_Toc2254395)

[Installeren docker en het inladen van jenkins 6](#_Toc2254396)

[Jenkins 8](#_Toc2254397)

[Voorbereiden GitHub 10](#_Toc2254398)

[Pipeline bouwen 10](#_Toc2254399)

[Definiëren Jenkinsfile 13](#_Toc2254400)

[Jenkinsfile stappen 15](#_Toc2254401)

[Errors 19](#_Toc2254402)

[Vmplayer 19](#_Toc2254403)

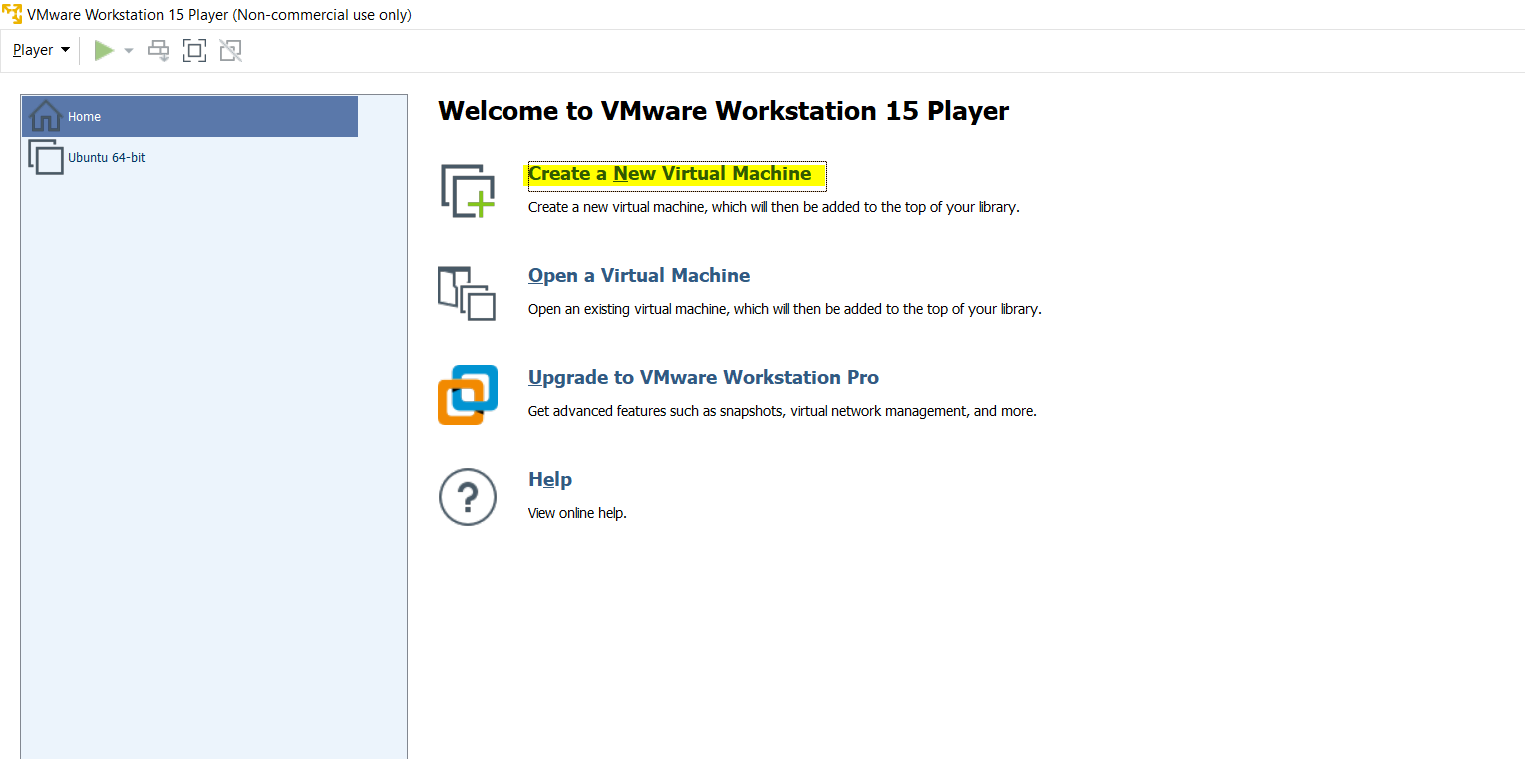
# Voorbereiding

## Downloaden en installeren vooraf

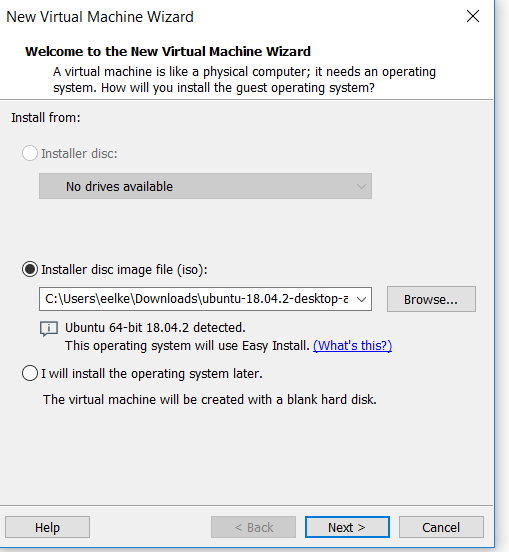
1. Download Ubuntu op <https://www.ubuntu.com/#download>
   1. Kies Ubuntu Desktop, de LTS versie (groene knop).
2. Download VMware Workstation Player op <https://www.vmware.com/nl/products/workstation-player/workstation-player-evaluation.html>
3. Installeer VMWare Workstation Player
4. Start VMware Workstation Player op
   1. Selecteer non-commercial use
   2. De aangeboden update kan worden overgeslagen

