Testausautomaation ja jatkuvan integraation käyttöönotto

Tiia Heino

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2022

Tietojenkäsittelyn tutkinto

Tiivistelmä

Tekijä(t)
Tiia Heino

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Testausautomaation ja jatkuvan integraation käyttöönotto

Sivu- ja liitesivumäärä
23 + 29

Opinnäytetyössä tarkasteltiin testausautomaation ja jatkuvan integraation hyötyjä, sekä laadittiin toimeksiantajalle käyttöönotto-ohje, jota on mahdollista hyödyntää tulevissa projekteissa.

Tietoperustassa tarkasteltiin testausautomaatiota ja jatkuvaa integraatiota yleisesti sekä teknologioita, jotka otettiin käyttöön tässä projektissa. Nämä teknologiat olivat Robot Framework, Selenium, Jenkins, Docker sekä GitLab.

Empiirisessä osassa selvennettiin lähtötilanne sekä ohjeen tuottaminen. Lähtötilanteen kuvauksessa läpikäytiin toimeksiantaja, sekä heidän testaustilanteensa ja lisäksi työn rajoitteet ja tavoitteet. Ohjeen tuottamisosassa kerrottiin työkalujen valinnasta, ympäristön pystytyksestä, testauksen suunnittelusta sekä testauksesta. Työkalujen valinta sisälsi tietoa mihin työkaluja käytettiin työssä, miksi ne valittiin ja mitä muita vaihtoehtoja oli. Ympäristön pystytysosiossa selvennettiin millainen ympäristö pystytettiin ja mitä riippuvaisuuksia kohdattiin. Testauksen suunnittelusta kerrottiin käymällä läpi tapoja, miten testejä suunniteltiin. Testausosiossa puolestaan käytiin läpi itse testausta ja sitä miten testit rakentuivat.

Lopussa pohdintaosiossa heijastellaan esimerkiksi opinnäytetyön tavoitteiden onnistumista, omaa oppimista sekä työn hyödyllisyyttä.

Asiasanat

Testausautomaatio, testaus, jatkuva integraatio, Robot Framework, Jenkins

Sisällys

1 Johdanto 1
2 Tietoperusta 2
2.1 Testausautomaatio 2
2.2 Jatkuva integraatio 2
2.3 Robot Framework 3
2.4 Selenium 6
2.5 Jenkins 6
2.6 Docker 7
2.7 GitLab 8
3 Empiirinen osa 9
3.1 Lähtötilanteen kuvaus 9
3.2 Ohjeen tuottaminen 9
3.2.1 Työkalujen valinta 9
3.2.2 Ympäristön pystytys 11
3.2.3 Testauksen suunnittelu 11
3.2.4 Testaus 12
4 Pohdinta 19
Lähteet 21
Liitteet 24
Liite 1. Testiautomaation ja jatkuvan integraation käyttöönoton ohje 24

1 Johdanto

Merkittävä osa jokapäiväisestä elämästämme pyörii tavalla tai toisella teknologian ympärillä eikä nykypäivän maailma toimisikaan ilman teknologiaa. Iso osa toiminnasta esimerkiksi infrastruktuurista liikenteeseen on riippuvaista tietokonepohjaisista järjestelmistä. (Sommerville 2016, 3) Kaikki tämä teknologia vaatii myös testausta ja on selvää, että kaiken nykypäivänä käytössä olevan teknologian manuaalinen testaus olisi hidasta ja itseään toistavaa. Kuten muussakin ohjelmistokehityksessä, myös web-sovelluskehityksessä sovelluksen julkaisu on riippuvainen testauksesta.

Työ suoritetaan Saimaan Web-Palvelut Oy:lle, joka tuottaa web-ratkaisuja asiakkaille. Toistaiseksi toimeksiantaja on käyttänyt projekteissaan manuaalitestausta, joka on projektien määrän ja laajuuden kasvaessa hyvin työlästä ja aikaa vievää. Tämän vuoksi testausautomaation käyttöönotto osaksi kehitystyötä on hyvä tehdä ennen kuin projektit ja niiden määrä kasvavat merkittävästi. Toisaalta jos testauksen tarve on pieni, voi automatisointi viedä enemmän aikaa kuin testaus muuten veisi.

Työn tavoitteena on luoda toimeksiantajalle testausautomaation käyttöönoton ohje, jonka avulla toimeksiantaja kykenee omissa projekteissaan hyödyntämään testausautomaatiota. Lisäksi tavoitteena on kasvattaa tietämystä ja osaamista testiautomaatiosta ja jatkuvasta integraatiosta sekä käytetyistä työkaluista.

2 Tietoperusta

2.1 Testausautomaatio

Testausautomaatiota on käytetty jossain muodossa jo useiden vuosikymmenien ajan, mutta nykyiseen muotoonsa se on muovautunut viime vuosikymmenien aikana, kun ohjelmistoprojekteissa on siirrytty vesiputousmallista ketteriin menetelmiin (Axelrod 2018, luku 1.1).

Dustinin, Garrettin ja Gaufin (2009, luku 1.1) mukaan testausautomaatio on osa ohjelmistokehitysprosessia. Testausautomaation käyttö ei poista tarvetta manuaalitestaajien asiantuntemukselle, ja testausautomaatio sekä manuaalitestaus kulkevatkin käsikädessä.

Hyödyntämällä testausautomaatiota on mahdollista vähentää ohjelmistotestaukseen kuluvaa aikaa ja rahaa, parantaa ohjelmiston laatua ja testauksen laajuutta sekä suorittaa tehtäviä, joita on vaikeaa saavuttaa manuaalistestauksella, kuten muistivuotojen havaitseminen (Dustin ym. 2009, luku 1.2).

Yleisimmät syyt testausautomaation epäonnistumiselle ovat ajan ja budjetin puute, työkalujen yhteensopimattomuus sekä asiantuntemuksen puute (Dustin ym. 2009, luku 1.4). Dustin, Garrett ja Gauff (2009, luku 2.1) kertovat, että luodakseen onnistuneen testausautomaation on otettava huomioon seuraavat kuusi avainkohtaa: tunne vaatimukset, kehitä automatisoidun testauksen strategia, testaa automatisoitu testauskehys, seuraa edistymistä ja säädä asianmukaisesti, toteuta testausautomaatioprosessit sekä valitse projektiin oikeat henkilöt.

2.2 Jatkuva integraatio

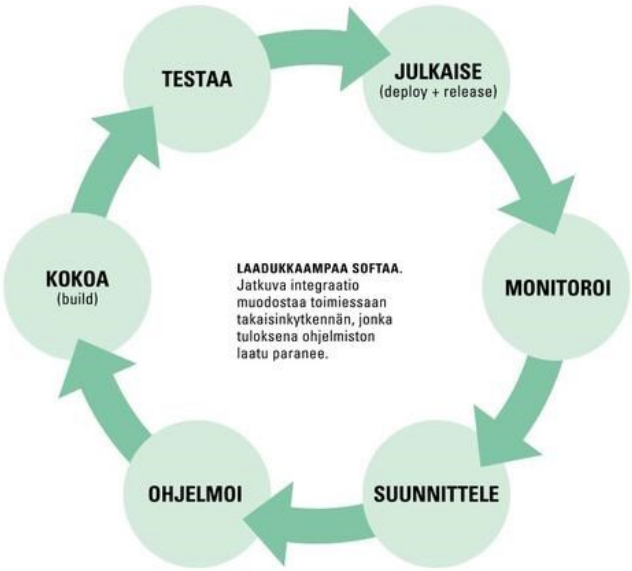
Jatkuvalla integraatiolla tarkoitetaan koodin jatkuvaa, esimerkiksi päivittäistä, integroimista ja yhteen liittämistä päätietovarastoon. Jatkuvan integraation avulla tuotteita voidaan julkaista tiiviimmässä tahdissa, löytää ohjelmointivirheet nopeammin ja parantaa ohjelmiston laatua.

Kun ohjelmoija integroi oman koodinsa päätietovarastoon vain harvoin, on koodi jo todennäköisesti muuttunut muidenkin ohjelmoijien integroidessa oman koodinsa. Tämä aiheuttaa virheitä ja ongelmia, joiden ratkomisessa kuluu ylimääräistä aikaa.

Jatkuvassa integraatiossa koodia liitetään päätietovarastoon jatkuvasti. (AWS s.a.) Versionhallintaohjelmistot ja -palvelut mahdollistavat muutoshistorian tarkastelun ja

ylläpidon. Toinen keskeinen asia on automatisoitu kasaaminen eli toimivan ohjelmistoversion pystyttämiseen vaadittujen vaiheiden automatisoidun suorittamisen. Näitä vaiheita voivat olla esimerkiksi testien pyörittäminen, lähdekoodin kääntäminen sekä riippuvuuksien asentaminen.

Jatkuvaan integraatioon liittyy usein myös jatkuva toimittaminen sekä jatkuva julkaiseminen. Jatkuva toimittaminen tarkoittaa julkaisuprosessin automatisointia eli uusi versio voidaan siirtää jakeluun vaikkapa nappia painamalla. Jatkuvan julkaisun kohdalla napin painalluskin on automatisoitu, eli kun koodi on päivitetty tietovaraston päähaaraan, se etenee jakeluun automaattisesti ilman ihmisen päätöstä. Nämä kolme menetelmää yhdessä muiden ohjelmistotuotannon vaiheiden kanssa muodostavat jatkuvan integraation ja jatkuvan toimittamisen sekä julkaisemisen elinkaaren, joka on nähtävissä kuvassa 1. (Mikkonen 2019.)



Kuva 1. Jatkuvan integraation ja jatkuvan toimittamisen sekä julkaisemisen elinkaari (Mikkonen 2019)

2.3 Robot Framework

Robot Framework on Python-ohjelmointikielellä kirjoitettu avainsanapohjainen testausautomaatiokehys, jonka versio 2 julkaistiin avoimena lähdekoodina vuonna 2008 (Bisht 2013, luku 1). Sitä käytetään useimmiten hyväksymistestivetoisessa sekä käyttäytymislähtöisessä ohjelmistokehityksessä (Molina 2021, luku 3.2).

Robot Frameworkin syntaksi on selkeää ja avainsanapohjaisuutensa vuoksi helposti luettavaa. Robot Framework on toimii hyvin muiden työkalujen kanssa ja se on laajennettavissa laajan kirjaston ja työkaluvalikoiman avulla. (Robot Framework s.a.)

Kuvassa 2 on esimerkki Robot Frameworkin testitapauksesta, jossa testataan toimiiko kirjautuminen olemassa olevilla tunnuksilla. Testi alkaa selaimen avaamisella kirjautumissivulle, jonka jälkeen sivulle syötetään testitapauksesta löytyvät käyttäjätunnukset eli ”demo” ja ”mode”. Tämän jälkeen käyttäjätunnukset lähetetään ja jos syötetyt tunnukset löytyvät kannasta, etusivun tulisi aueta selaimessa. Jos sivu aukeaa onnistuneesti, selain sulkeutuu ja testi ilmoittaa onnistumisen testiraportissa ja -lokissa. Kuvissa 3 ja 4 on esimerkit testiraportista ja -lokista, jotka luodaan automaattisesti kun testisarjaa ajetaan. Kuvien tapauksessa kaikki testit ovat menneet läpi, mutta mikäli näin ei olisi, dokumentit muuttuisivat punaiseksi ja testien epäonnistumisen syy olisi luettavissa dokumenteista. Testilokin ja -raportin avulla nähdään myös esimerkiksi se, kuinka paljon kunkin testin ajamiseen on mennyt aikaa. Testiraportti sisältää yleiskatsauksen testien ajon tuloksista, kun taas testiloki sisältää yksityiskohtaista tietoa testitapauksien ajosta (Robot Framework 2016).

```
*** Settings ***
Documentation     A test suite with a single test for valid login.
...
...               This test has a workflow that is created using keywords in
...               the imported resource file.
Resource          resource.txt

*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

Kuva 2. Testitapaus (Robot Framework s.a.)

Login Tests Test Report

file:///Users/tkairi/Coding/webdemo/report.html#totals?critical

LOG

Login Tests Test Report

Generated
20160127 17:47:33 GMT +03:00
23 minutes 56 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20160127 17:47:22.613
End Time:	20160127 17:47:33.696
Elapsed Time:	00:00:11.083
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	8	8	0	00:00:08	
All Tests	8	8	0	00:00:08	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Login Tests	8	8	0	00:00:11	
Login Tests . Gherkin Login	1	1	0	00:00:03	
Login Tests . Invalid Login	6	6	0	00:00:05	
Login Tests . Valid Login	1	1	0	00:00:03	

Test Details

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests

Status: 8 total, 8 passed, 0 failed

Total Time: 00:00:08.068

Name	Documentation	Tags	Crit.	Status	Message	Elapsed	Start / End
Login Tests . Gherkin Login . Valid Login			yes	PASS		00:00:03.273	20160127 17:47:22.815 20160127 17:47:26.088

Kuva 3. Testiraportti (Robot Framework s.a.)

Login Tests Test Log

file:///Users/tkairi/Coding/webdemo/log.html

REPORT

Login Tests Test Log

Generated
20160127 17:47:33 GMT +03:00
8 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	8	8	0	00:00:08	
All Tests	8	8	0	00:00:08	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Login Tests	8	8	0	00:00:11	
Login Tests . Gherkin Login	1	1	0	00:00:03	
Login Tests . Invalid Login	6	6	0	00:00:05	
Login Tests . Valid Login	1	1	0	00:00:03	

Test Execution Log

SUITE Login Tests 00:00:11.083

Full Name:	Login Tests
Source:	/Users/tkairi/Coding/webdemo/login_tests
Start / End / Elapsed:	20160127 17:47:22.613 / 20160127 17:47:33.696 / 00:00:11.083
Status:	8 critical test, 8 passed, 0 failed 8 test total, 8 passed, 0 failed

SUITE Gherkin Login 00:00:03.424

SUITE Invalid Login 00:00:04.511

SUITE Valid Login 00:00:03.082

Full Name:	Login Tests.Valid Login
Documentation:	A test suite with a single test for valid login. This test has a workflow that is created using keywords in the imported resource file.
Source:	/Users/tkairi/Coding/webdemo/login_tests/valid_login.robot
Start / End / Elapsed:	20160127 17:47:30.609 / 20160127 17:47:33.691 / 00:00:03.082

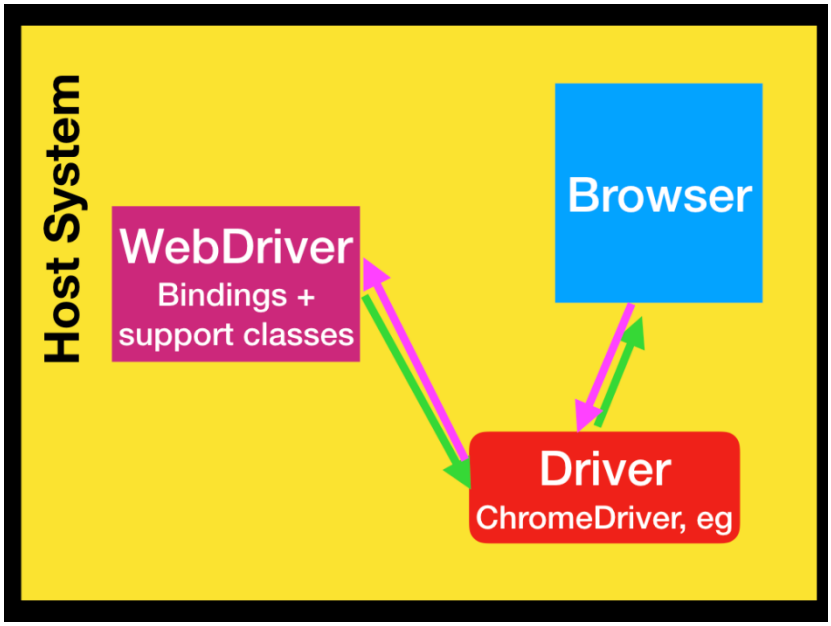
Kuva 4. Testiloki (Robot Framework s.a.)

2.4 Selenium

Selenium on automatisoitu avoimen lähdekoodin testikehys, jota käytetään pääasiassa verkkosovellusten testaamiseen.

Vuonna 2004 alkunsa saanut Selenium on nykypäivänä sarja työkaluja, joilla on kullakin omat tehtävänsä. Selenium koostuu siis työkaluista nimeltä Selenium IDE, Selenium RC, Selenium WebDriver ja Selenium Grid. (Ramya & Sindhura 2017, 2.)

Selenium WebDriverin tehtävä on vastaanottaa komentoa ja suorittaa ne selaimessa (Selenium s.a.). WebDriver siirtää komentoja selaimelle selainkohtaisen selainajurin, kuten GeckoDriver ja ChromeDriver, läpi ja tällöin testatessa WebDriver hoitaa kaiken painikkeiden painelun ja tekstikenttiin kirjoittamisen. Tämä kuvassa 5 nähtävissä oleva prosessi on suoraa viestintää, mutta WebDriveria on mahdollista käyttää myös etäviestinnällä niin, että WebDriverin ja selainajurin välissä on Selenium Server, Selenium Grid tai Remote WebDriver. (Selenium 2021.)



Kuva 5. Suora viestintä (Selenium 2021)

2.5 Jenkins

Jenkins on avoimen lähdekoodin automaatiopalvelin ja on alunperin luotu jatkuvan integraation tarpeisiin, mutta nykypäivänä Jenkins huolehtii myös jatkuvasta julkaistusta ja jatkuvasta toimituksesta. Jenkins tukee useita versionhallintajärjestelmiä ja siihen on olemassa laajennuksia esimerkiksi käyttöliittymiin ja Android ja iOS-kehitykseen.

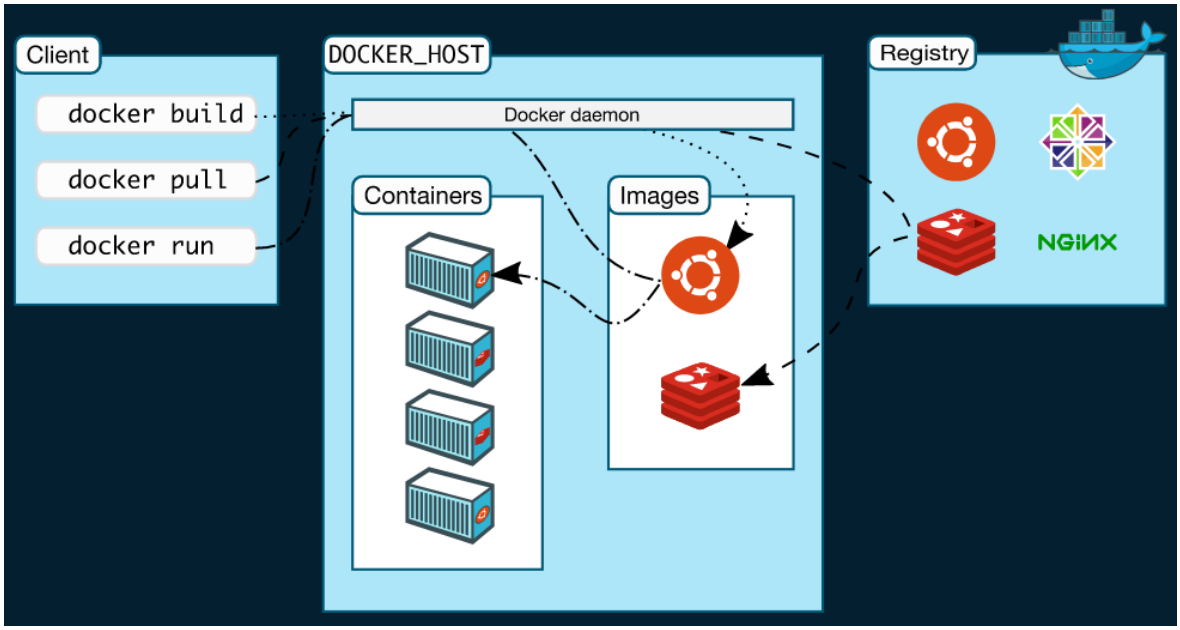
Jenkins on muun muassa helpon käyttöliittymänsä ja hyvien laajennusmahdollisuuksiensa vuoksi yksi suosituimmista automaatiopalvelimista. (Soni 2017, luku 2.1.)

Jenkinsfile on tiedosto, joka sisältää Jenkins pipelinen eli putken määritelmän. Jenkins putki on sarja laajennuksia, jotka mahdollistavat jatkuvan integraation putken luomisen Jenkinsiin. Putken syntaksi voi olla deklaratiivinen tai skriptattu. (Jenkins s.a.) Deklaratiivinen ja skriptattu putki eroavat toisistaan koodillisesti ja deklaratiivisissa putkissa stages jaetaan steps, toisin kuin skriptatussa (Kszczanowicz & Pomianowski 2022.).

2.6 Docker

Docker on avoimeen lähdekoodin perustuva alusta jossa sovelluksia voidaan ajaa, kehittää ja toimittaa kontitettussa ympäristössä. Alustan avulla voidaan pystyttää standardoitu kehitys- tai ajoympäristö eli kontti, joka on erillään muusta infrastruktuurista.

Dockerin arkkitehtuuri on asiakas-palvelin-tyyppinen, jolloin Docker client eli asiakas mahdollistaa erinäisten Docker komentojen ajamisen, joita Docker daemon vastaanottaa ja toimii sitten komentojen mukaan, esimerkiksi hakee imagen registrystä eli rekisteristä. Tämä prosessi on nähtävissä kuvassa 6. Docker daemon on taustaprosessi, joka hallinnoi muun muassa imageja ja kontteja (Chen 2020). Docker rekisteri on paikka, johon Docker tallentaa ja josta se hakee imageja. Oletuksena Docker käyttää imagen hakuun Dockerhub
-rekisteriä, joka on julkinen rekisteri. Halutessaan on mahdollista pystyttää Docker käyttämään yksityistä rekisteriä.

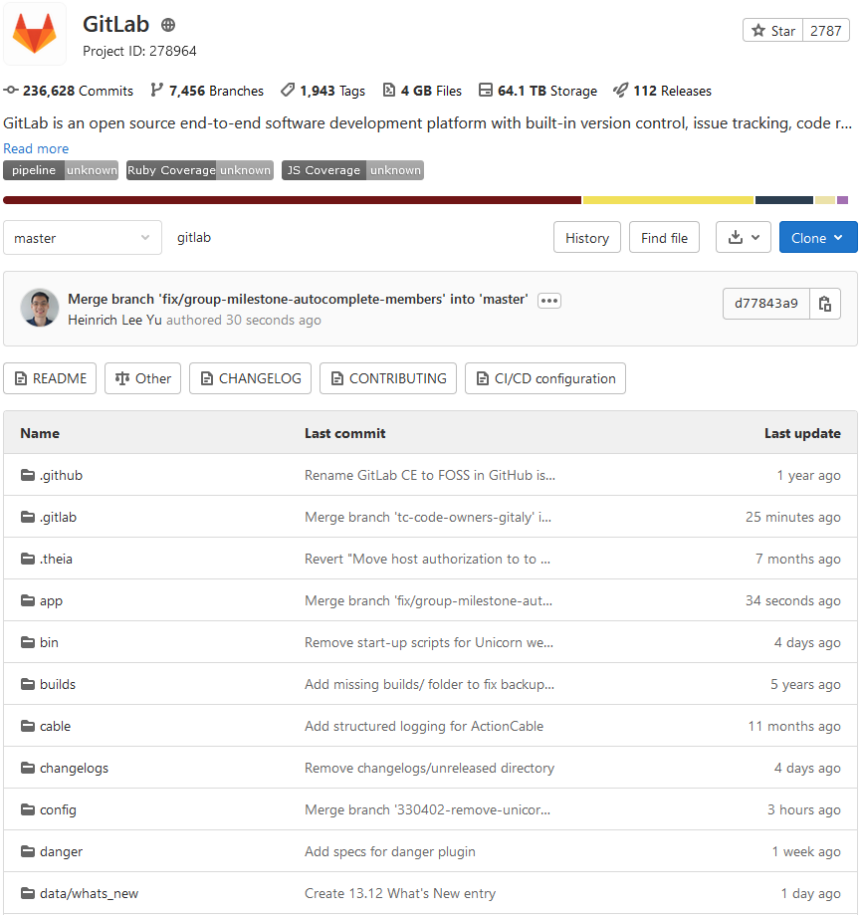


Kuva 6. Dockerin arkkitehtuuri (Docker s.a. a)

Kontteihin asennettavat asiat voidaan määrittää Docker imagessa, joka toimii ohjeena kontin luontia varten. Image luodaan kirjottamalla Dockerfile tiedosto, johon määritetään ympäristö ja mitä työkaluja konttiin halutaan. Imagea luodessa on myös mahdollista hyödyntää jo olemassa olevia image-tiedostoja ja tehdä muutoksia tarvittaessa. Dockerin suosion vuoksi monesta ohjelmasta on saatavilla valmiita image-tiedostoja jotka ovat avoimessa käytössä kaikille. Tässä projektissa käytettiin valmista Jenkins imagea, joka on saatavilla Dockerhubissa. Valmiin imagen sijaan on mahdollista käyttää myös itse räätälöityä imagea, joka pohjautuu Dockerhubissa saatavilla olevaan imageen. (Docker s.a. a)

2.7 GitLab

GitLab on vuonna 2011 julkaistu verkkopohjainen koodinhallinta- ja ongelmanseuranta-alusta. Sen ominaisuuksiin kuuluu muun muassa ongelmien ja projektien hallinta, jatkuva integraatio ja julkaisu, koodin tarkastelu sekä sen muokkaus. Palvelua on mahdollista käyttää GitLabin ylläpitämällä verkossa olevalla alustalla, josta esimerkki kuvassa 7, tai ylläpitää omaa instanssia. (O'Grady 2018, luku 1.)



Kuva 7. GitLabin tietovarastonäkymä

3 Empiirinen osa

3.1 Lähtötilanteen kuvaus

Työn toimeksiantaja on web-ratkaisuja, esimerkiksi verkkokauppoja ja organisaatioiden kotisivuja, tarjoava yritys Saimaan Web-Palvelut Oy. Yritys on tähän asti käyttänyt tuotteidensa testaamiseen ainoastaan manuaalitestausta. Yritys toimii pienellä työntekijämäärällä, joten kaikki testaamiseen menevä aika on muusta työstä pois. Tästä syystä toimeksiantaja halusi ottaa selvää testausautomaatiosta ja jatkuvasta integraatiosta, sekä siitä miten ne voisi ottaa käyttöön manuaalitestauksen rinnalle.

Työlle ei ole asetettu rajoitteita, mutta suurin rajoittava tekijä on luultavasti ohjeen laatijan osaaminen tämän aihepiirin osalta.

Lopputuotoksen tulee olla selkeä ja informatiivinen toimeksiantajan omissa projekteissa hyödynnettävissä oleva ohje, jonka avulla toimeksiantaja kykenee kehittämään testaustaan ja testausautomaatioprosessejaan. Ohjeessa esiteltyjen toimenpiteiden täytyy olla suoritettavissa asiakkaan omassa ympäristössä. Toimeksiantaja antaa oman arvionsa ohjeen käytettävyydestä ja laadusta.