## VMware Workstation Player:

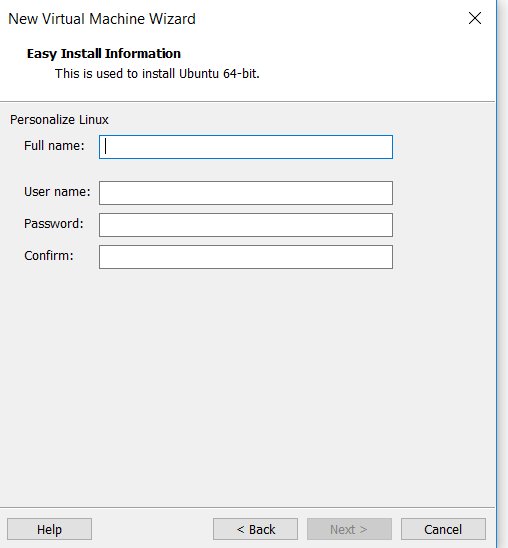
1. Selecteer create a new virtual machine



1. Refereer naar de Ubuntu ISO die je net hebt gedownload



1. Vul de gevraagde gegevens in



1. Druk op next> next> next> finish
2. Start de virtuele machine, deze gaat nu Ubuntu automatisch installeren.\*

*\* Als dit niet lukt, zie de laatste pagina voor de oplossingen voor de errors.*

1. Indien er een melding komt voor het installeren van VM Player software voor Ubuntu, selecteer ja.

# Installeren docker en het inladen van jenkins

1. Update van ubuntu

sudo apt-get update

1. Repo van docker voorbereiden

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo apt-key fingerprint 0EBFCD88

sudo add-apt-repository \

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

$(lsb\_release -cs) \

stable"

1. Installeer docker

sudo apt-get install docker-ce docker-ce-cli containerd.io

1. Jenkins in docker container laden

sudo docker run \

-u root \

--rm \

-d \

-p 8080:8080 \

-p 50000:50000 \

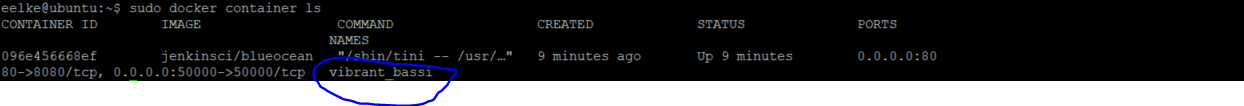
-v jenkins-data:/var/jenkins\_home \

-v /var/run/docker.sock:/var/run/docker.sock \

jenkinsci/blueocean

1. Lijst met docker containers

sudo docker container ls



* 1. Noteer de container naam die je terugkrijgt zoals hierboven is weergegeven

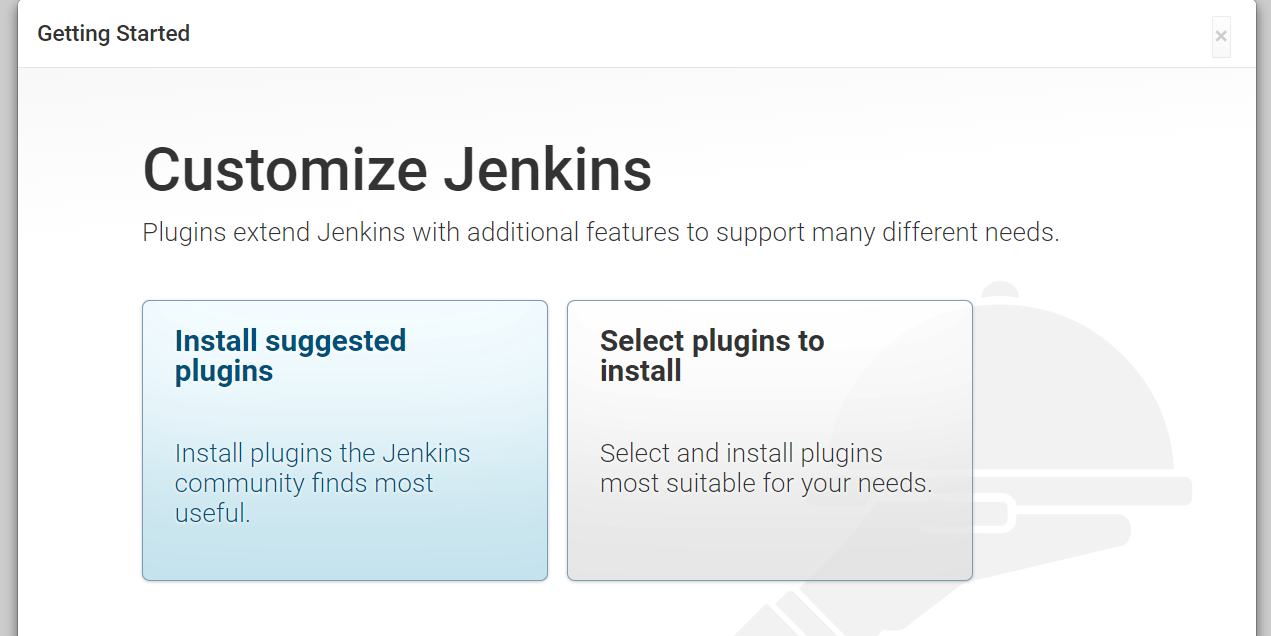
1. admin pass uitlezen

sudo docker exec -it <container name> cat/var/jenkins\_home/secrets/initialAdminPassword