3.2 Ohjeen tuottaminen

Jotta ohjeen pystyi kirjoittamaan, oli ympäristön pystyttäminen ja testien laatiminen tehtävä myös itse. Prosessi lähti käyntiin tutustumalla tuotteeseen, eli toimeksiantajan pystyttämään verkkokauppa-pohjaan, jossa on samanlaisia ominaisuuksia kuin heidän oikeissa tuotteissaan.

3.2.1 Työkalujen valinta

Seuraavassa vaiheessa valittiin työkalut. Toimeksiantaja ei asettanut rajoitteita työkalujen suhteen, mutta toivoi työkaluja, jotka olisivat helposti lähestyttäviä. Työkaluiksi valikoituivat Robot Framework, Jenkins, Docker, Selenium sekä GitLab. Nämä työkalut valikoituivat erilaisista syistä, mutta kaikille yhteistä on se, että ne perustuvat avoimeen lähdekoodiin ja ovat saatavilla ilmaiseksi tämän kokoluokan yrityksille. Lisäksi ne toimivat hyvin yhdessä.

Testisarjan kirjoittamiseen valikoitui avainsanapohjainen testiautomaatiokehys Robot Framework. Helposti lähestyttävän ja ymmärrettävän syntaksin ja hyvien laajentamismahdollisuuksien lisäksi aiempi kokemus Robot Frameworkistä auttoi valinnassa. Muita suosittuja testiautomaatiokehyksiä ovat esimerkiksi Selenium ja

Cypress. Selenium olisi ollut hyvä valinta, sillä Selenium WebDriver oli valittu projektiin. Selenium on kuitenkin rajoitetumpi Robot Frameworkiin verrattuna ja esimerkiksi omien testitapausten luominen ei ole mahdollista eikä se sisällä sisäänrakennettuja kirjastoja (Geeks for Geeks 2022). Cypress puolestaan sopii ainoastaan selainpohjaisten tuotteiden testaamiseen, joten jos tarve toisenlaiselle testaamiselle syntyisi, pitäisi valita uusi työkalu. Lisäksi Cypress tukee tällä hetkellä ainoastaan Chrome-selainta. (Appsurify 2021.)

Automaatiopalvelin Jenkins valittiin projektiin ajamaan testejä automatisoidusti. Tässä tapauksessa testit ajettiin, kun tietovarastoon tehtiin muutoksia. Muita jatkuvan integraation palveluita ovat esimerkiksi Azure DevOps sekä GitLabin CI/CD palvelu. Azure DevOps olisi tarjonnut lisäksi mahdollisuuden versionhallintaan sekä käännöspakettien hallintaan, mutta koska ilmaisella käyttötasolla käyttäjien määrä on rajoitettu ja CI/CD putkien määrä sekä käyttöaika olivat rajalliset, ei tätä palvelua valittu (Azure s.a.). GitLabin CI/CD palvelu olisi ollut hyvin yhteentoimiva toimeksiantajan nykyisen versionhallinnan kanssa, mutta myös tässä palvelussa ilmaisessa versiossa oli käyttörajoitteita (GitLab s.a.). Jenkinsin suosion lisäksi nämä syyt vaikuttivat valintaan, sillä Jenkins on täysin ilmainen ja rajoitteeton.

Docker valittiin projektiin Jenkinsin vaivattomampaa pystyttämistä varten käyttäen Dockerin tarjoamaa Jenkins imagea. Täten voitiin välttää fyysisen koneen tarve. Docker valikoitui työkaluksi aiemman käyttökokemuksen sekä yrityksen koon vuoksi, sillä Dockeria on mahdollista käyttää ilmaiseksi henkilökohtaisella lisenssillä (Docker s.a. b). Vaihtoehtoinen palvelu olisi ollut Podman, jonka vahvuuksia ovat avoin lähdekoodi sekä maksuttomuus myös yrityskäytössä (Podman s.a.). Toiminnallisuuksiltaan se on hyvin samankaltainen kuin Docker, mutta Podmanin arkkitehtuuri ei sisällä daemonia. Tällöin systemdin käyttö olisi tarpeen, mutta koska toimeksiantaja käyttää Windowsia, sen käyttöönotto olisi liian monimutkaista verrattuna siihen, että valitsisi projektiin mukaan Dockerin. (Figueiredo & Gamela 2021.) Systemd on sarja Linux-järjestelmän perusrakennuspalikoita, joka tarjoaa järjestelmä- ja palveluhallinnan (Systemd 2022).

Testattavana tuotteena oli web-sivusto, joten mukaan valikoitui Selenium ja tarkemmin Selenium WebDriver, josta myös oli aiempaa kokemusta. Selenium WebDriverin avulla automatisoitiin painikkeiden painelua, tekstikenttiin kirjoittelua ja muuta toimintaa, jota web-sivuston testaaminen vaatii. Yksi syy valintaan oli se, että Robot Frameworkillä on olemassa SeleniumLibrary, joka käyttää Selenium WebDriveriä selaimen kontrolloimiseen. Myös Selenium WebDriverin kohdalla Cypress olisi ollut vaihtoehto, mutta edellä mainittujen selainrajoitteiden vuoksi se osoittautui huonoksi vaihtoehdoksi.

Toinen vaihtoehtoinen työkalu olisi ollut WebDriverIO, mutta jos WebDriverIO:n olisi valinnut olisi myös Robot Framework poistaa valituista työkaluista.

Koodin versionhallintaan valittiin työkaluksi GitLab, joka oli jo valmiiksi toimeksiantajan käytössä. Vaihtoehtoja löytyy useita ja suosituin niistä on varmasti GitHub. Sekä GitLabilla että GitHubilla on ilmaiset versiot, mutta GitHubin ilmaisversio on tarkoitettu avoimen lähdekoodin projekteille (GitHub s.a.).

3.2.2 Ympäristön pystytys

Ympäristön pysytyksessä otettiin huomioon asiakkaan nykyinen käyttöjärjestelmä, mutta olettamus oli ohjetta laatiessa, että mitään tarvittavia työkaluja ja laajennuksia ei ollut valmiiksi asennettuna. Testiympäristöä ei ole aina helppoa yksiselitteisesti jakaa tiettyyn kategoriaan, mutta yksi jaottelu on jakaa testiympäristötyypit kymmeneen luokkaan. Nämä ovat yksikkötestaus, järjestelmäintegraatiotestaus, kaaostestaus, suorituskykytestaus, turvallisuustestaus, laadunvarmistustestaus, alfatestaus, beetatestaus, regressiotestaus sekä hyväksymistestaus (Reynolds 2022). Suurissa yrityksissä ja projekteissa kaikki edellä mainitut testiympäristötyypit ovat olennaisia ja yhdentyyppistä testiympäristöä saattaa olla montakin kappaletta. Tämän projektin testiympäristö lienee lähimpänä laadunvarmistustestausta, sillä testit keskittyivät ominaisuuksien toimivuuden testaamiseen.

Ympäristön pystytys sisälsi kaikkien työkalujen ja muiden tarvittavien laajennuksien ja ohjelmien asennuksen. Joidenkin työkalujen kohdalla esiintyi riippuvuuksia, kuten Robot Frameworkin kohdalla Python. Robot Framework tarvitsee toimiakseen Pythonin, joten myös se oli asennettava. Dockerin toimivuuden varmistaminen Windows-ympäristössä vaati WSL2:n asennuksen. Jenkins noodit vaativat toimiakseen Javan ja koska OpenJDK on ilmainen myös yrityskäytössä, se valittiin Oraclen Javan sijasta.

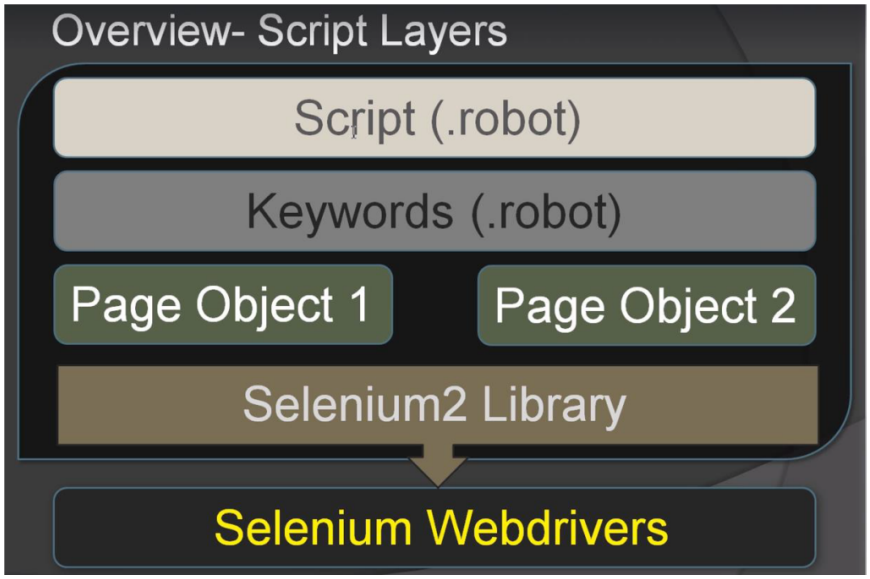
3.2.3 Testauksen suunnittelu

Ympäristön pystytyksen jälkeen oli aika suunnitella testausta tarkemmin. Alkuun oli syytä tarkastella tuotetta tarkemmin ja asettua käyttäjän asemaan, jotta selviäisi millaisia tapahtumia sivulla on. Seuraavaksi näitä mahdollisesti testattavia tapahtumia ja elementtejä kirjattiin ylös. Testauksen rajaaminen oli hieman haastavaa, sillä testattavaa löytyi melko paljon ja ilman kokemusta oli hankalaa määritellä mitkä testit ovat tärkeitä ja mitkä eivät. Testattaviksi tapahtumiksi päätyivät esimerkiksi hakukentän avulla haetun tuotteen lisääminen koriin, selaamalla haetun tuotteen lisääminen koriin sekä kirjautuminen. Testeihin valikoituivat tapahtumat, jotka koettiin oleellisimmaksi toimeksiantajan omien projektien kannalta.

Testien suunnittelun seuraavassa vaiheessa testitapausten vaiheita kirjattiin ylös. Esimerkiksi kirjautumistapahtumassa vaiheita on selaimen avaaminen, halutulle sivustolle meneminen, kirjautumissivulle meneminen, tunnuksen ja salasanan syöttö, kirjautumispainikkeen painaminen sekä kirjautumistapahtuman onnistumisen varmentaminen oikean sivun auetessa.

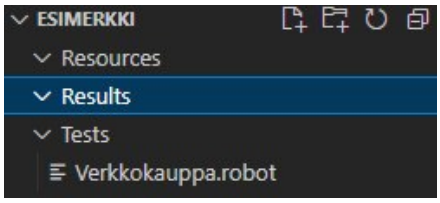
3.2.4 Testaus

Robot Framework testisarja koostuu kerroksista, jotka kommunikoivat toisilleen. Ensimmäisenä on skriptitiedosto, joka kommunikoi keywords-, eli avainsanatiedostoille, jotka puolestaan kommunikoivat page object-, eli sivun elementti -tiedostoille. Sivun elementti -tiedostot kommunikoivat SeleniumLibrarylle, joka kommunikoi Seleniumin selainajureiden kanssa. Tämä rakenne on nähtävissä kuvassa 8.

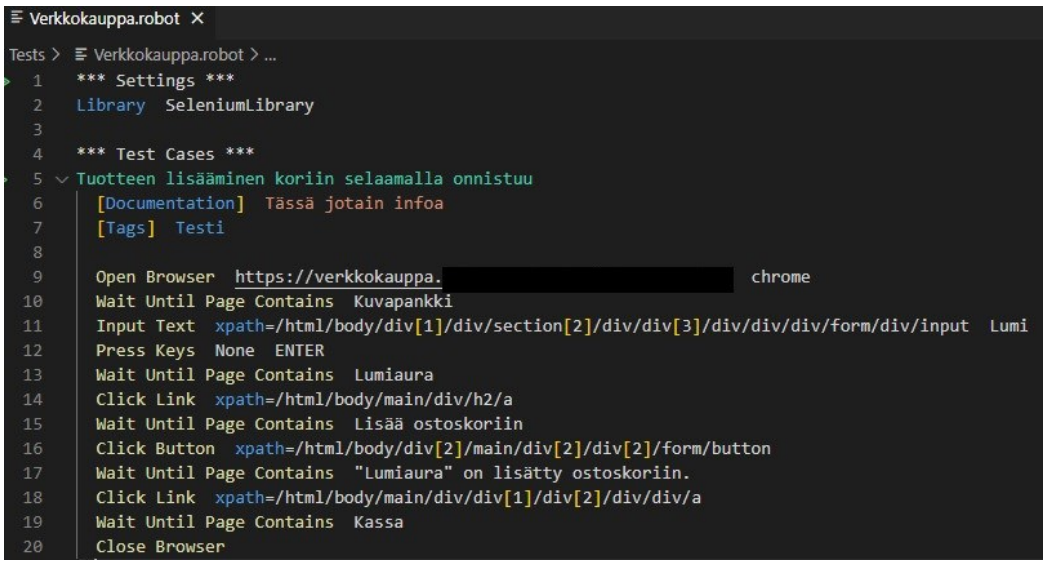


Kuva 8. Robot Frameworkin skriptikerrokset (Channenahalli Ramachandraiah 2018)

Seuraavaksi aloitettiin testien kirjoittaminen. Etenkin tässä vaiheessa oli paljon hyötyä aiemmin käydystä Udemyn tarjoamasta Bryan Lambin Robot Framework-kurssista. Testisarjan luominen lähti hyvin yksinkertaisella tiedostorakenteella ja parilla testitapauksella. Tässä tilanteessa koko koodi sijaitsi yhdessä tiedostossa eli projektin ajettavassa tiedostossa, jota kutsuttiin nimellä Verkkokauppa.robot. Testien kirjoitusvaihetta mallintamaan otettiin kuvakaappauksia tiedostorakenteesta ja koodista testisarjassa, jossa on vain yksi testi. Alkuvaiheen tiedostorakenne on nähtävissä kuvassa 9 ja Verkkokauppa.robot -tiedoston sisältö kuvassa 10.



Kuva 9. Tiedostorakenne alussa



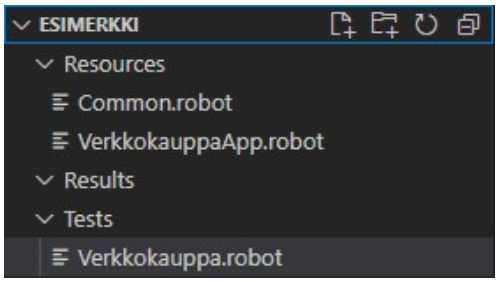
Kuva 10. Verkkokauppa.robot alkutilanne

Kuten voi nähdä kuvasta 11. seuraavaksi jaetaan testitapaus avainsanoihin. Avainsanoja ovat siis esimerkiksi Aloita Testi ja Etsi Tuotteita.

```
Verkkokauppa.robot ×
Tests > Verkkokauppa.robot > Tuotteen lisääminen koriin selaamalla onnistuu
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Test Cases ***
5  Tuotteen lisääminen koriin selaamalla onnistuu
6    [Documentation]  Tässä jotain infoa
7    [Tags]  Testi
8    Aloita Testi
9    Etsi Tuotteita
10   Valitse Tuote Hakutuloksista
11   Lisää Tuote Ostoskoriin
12   Siirry Kassalle
13   Lopeta Testi
14
15 *** Keywords ***
16 Aloita Testi
17   Open Browser  about:blank  chrome
18
19 Etsi Tuotteita
20   Go To  https://verkkokauppa.saimaanwebpalvelut.zoner.dev/
21   Wait Until Page Contains  Kuvapankki
22   Input Text  xpath=/html/body/div[1]/div/section[2]/div/div[3]/div/div/div/form/div/input  Lumi
23   Press Keys  None  ENTER
24   Wait Until Page Contains  Lumiaura
25
26 Valitse Tuote Hakutuloksista
27   Click Link  xpath=/html/body/main/div/h2/a
28   Wait Until Page Contains  Lisää ostoskoriin
29
30 Lisää Tuote Ostoskoriin
31   Click Button  xpath=/html/body/div[2]/main/div[2]/div[2]/form/button
32   Wait Until Page Contains  "Lumiaura" on lisätty ostoskoriin.
33
34 Siirry Kassalle
35   Click Link  xpath=/html/body/main/div/div[1]/div[2]/div/div/a
36   Wait Until Page Contains  Kassa
37
38 Lopeta Testi
39   Close Browser
```

Kuva 11. Verkkokauppa.robot vaiheessa 2

Seuraavassa vaiheessa luotiin Resources -kansion alle tiedostot Common.robot sekä VerkkokauppaApp.robot, tämä on nähtävissä kuvassa 12. Kuten kuvasta 13 voi havaita, aiemmin kokonaan Verkkokauppa.robot -tiedostoon edellisessä vaiheessa luodut avainsanat on sijoitettu Common.robot sekä VerkkokauppaApp.robot -tiedostoihin. Esimerkiksi Etsi Tuotteita löytyy VerkkokauppaApp.robot -tiedostosta, kuten voidaan nähdä kuvasta 14. Kyseinen avainsana sisältää esimerkiksi verkkokaupan sivustolle siirtymisen, eli Go To. Common.robot puolestaan sisältää testitapauksen aloituksen ja lopetuksen, kuten huomataan kuvasta 15.



Kuva 12. Tiedostorakenne vaiheessa 3