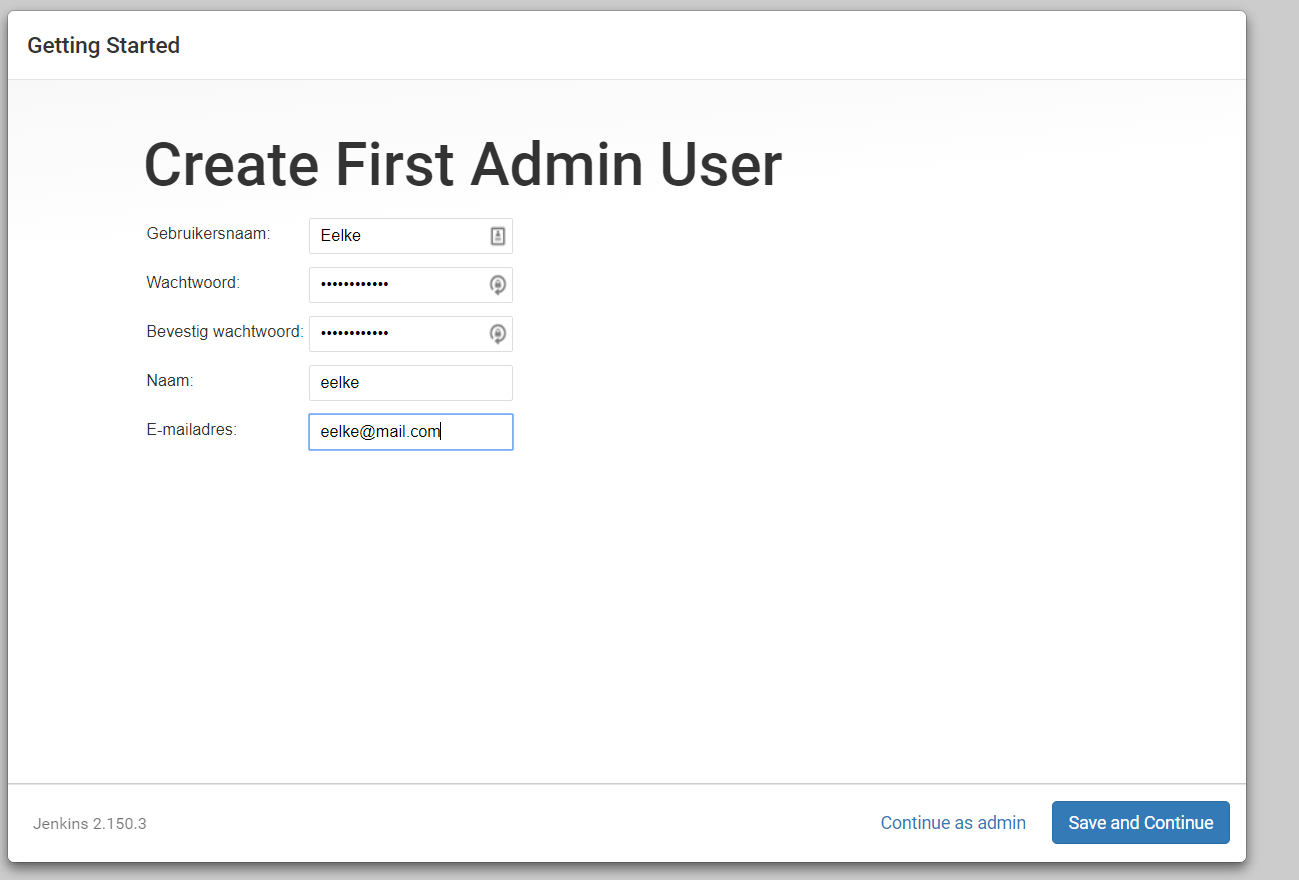
1. Browse nu naar localhost:8080
2. Vul het wachtwoord in dat het laatste commando gaf