```
Verkkokauppa.robot ×   VerkkokauppaApp.robot   Common.robot
Tests > Verkkokauppa.robot > Tuotteen lisääminen koriin selaamalla onnistuu
1  *** Settings ***
2  Resource  ../Resources/VerkkokauppaApp.robot
3  Resource  ../Resources/Common.robot
4  Test Setup  Common.Aloita Testi
5  Test Teardown  Common.Lopeta Testi
6
7  *** Test Cases ***
8  Tuotteen lisääminen koriin selaamalla onnistuu
9    [Documentation]  Tässä jotain infoa
10   [Tags]  Testi
11   VerkkokauppaApp.Etsi Tuotteita
12   VerkkokauppaApp.Valitse Tuote Hakutuloksista
13   VerkkokauppaApp.Lisää Tuote Ostoskoriin
14   VerkkokauppaApp.Siirry Kassalle
```

Kuva 13. Verkkokauppa.robot vaiheessa 3

```
Verkkokauppa.robot   VerkkokauppaApp.robot ×   Common.robot
Resources > VerkkokauppaApp.robot > Siirry Kassalle
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Etsi Tuotteita
6    Go To  https://verkkokauppa.[redacted]
7    Wait Until Page Contains  Kuvapankki
8    Input Text  xpath=/html/body/div[1]/div/section[2]/div/div[3]/div/div/div/form/div/input  Lumi
9    Press Keys  None  ENTER
10   Wait Until Page Contains  Lumiaura
11
12 Valitse Tuote Hakutuloksista
13   Click Link  xpath=/html/body/main/div/h2/a
14   Wait Until Page Contains  Lisää ostoskoriin
15
16 Lisää Tuote Ostoskoriin
17   Click Button  xpath=/html/body/div[2]/main/div[2]/div[2]/form/button
18   Wait Until Page Contains  "Lumiaura" on lisätty ostoskoriin.
19
20 Siirry Kassalle
21   Click Link  xpath=/html/body/main/div/div[1]/div[2]/div/div/a
22   Wait Until Page Contains  Kassa
```

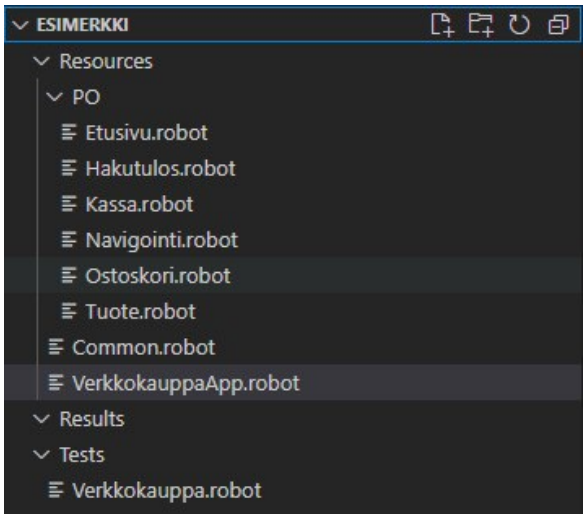
Kuva 14. VerkkokauppaApp.robot vaiheessa 3

```
Verkkokauppa.robot   VerkkokauppaApp.robot   Common.robot ×
Resources > Common.robot > Lopeta Testi
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Aloita Testi
6    Open Browser  about:blank  chrome
7
8  Lopeta Testi
9    Close Browser
```

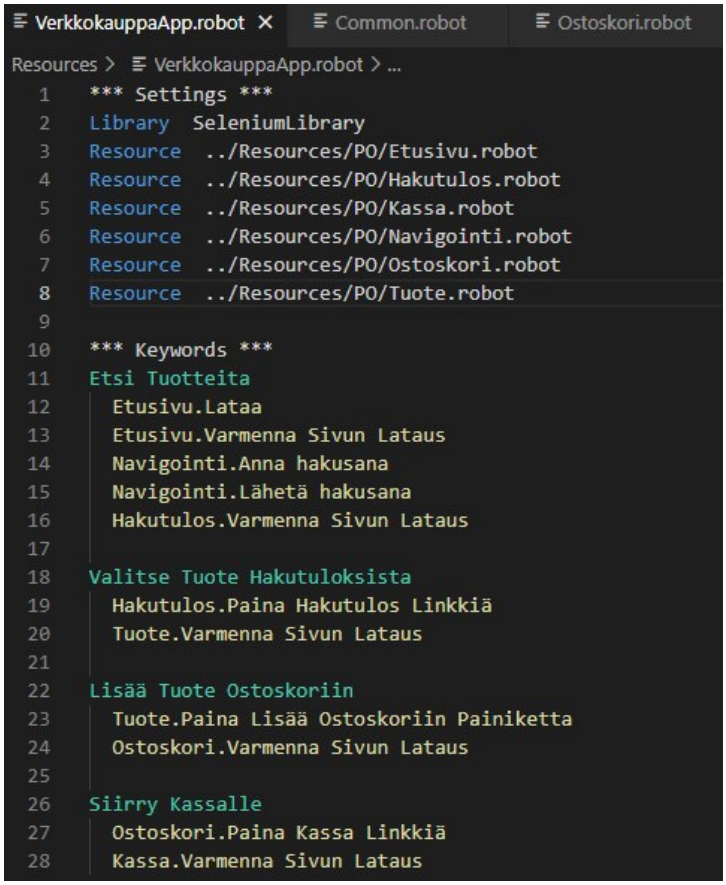
Kuva 15. Common.robot vaiheessa 3

Kuvan 16 mukaisesti seuraavaksi luotiin page object eli sivun elementti -tiedostot PO-kansioon. Jokainen testitapauksessa esiintyvä sivu sekä navigointipalkki saivat oman

sivun elementti -tiedostonsa. VerkkokauppaApp.robot -tiedostossa viime vaiheessa sijainneet testien vaiheet on nyt siirretty oikeisiin sivun elementti -tiedostoihin sen mukaan millä sivulla nämä tapahtumat ovat ja tilalle on tullut lisää avainsanoja sekä sivun elementti -tiedostojen nimiä, kuten nähdään kuvassa 17. Esimerkiksi avainsana Lataa löytyy Etusivu.robot -tiedostosta, ja tämä avainsana sisältää verkkokaupan sivun avaamisen. Etusivu.robot on nähtävissä kuvassa 18.



Kuva 16. Tiedostorakenne vaiheessa 4



Kuva 17. VerkkokauppaApp.robot vaiheessa 4

```
kauppaApp.robot   Common.robot   Ostoskori.robot   Kassa.robot   Etusivu.robot ×
Resources > PO > Etusivu.robot > Varmenna Sivun Lataus
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Lataa
6    Go To  https://verkkokauppa.[redacted]
7
8  Varmenna Sivun Lataus
9    Wait Until Page Contains  Kuvapankki
```

Kuva 18.Etusivu.robot vaiheessa 4

Viimeisessä vaiheessa lisätään Verkkokauppa.robot -tiedostoon variables- eli muuttujatosio, johon tässä esimerkkitapauksessa lisättiin selaimelle, sivuston urlille sekä hakutermille muuttujat, kuten näkyy kuvassa 19. Tällöin ne on helppo muuttaa halutessaan. Kuvassa 20 on nähtävillä miltä tiedosto, jossa muuttuja on käytössä, näyttää. Myös esimerkiksi xpath-polut tai tekstit, joita testin vaiheet odottavat sivulla näkevänsä voi halutessaan sijoittaa muuttujaan.

```
Verkkokauppa.robot ×   Common.robot   Etusivu.robot   Navigointi.robot
Tests > Verkkokauppa.robot > ${HAKUTERMI}
1  *** Settings ***
2  Resource  ../Resources/VerkkokauppaApp.robot
3  Resource  ../Resources/Common.robot
4  Test Setup  Common.Aloita Testi
5  Test Teardown  Common.Lopeta Testi
6
7  *** Variables ***
8  ${SELAIN} =  Chrome
9  ${URL} =  https://verkkokauppa.[redacted]
10 ${HAKUTERMI} =  Lumi
11
12 *** Test Cases ***
13 Tuotteen lisääminen koriin selaamalla onnistuu
14   [Documentation]  Tässä jotain infoa
15   [Tags]  Testi
16   VerkkokauppaApp.Etsi Tuotteita
17   VerkkokauppaApp.Valitse Tuote Hakutuloksista
18   VerkkokauppaApp.Lisää Tuote Ostoskoriin
19   VerkkokauppaApp.Siirry Kassalle
```

Kuva 19. Verkkokauppa.robot vaiheessa 5