# Jenkins

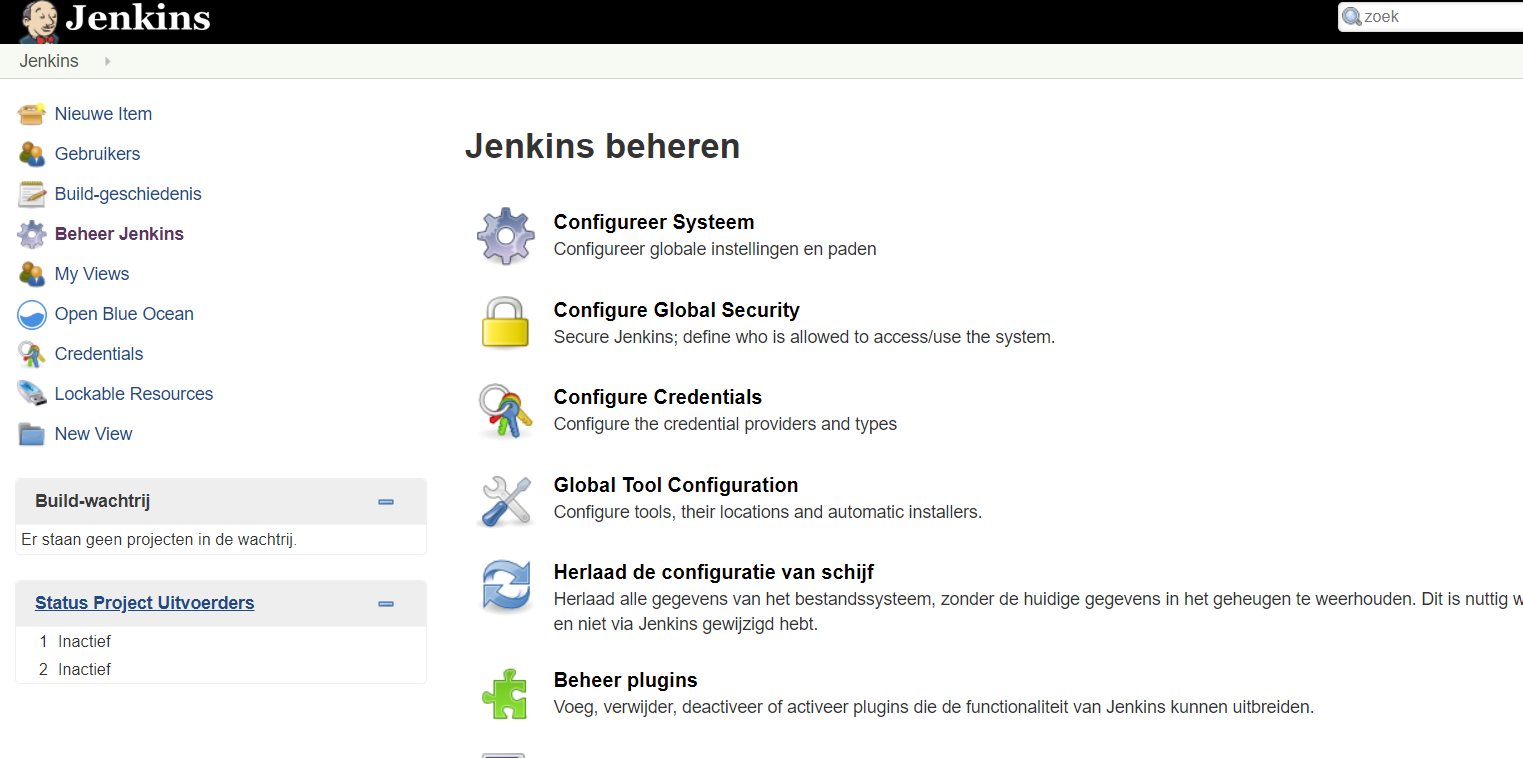
1. Selecteer install suggested plugins



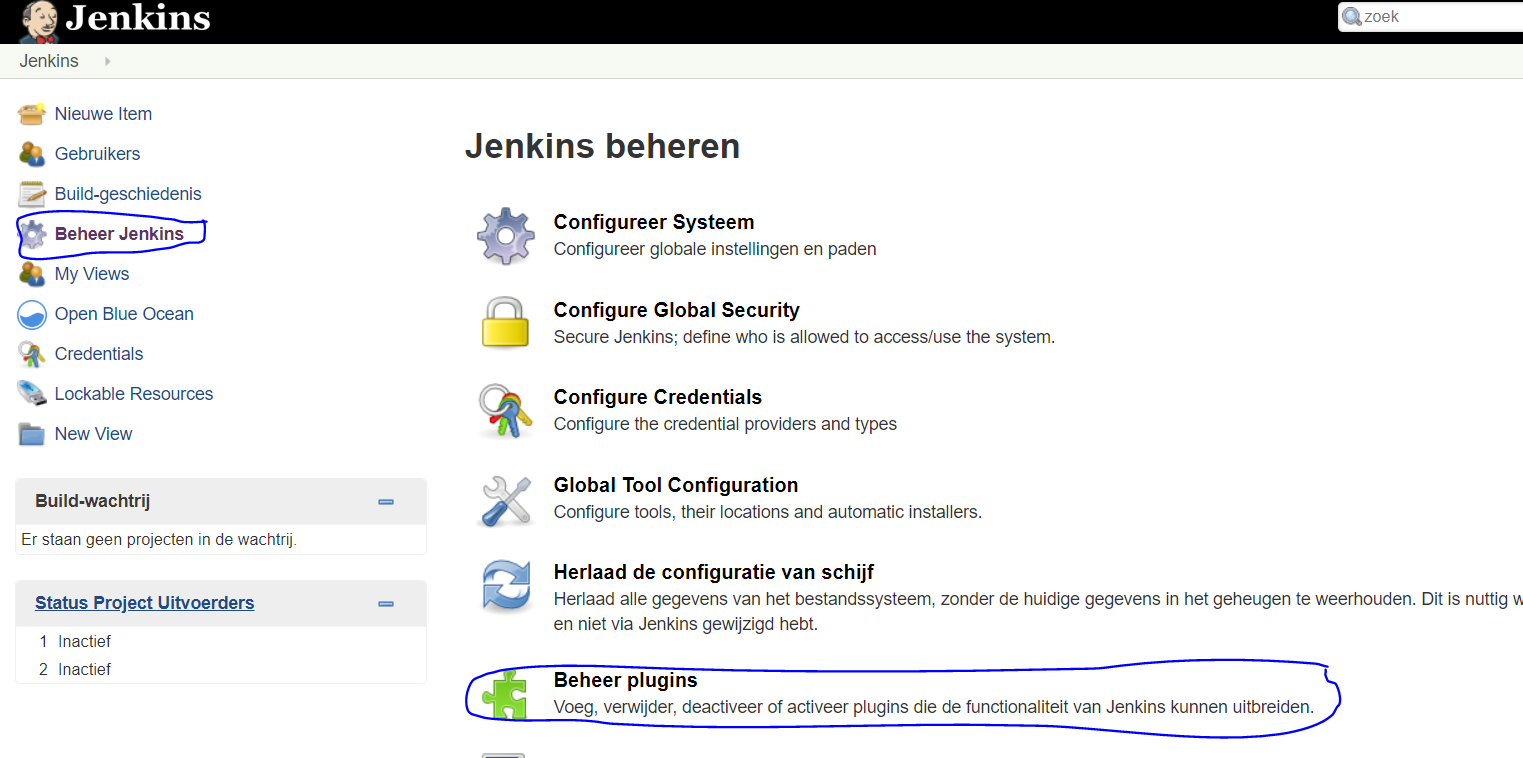
1. Vul op de volgende pagina de gewenste gegevens in