```
Verkkokauppa.robot    Common.robot ×    Etusivu.robot
Resources > Common.robot > Aloita Testi
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Aloita Testi
6    Open Browser  about:blank  ${SELAIN}
7
8  Lopeta Testi
9    Close Browser
```

Kuva 20. Common.robot vaiheessa 5

Lopuksi, kun ympäristö oli pystytetty ja testit suunniteltu ja luotu, täytyi Jenkinsiin luoda vielä ajamista varten Jenkins Pipeline ja Jenkinsfile. Kun määritykset oli tehty oikein, oli testiympäristö valmis ja testejä pystyi ajamaan.

Pohdinta

Aihe on ajankohtainen ja tarpeellinen toimeksiantajayritykselle, sillä pelkällä manuaalitestauksella tuotteiden testaaminen on aikaa vievää, raskasta sekä paikoin jopa puutteellista. Opinnäytetyö ei tarjoa ratkaisua kaikkiin testauksen ongelmiin, sillä testit keskittyvät ominaisuuksien ja tapahtumien toimivuuteen. Esimerkiksi suorituskyvyn testaaminen olisi varmasti hyödyllistä.

Lopputuotos eli testausautomaation ja jatkuvan integraation käyttöönoton ohje on tavoitteiden mukainen, sillä se on helppolukuinen ja sisältää tarvittavan informaation. Toimeksiantaja kiitteli ohjeen selkeyttä ja lisäsi, että opinnäytetyö on lisännyt kiinnostusta testausautomaatioon. Toimeksiantajan palautteen mukaan testiympäristö on mahdollista pystyttää heidän olemassa olevaan ympäristöönsä, ja toimeksiantajalle toimitetun testiversion käyttöönotto suoriutui onnistuneesti. Toimeksiantajaa koki testien ajamisen lopputuloksena saatavat testilokit ja -raportit hyödyllisenä ja totesi, että kehittäjä näkee web-palvelun kehityskohteet niistä helposti. Lisäksi toimeksiantaja kertoi tulevaisuudessa ottavansa testausautomaation käyttöönsä, sillä niiden avulla voidaan selvästi säästää aikaa projektien testaamisessa.

Mielestäni työllä on hyvät jatkojalostamismahdollisuudet siinä, että testejä voidaan luoda lisää sekä parannella jo olemassa olevia. Lisäksi testiympäristöä on mahdollista kehittää esimerkiksi luomalla useampia putkia, jotka hoitavat eri testien ajoja. Testausta olisi mahdollista kehittää siten, että testataan ominaisuuksien toimivuuden lisäksi muutakin, esimerkiksi edellä mainittua suorituskykyä tai vaikkapa tietoturvaa.

Työn tietoperustan ja muun kirjallisen taustan luotettavuuden arvioisin olevan hyvä, sillä käytin paljon työkalujen omia sivustoja ja dokumentaatiota lähteinä. Esimerkiksi testien luotettavuutta on vaikeampi arvioida, sillä ilman suurempaa aiempaa kokemusta ja työelämän kautta karttunutta asiantuntijuutta on hankala arvioida testien oikeellisuutta ja tarpeellisuutta. Näkisin kuitenkin onnistuneeni tasooni nähden.

Tiedonhaun näkisin yhtenä vahvuutena, sillä löysin tarvittavaa tietoa esimerkiksi työkaluista suhteellisen helposti ja onnistuin mielestäni käyttämään niitä hyvin. Lisäksi lähteiden laatu oli nähdäkseni melko hyvä. Toinen vahvuus on ehdottomasti kirjallinen ulosanti, sillä kirjoittaminen oli kautta työn melko helppoa ja tekstin sujuvuuden varmistaminen on tullut melko luonnostaan.

Opinnäytetyön suurin kompastuskivi oli varmasti aikataulutus. Alusta lähtien olisi pitänyt tehdä selvä aikataulu, jota seurata ja muokata sitä tarvittaessa. Lisäksi valittu aihe oli

melko haastava, sillä osaamista ei juuri ollut. Tästä syystä opinnäytetyön lopputuotos hieman muuttui työn edetessä ja alunperin olisikin pitänyt ottaa selvää ja suunnitella paremmin, mitä tulee tehdä tuloksen saavuttamiseksi.

Yksi työn tavoitteista oli kehittää omaa osaamista aihepiiriin liittyen. Se ehdottomasti toteutui, sillä kokemusta aiheesta ei juuri ollut ennen opinnäytetyöprojektia. Sen puolesta puhuu myös lopputuotoksen valmiiksi saaminen, sillä ilman osaamisen kehittymistä se ei olisi ollut mahdollista. Opin paljon etenkin Robot Framework testisarjan kirjoittamisesta sekä testiympäristön pystyttämisestä. Testiympäristöä pystyttäessä opin paljon eri työkaluista, joita siihen tarvitaan sekä siitä miten ne otetaan onnistuneesti käyttöön ja muodostetaan yhtenäinen testiympäristö.

Ammatillisesta kehityksestä on vaikea sanoa, sillä työn luonne oli hieman erilainen verrattuna töihin, jotka tehdään esimerkiksi suurille yrityksille. Tässä kyseessä oli entuudestaan tuttu pienikokoinen yritys. Lisäksi opinnäytetyön teko hoitui täysin etänä ja itsenäisesti.

Lähteet

Appsurify 2021. 5 Best Automation Tools in 2022: Pros and Cons. Luettavissa: https://appsurify.com/resources/5-best-automation-tools-in-2022-pros-and-cons/. Luettu: 20.11.2022.

AWS s.a. What is Continuous Integration? Luettavissa: https://aws.amazon.com/devops/continuous-integration/. Luettu: 9.11.2022.

Axelrod, A. 2018. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects. Apress. New York City. Luettavissa: https://learning.oreilly.com/library/view/complete-guide-to/9781484238325/. Luettu: 24.5.2021.

Azure s.a. Pricing for Azure DevOps. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/. Luettu: 19.11.2022.

Bisht, S. 2013. Robot Framework Test Automation. Packt Publishing. Birmingham. Luettavissa: https://learning.oreilly.com/library/view/robot-framework-test/9781783283033/. Luettu: 25.5.2021.

Channenahalli Ramachandraiah, A. 2018. Robot Framework – An unglorified hero – pt. 2. Luettavissa: https://www.devonblog.com/test-automation/robot-framework-an-unglorified-hero-pt-2/. Luettu: 11.5.2022.

Chen, J. 2020. Attacker's Tactics and Techniques in Unsecured Docker Daemons Revealed. Luettavissa: https://unit42.paloaltonetworks.com/attackers-tactics-and-techniques-in-unsecured-docker-daemons-revealed/. Luettu: 19.11.2022.

Docker s.a. a. Docker overview. Luettavissa: https://docs.docker.com/get-started/overview/. Luettu: 1.11.2022

Docker s.a. b. Docker Personal. Luettavissa: https://www.docker.com/products/personal/. Luettu: 19.11.2022.

Dustin, E., Garrett, T & Gauf, B. 2009. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality. Addison-Wesley Professional. Boston. Luettavissa: https://learning.oreilly.com/library/view/implementing-automated-software/9780321619600/. Luettu: 21.6.2021.

Figueiredo, R & Gamela, A. 2021. Podman vs Docker: What are the differences? Luettavissa: https://www.imaginarycloud.com/blog/podman-vs-docker/#Differences. Luettu: 19.11.2022.

Geeks for Geeks 2022. Robot Framework vs Selenium. Luettavissa: https://www.geeksforgeeks.org/robot-framework-vs-selenium/?fbclid=IwAR3pj4IycEH7Ab3Kw7kLfFE3XGPqKoui0XYTljbUXSEZRpC1PeRhEarwwDI. Luettu: 20.11.2022.

GitHub s.a. Get the complete developer platform. Luettavissa: https://github.com/pricing. Luettu: 20.11.2022.

GitLab s.a. Get The One DevOps Platform. Luettavissa: https://about.gitlab.com/pricing/. Luettu: 19.11.2022.

Jenkins s.a. Pipeline. Luettavissa: https://www.jenkins.io/doc/book/pipeline/. Luettu: 19.11.2022.

Kszczanowicz, L & Pomianowski, Z. 2022. Declarative vs Scripted Pipeline – Key differences. Luettavissa: https://dsstream.com/declarative-vs-scripted-pipeline-key-differences/?fbclid=IwAR3KOL9jUFJnb2vTTmFv_rXGG0fV1XS6E-OJKoXnuS1Z6AeWPfVX3xo_b3I. Luettu: 19.11.2022.

Microsoft 2021. What is the Windows Subsystem for Linux? Luettavissa: https://docs.microsoft.com/en-us/windows/wsl/about. Luettu: 25.4.2022.

Mikkonen, J. 2019. Mikä ihmeen Jenkins? Jatkuva integraatio pitää koodin tuoreena. Luettavissa: https://www.tivi.fi/uutiset/mika-ihmeen-jenkins-jatkuva-integraatio-pitaa-koodin-tuoreena/61d2bad5-ca2b-4f12-a4f4-b3809e0e0dd7. Luettu 16.11.2022.

Molina, A. 2021. Crafting Test-Driven Software with Python. Packt Publishing. Birmingham. Luettavissa: https://learning.oreilly.com/library/view/crafting-test-driven-software/9781838642655/. Luettu: 22.6.2021

O'Grady, A. 2018. GitLab Quick Start Guide. Packt Publishing. Birmingham. Luettavissa: https://learning.oreilly.com/library/view/gitlab-quick-start/9781789534344/. Luettu: 22.6.2021.

Podman s.a. What is Podman? Luettavissa: https://docs.podman.io/en/latest/. Luettu: 19.11.2022.

Ramya, P & Sindhura, V. 2017. Institute of Electrical and Electronics Engineers. New York City. Luettavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8117878. Luettu: 26.5.2021.

Reynolds, J. 2022. Understanding the Types of Test Environments. Luettavissa: https://www.enov8.com/blog/understanding-the-types-of-test-environments/. Luettu: 19.11.2022.

Robot Framework 2016. Robot Framework User Guide. Luettavissa: https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html?fbclid=IwAR2SewSSnAA4-GJMfjdPqv48eKMi6zYmKcnXG_hZxlW_ssTD74SEJuuDU4U#different-output-files. Luettu: 16.11.2022.

Robot Framework s.a. Introduction. Luettavissa: https://robotframework.org/. Luettu: 25.5.2021.

Selenium 2021. Understanding the components. Luettavissa: https://www.selenium.dev/documentation/en/webdriver/understanding_the_components/. Luettu: 26.5.2021.

Selenium s.a. The Selenium Browser Automation Project. Luettavissa: https://www.selenium.dev/documentation/en/. Luettu: 26.5.2021.

Sommerville, I. 2016. Software Engineering. Pearson Education. London. Luettavissa: https://www.vlebooks.com/Vleweb/Product/Index/807705?page=0. Luettu: 24.6.2021.

Soni, M. 2017. Jenkins Essentials – Second Edition. Packt Publishing. Birmingham. Luettavissa: https://learning.oreilly.com/library/view/jenkins-essentials-/9781788471060/. Luettu: 21.6.2021.

Systemd 2022. System and Service Manager. Luettavissa: https://systemd.io/. Luettu: 19.11.2022.

Liitteet

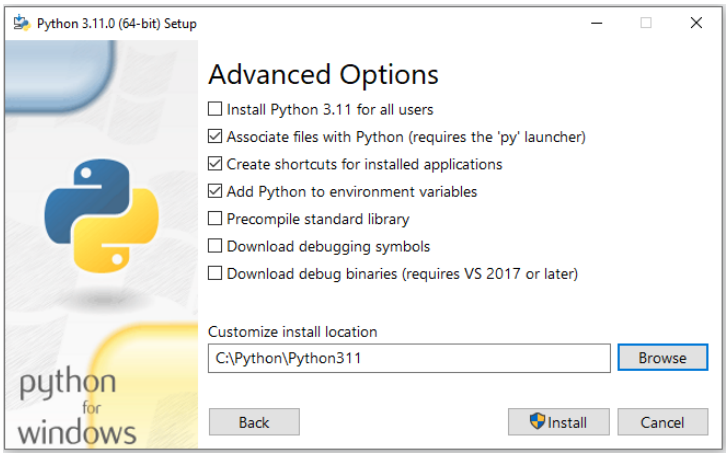
Liite 1. Testiautomaation ja jatkuvan integraation käyttöönoton ohje

Python

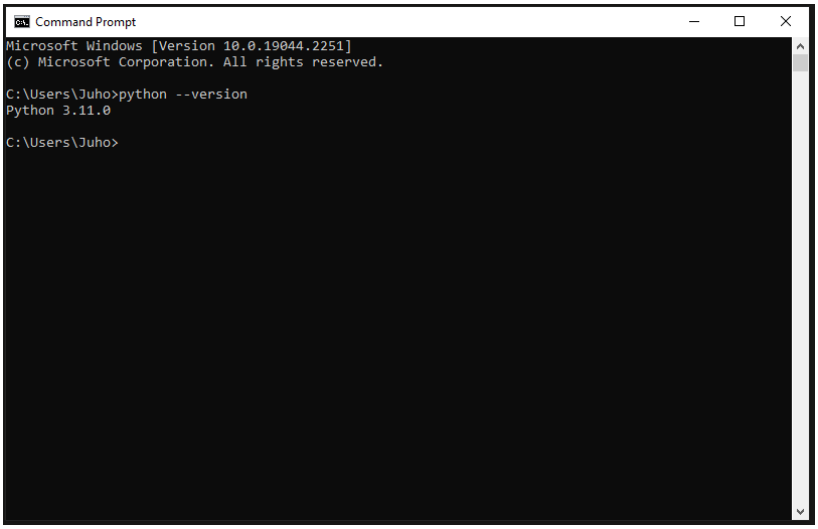
Pythonista on suositeltavaa ladata uusin saatavilla oleva versio ja asennuspaketti on turvallisinta ladata python.org sivuston kautta. Robot Framework vaatii toimiakseen vähintään Python 3.6 tai uudemman version.

1. Python version 3.11 voi ladata osoitteesta https://www.python.org/ftp/python/3.11.0/python-3.11.0-amd64.exe
2. Lisätään Python pathiin jo asennusvaiheessa valitsemalla Add python.exe to PATH.
3. Valitaan Customize installation, jotta asennuspolku voidaan määrittää itse, muut valinnat jätetään sellaisiksi kuin ne oletuksena ovat.

4. Asennuspoluksi määritetään C:\Python\Python311\. Tällä tavalla on helppo hallita asennettuja Python versioita, mikäli on tarve useammalle versiolle samanaikaisesti.



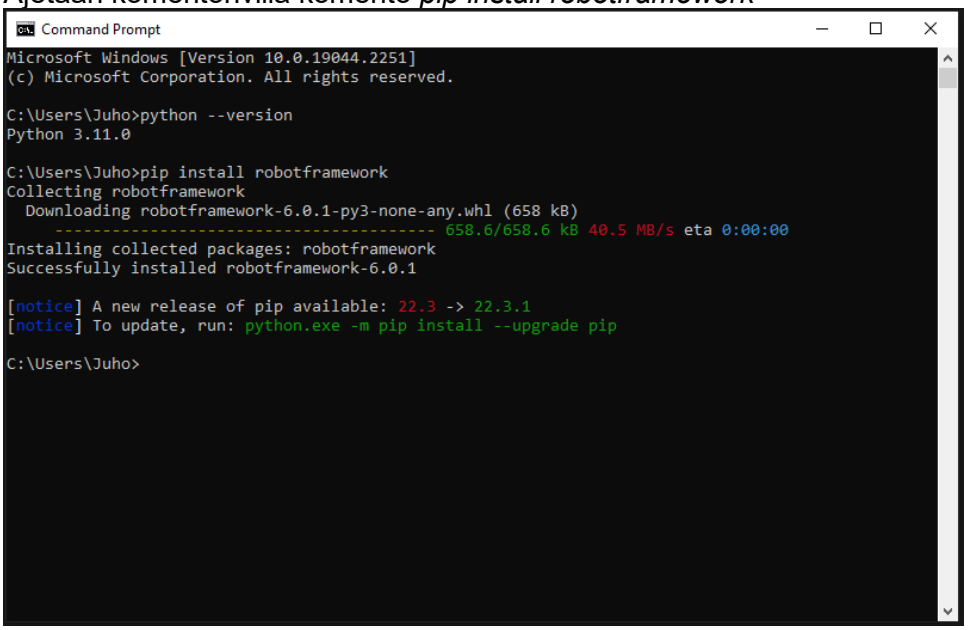
5. Asennuksen päätteeksi on hyvä tarkastaa, että Python on asentunut oikein ja että se löytyy pathistä. Tämä on helpointa toteuttaa ajamalla komentorivillä tai powershellillä komento *python --version*. Komento toimii ainoastaan siinä tapauksessa, että Python on pathissa.



Robot Framework

Robot Frameworkin uusimman version voi asentaa komentorivin kautta.

1. Ajetaan komentorivillä komento *pip install robotframework*



2. Asennuksen toimivuus voidaan tarkistaa ajamalla komento *robot --version*, joka kertoo mikä versio Robot Frameworkista on asennettu, sekä mitä Python versiota se käyttää.

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Juho>python --version
Python 3.11.0

C:\Users\Juho>pip install robotframework
Collecting robotframework
  Downloading robotframework-6.0.1-py3-none-any.whl (658 kB)
     ---------------------------------------- 658.6/658.6 kB 40.5 MB/s eta 0:00:00
Installing collected packages: robotframework
Successfully installed robotframework-6.0.1

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Juho>robot --version
Robot Framework 6.0.1 (Python 3.11.0 on win32)

C:\Users\Juho>
```