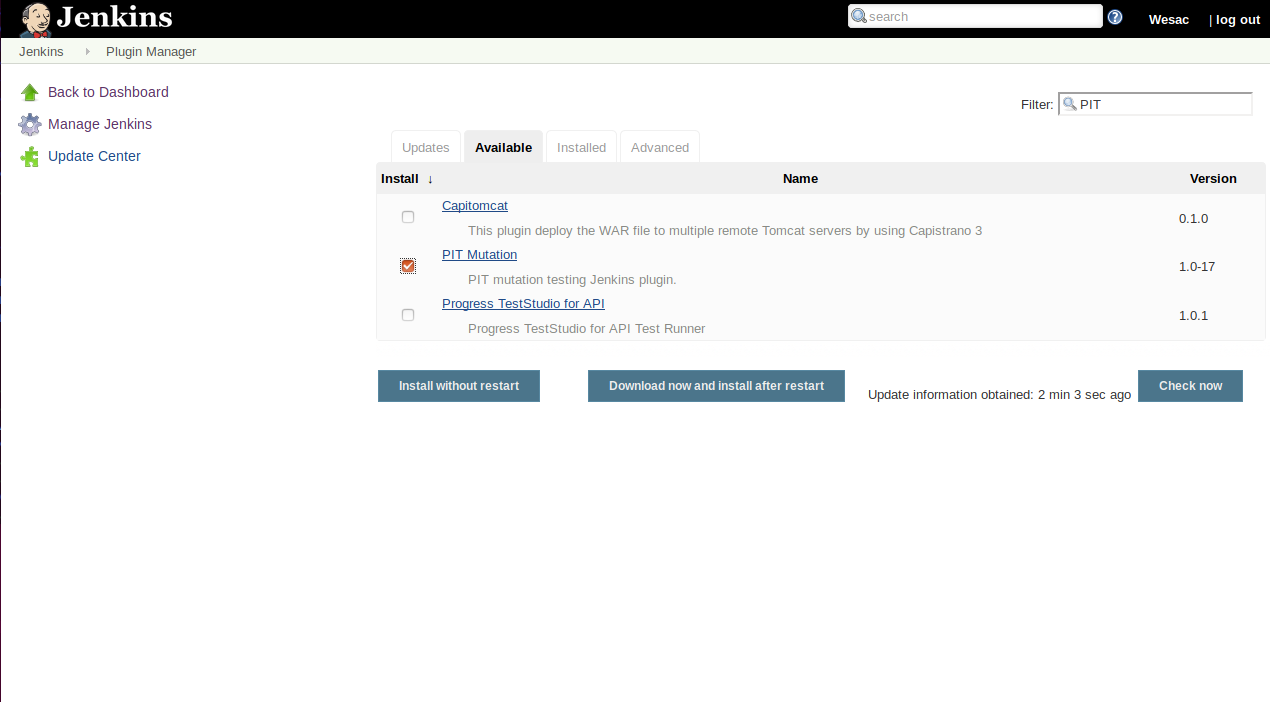
1. Laat op de volgende pagina de url default en druk op finish