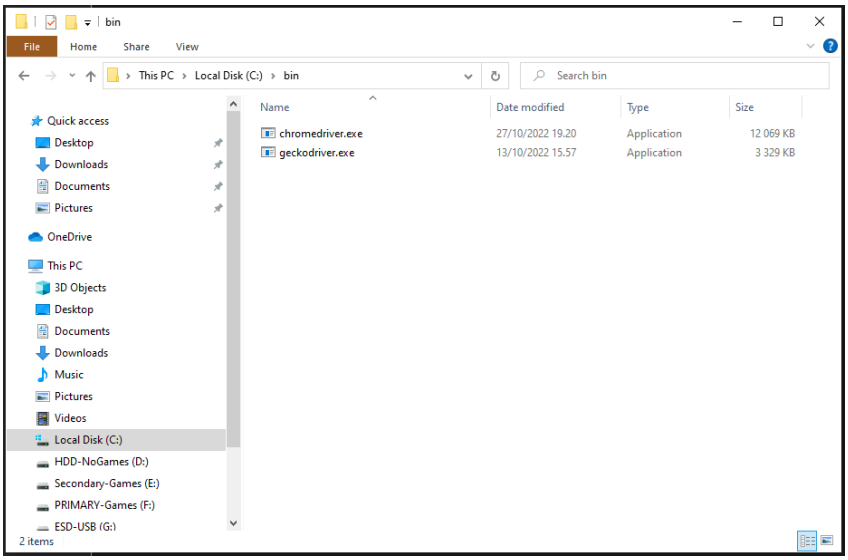
Selenium

Selenium asennetaan komentorivin kautta. Lisäksi asennetaan tarpeelliset selainkohtaiset selainajurit.

1. Aja komentorivillä komento pip install robotframework-seleniumlibrary.
2. Asennuksen jälkeen ladataan selainajurit osoitteesta https://www.selenium.dev/documentation/webdriver/getting_started/install_drivers/.

Browser	Supported OS	Maintained by	Download	Issue Tracker
Chromium/Chrome	Windows/macOS/Linux	Google	Downloads	Issues
Firefox	Windows/macOS/Linux	Mozilla	Downloads	Issues
Edge	Windows/macOS/Linux	Microsoft	Downloads	Issues
Internet Explorer	Windows	Selenium Project	Downloads	Issues
Safari	macOS High Sierra and newer	Apple	Built in	Issues

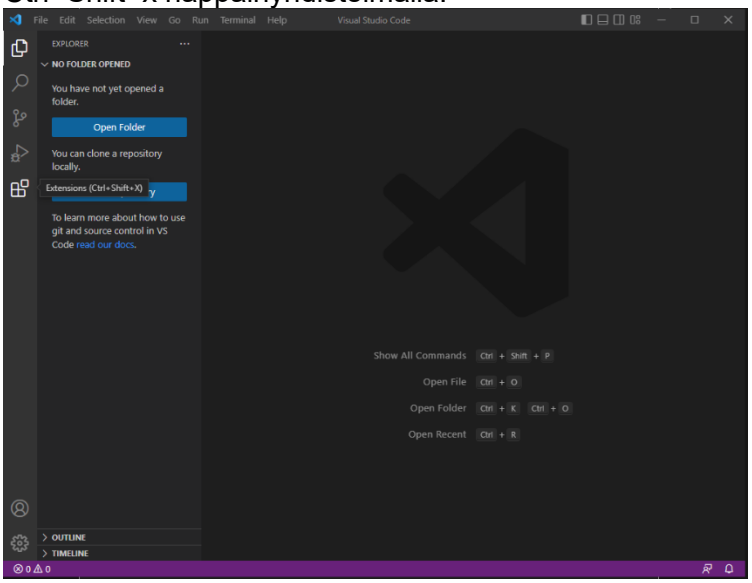
3. Ajureiden latauksen jälkeen luodaan bin -niminen kansio polkuun C:\bin ja myös se tulee lisätä pathiin.
4. Kun kansio on luotu, tulee ladatut paketit purkaa sen alle.



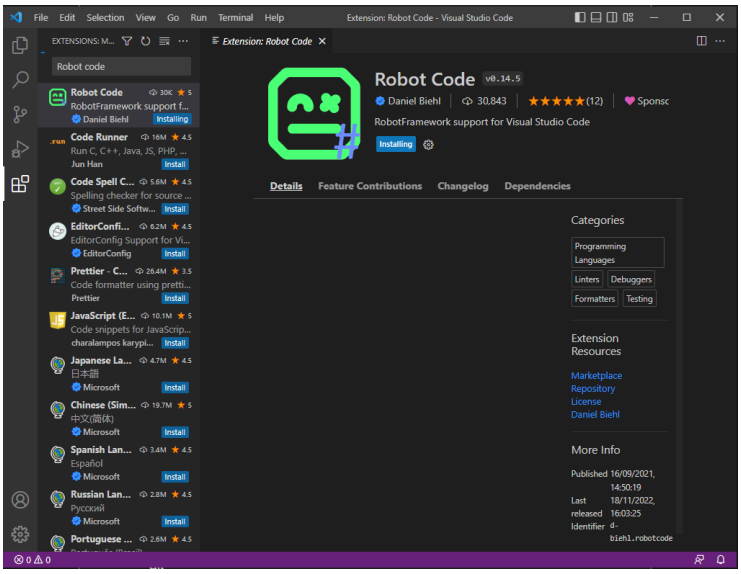
Visual Studio Code

Testien kirjoittamiseen tarvitaan tekstieditori, joka tässä tapauksessa on Visual Studio Code.

1. Ladataan asennuspaketti osoitteessa https://code.visualstudio.com/Download.
2. Asennetaan ladattu paketti.
3. Asennuksen jälkeen avataan VS Code ja lisätään tarvittavat laajennukset. Laajennusvalikkoon pääse valitsemalla VS Coden näkymästä oikealta tai Ctrl+Shift+x näppäinyhdistelmällä.



4. Kirjoitetaan laajennusnäkymässä hakukenttään Robot Code, valitaan avautuvasta listasta alla olevan kuvan mukainen Robot Code-laajennus ja asennetaan se.