1. Klik in jenkins op beheer jenkins en vervolgens op beheer plugins



1. Zoek in het tabje beschikbaar naar de PIT mutation locale en installeer deze(zonder herstart)

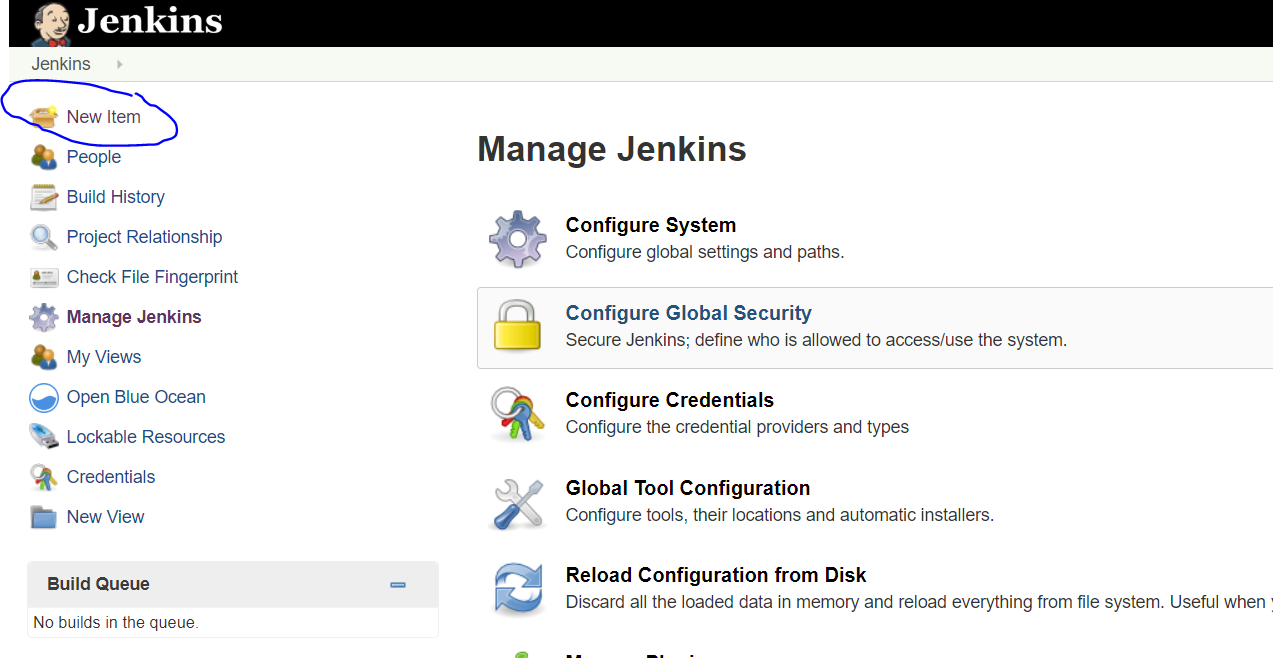


## Voorbereiden GitHub

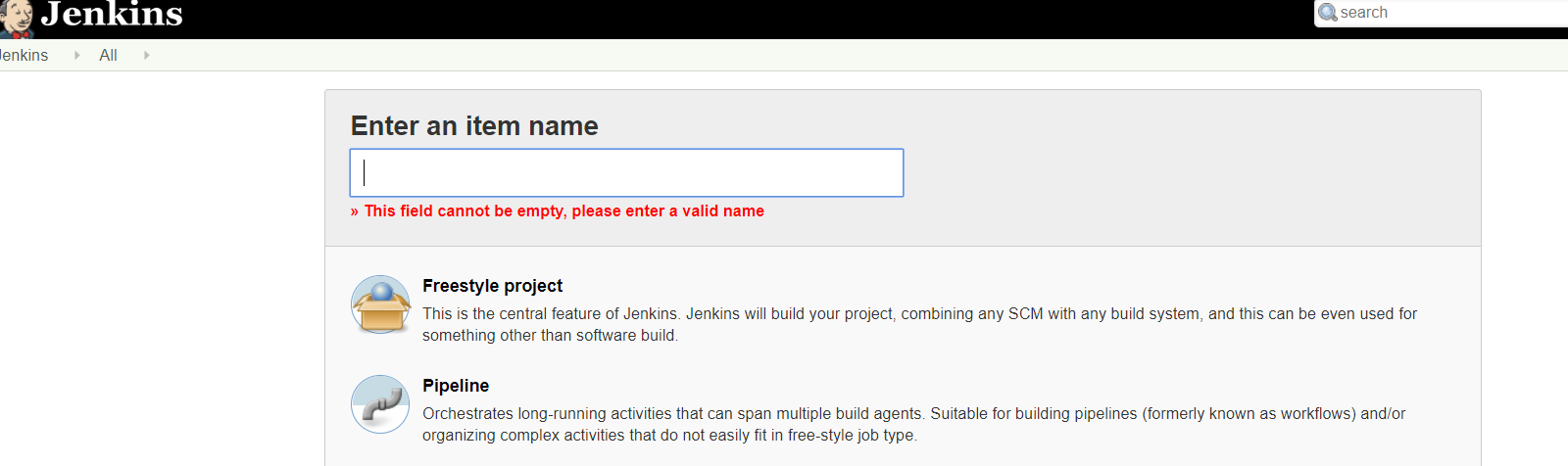
* Fork mijn repo: <https://github.com/EelkeHoffman/daoCD>

## Pipeline bouwen

1. Bouwen daadwerkelijke pipeline



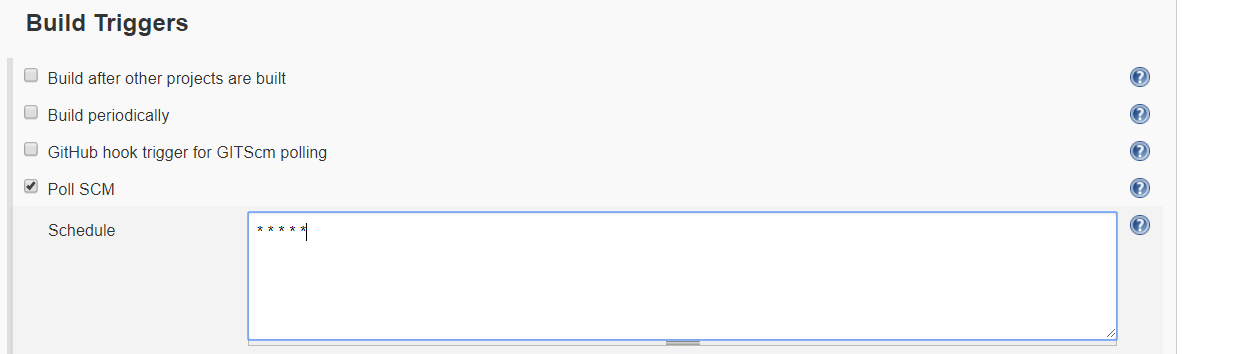
1. Maak in jenkins een nieuw item aan
2. Geef een naam op en selecteer Pipeline



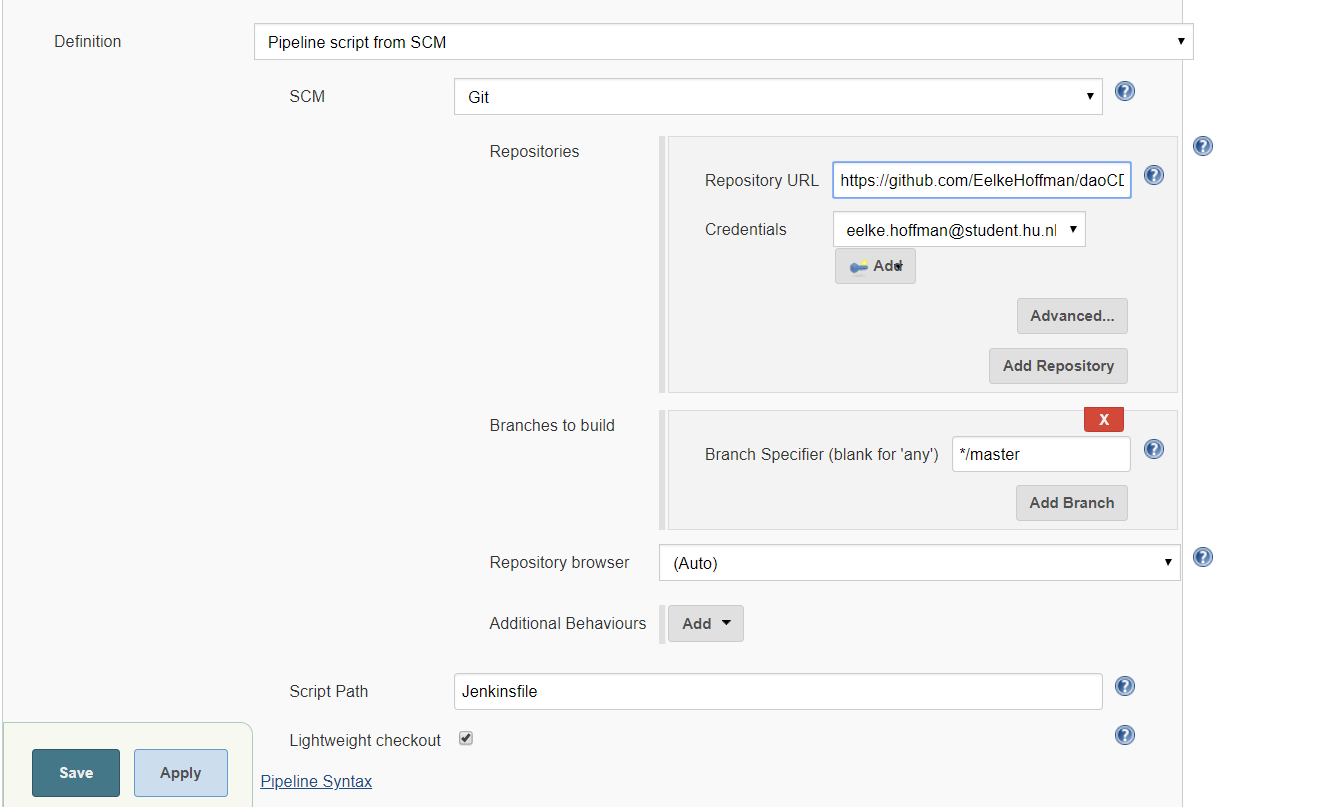
1. Selecteer onder Build Triggers Poll SCM en geef als schedule:

\* \* \* \* \*

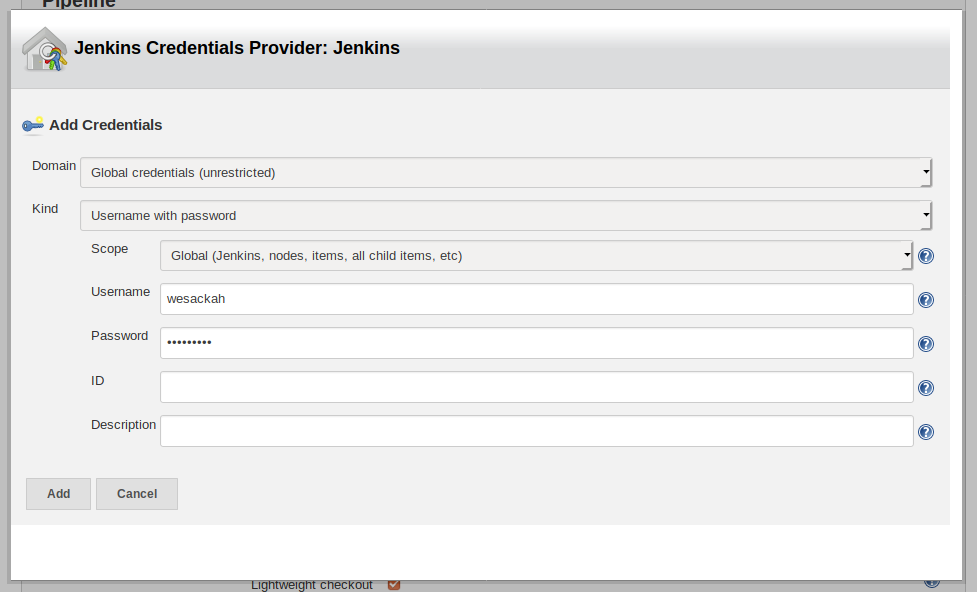
*Nu zal hij een waarschuwing geven dat hij nu elke minuut de repo checkt. Normaal is dit veel te vaak maar voor testen maakt dit niet uit.*



1. Selecteer onder pipeline definition (onderaan de page) pipeline script from SCM.
2. Stel SCM in op Git
3. Zet in de repo url de url naar **jouw** repo (de fork die je eerder maakte).



1. Voeg ook je credentials (GitHub) toe



Nu zal hij voor het pipeline script in jouw repo zoeken naar een bestand dat Jenkinsfile heet (*let op: zonder file extensie*) in dit bestand zal je gaan defineren wat de pipeline gaat doen (<https://jenkins.io/doc/book/pipeline/jenkinsfile/>)

# Definiëren Jenkinsfile

*We raden je aan dit te doen in Github (of een vergelijkbare editor)*

1. In je repo: **Create New File**
2. Het script moet worden opgeslagen als **Jenkinsfile** (zonder extensie)  
   *De stappen van deze file staan achterin het document.*
3. Kopieer onderstaande tekst naar het nieuwe bestand en opslaan

pipeline{

agent{

docker {

image 'maven:3-alpine'

args '-v /root/.m2:/root/.m2'

}

}

stages{

stage('unittests'){

steps{

sh 'mvn test'

}}

stage('mutation tests'){

steps{

sh 'mvn org.pitest:pitest-maven:mutationCoverage'

}

post{

always {

junit 'target/surefire-reports/\*.xml'

}

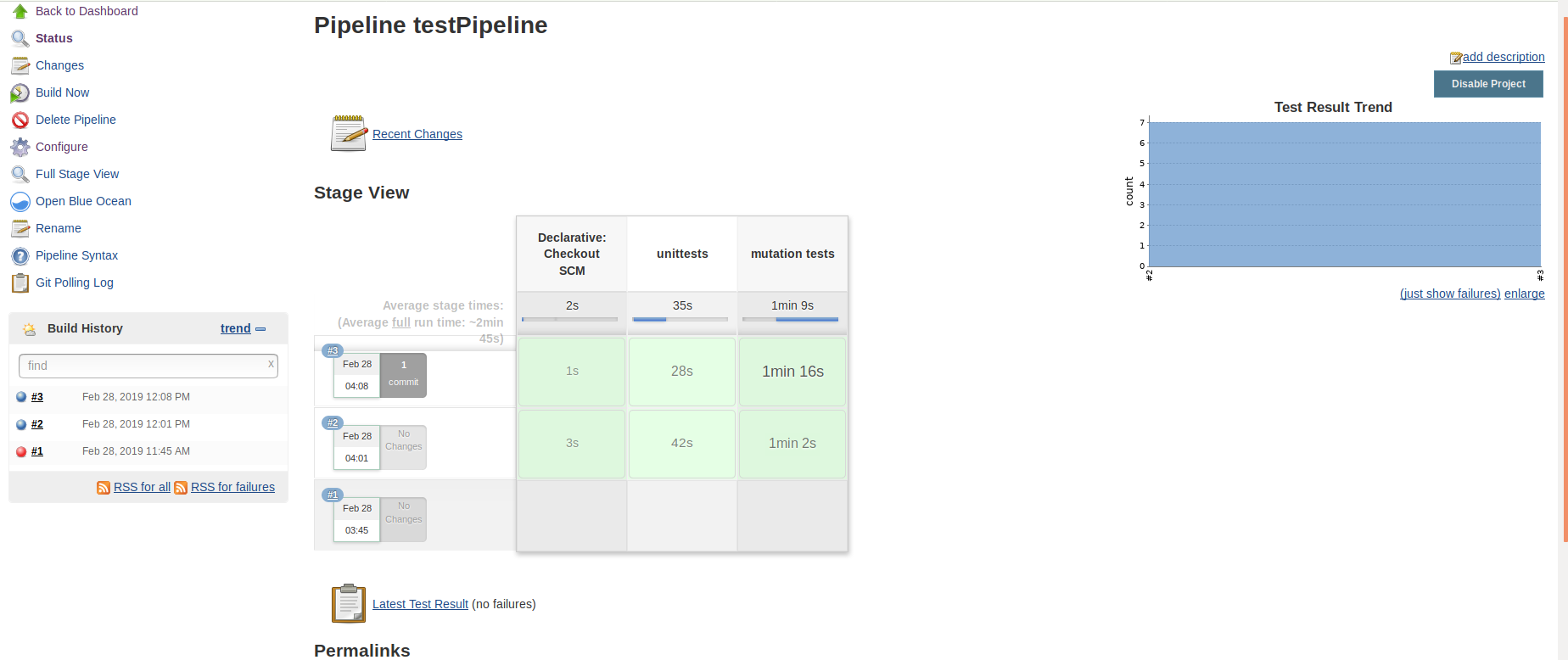
}

}}

}

Zoals je kan zien worden de junit resultaten weg geschreven(junit is een unittest framework)

1. Nu gaan we de pipeline handmatig bouwen(**build now**)



1. Voeg een comment toe aan een willekeurig bestand en kijk of de pipeline opnieuw gebouwd wordt.

# Jenkinsfile extra uitleg

Het eerder gekopieerde bestand bestaat uit de volgende stappen. Dit is extra context en hoeft dus niet nog eens uitgevoerd te worden.

Eerst zet je het skelet van het script op:

Pipeline{

Agent{

//waarop draait deze pipeline(vm/container)

}

Stages{

// welke stappen heeft het script

}

Post{

//welke stappen moeten gebeuren na de stages (vaak stappen mbt reporting)

}

}

Nu voegen we een agent toe waarin hij de unittests zal draaien:

Pipeline{

Agent{

docker {

image 'maven:3-alpine'

args '-v /root/.m2:/root/.m2'

}

}

Stages{

// welke stappen heeft het script

}

Post{

//welke stappen moeten gebeuren na de stages (vaak stappen mbt reporting)

}

}

Zoals je aan het script kunt zien zal dit een agent zijn die draait in een docker container

Nu voegen we de eerste stage toe: het draaien van de unit test:

Pipeline{

Agent{

docker {

image 'maven:3-alpine'

args '-v /root/.m2:/root/.m2'

}

}

Stages{

stage('unittests'){

steps{

sh 'mvn test'

}}

Post{

//welke stappen moeten gebeuren na de stages (vaak stappen mbt reporting)

}

}

Vervolgens voegen we de tweede stage toe, het uitvoeren van de mutatie tests:

Pipeline{

Agent{

docker {

image 'maven:3-alpine'

args '-v /root/.m2:/root/.m2'

}

}

Stages{

stage('unittests'){

steps{

sh 'mvn test'

}}

stage('mutation tests'){

steps{

sh 'mvn org.pitest:pitest-maven:mutationCoverage'

}

Post{

//welke stappen moeten gebeuren na de stages (vaak stappen mbt reporting)

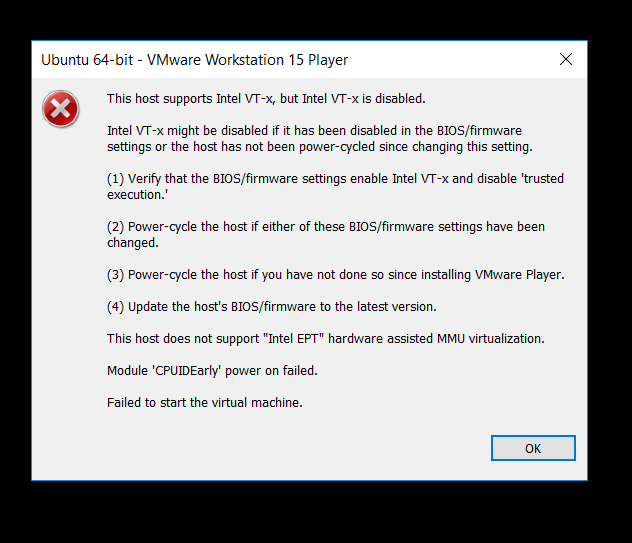
}

}

# Errors

## Vmplayer

Een mogelijke error bij het opstarten van Ubuntu in de VM Player is:



1. De computer opnieuw starten
2. F10 of Del ingedrukt houden om de bios in te gaan.
3. Afhankelijk van je computer, staat er in security of configuration 🡪 Virtual Technology <Disabled>, deze moet je nu aanzetten.
4. Save changes en de computer start op.
5. Start weer VM Player, en probeer het opnieuw.