Git

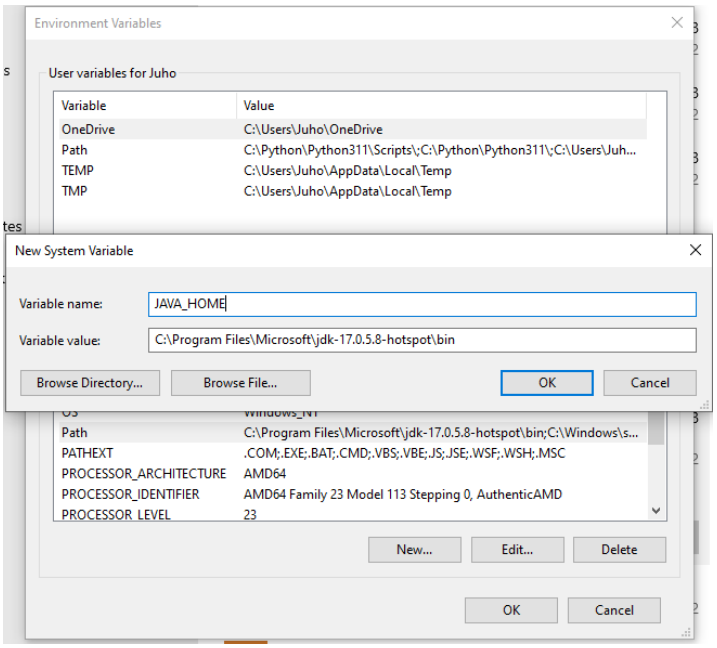
Jotta koodia voidaan siirtää versionhalintaan tarvitaan Git.

1. Ladataan asennuspaketti osoitteesta https://git-scm.com/downloads.
2. Asennetaan ladattu paketti, käyttäen asennuksen oletusarvoja.

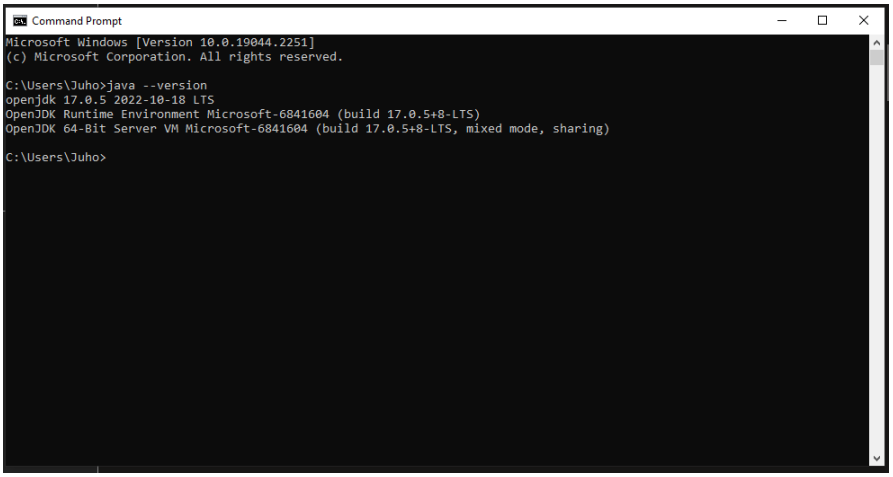
Java

Javaa tarvitaan Jenkins noodien pystytyksessä. Oraclen Javan sijasta käytetään OpenJDK:tä.

1. Ladataan asennuspaketti osoitteesta https://learn.microsoft.com/en-us/java/openjdk/download.
2. Asennetaan ladattu paketti, käyttäen oletusarvoja.
3. Lisätään ympäristömuuttujiin JAVA_HOME muuttuja ja annetaan arvoksi polku johon OpenJDK asennettiin.



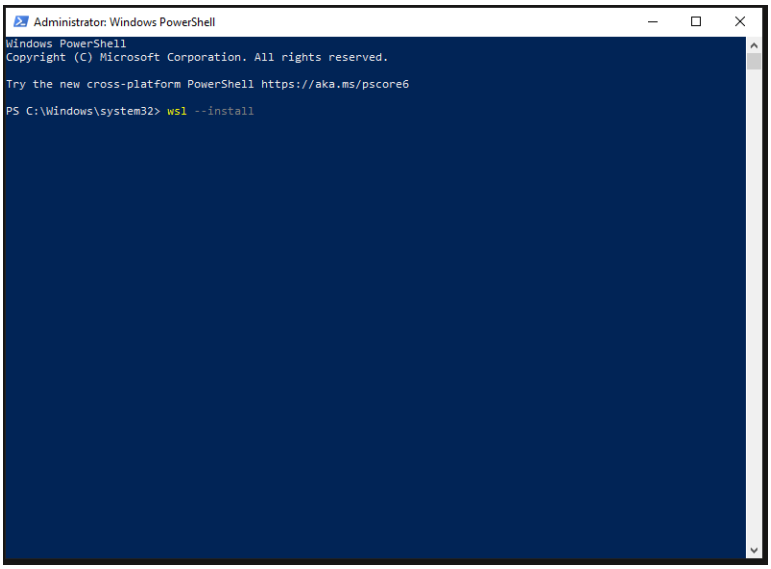
4. Tarkistetaan Javan toimivuus ajamalla komento *java --version* komentorivillä.



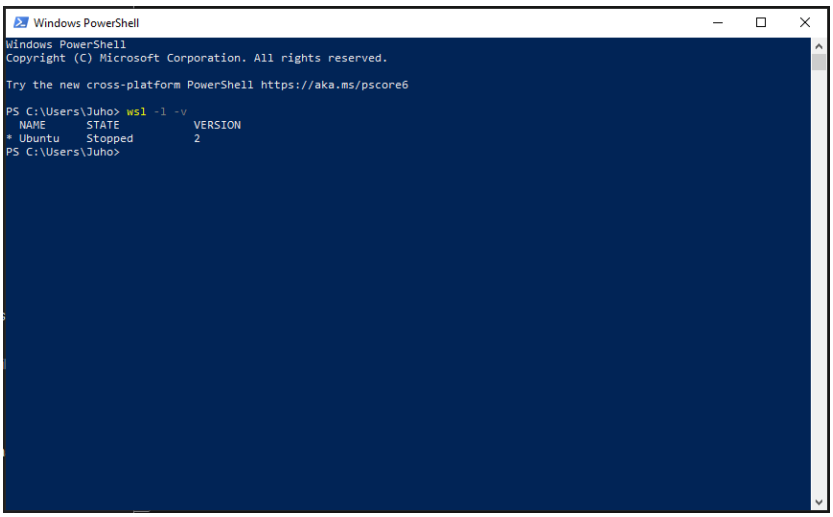
WSL2

Ennen Dockerin asennusta asennetaan koneelle WSL2 eli Windows Subsystem for Linux, jota käytämme Dockerin käytön mahdollistamiseen Hyper-V:n sijasta.

1. Avataan Powershell järjestelmänvalvojana ja ajetaan komento *wsl --install*.



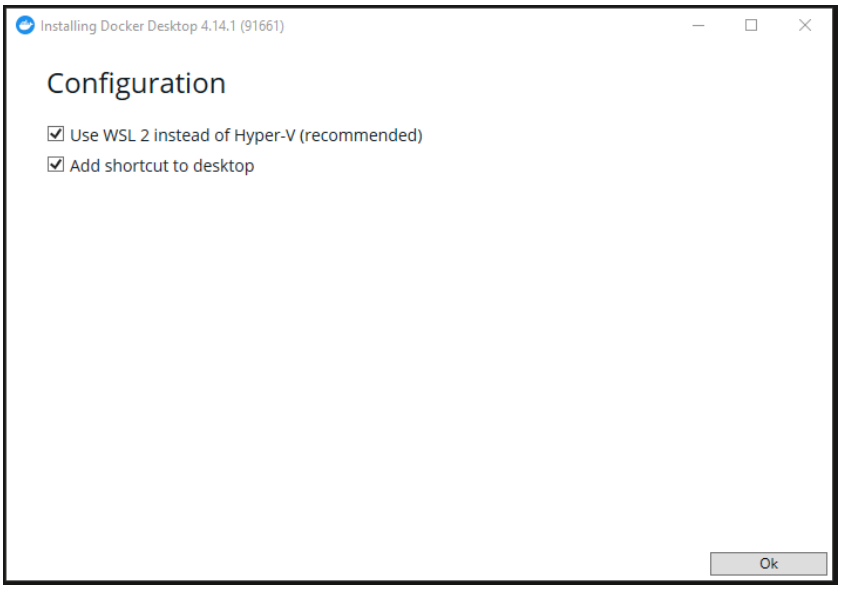
2. Kun asennuskomento on suorittanut asennuksen, tulee kone käynnistää uudelleen.
3. Koneen käynnistyttyä uudelleen WSL2 on käytettävissä ja WSL2 komentorivin tulisi avautua kirjauduttaessa sisään.
4. Ennen Dockerin asennusta on hyvä tarkistaa WSL:n versio ajamalla Powershellillä komento *wsl -l -v*. Version tulee olla 2, jos versio on 1, tulee ajaa *wsl --set-version <nimi> 2*. Tässä tapauksessa nimi olisi Ubuntu.



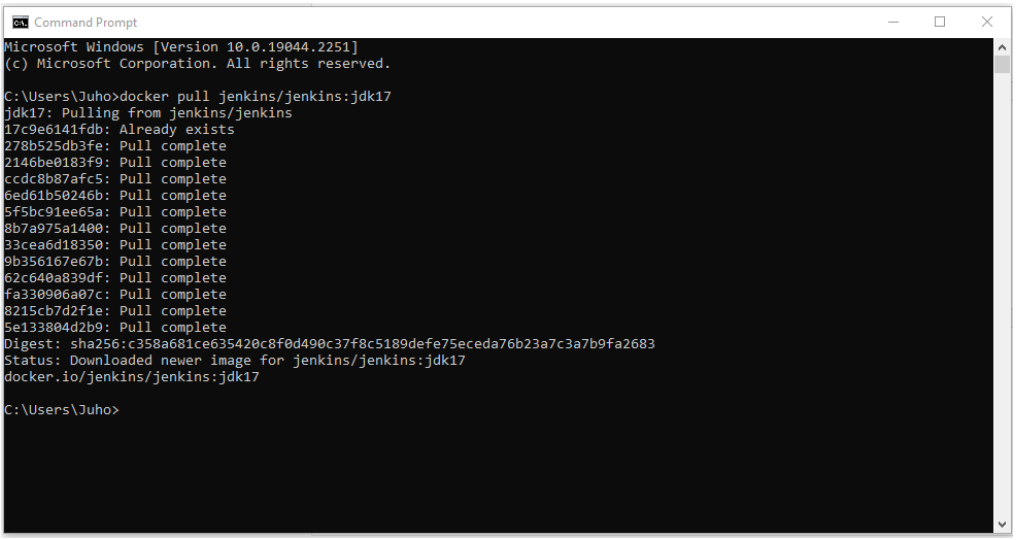
Docker

Jotta Jenkins saadaan pystytettyä, tarvitaan Docker, ja koska käytössä on Windows-ympäristö, käytetään Docker Desktopia. Docker on myös mahdollista pystyttää WSL:n sisälle.

1. Ladataan asennuspaketti osoitteesta https://www.docker.com/products/docker-desktop/.
2. Aloitetaan asennus. Configuration-vaiheessa asetetaan asetukset alla olevan kuvan mukaisesti.



3. Kun asennus on valmis, Dockerin voi käynnistää.
4. Dockerin käynnistyttyä, voidaan hakea tarvittava Jenkins image koneelle avaamalla komentorivi ja ajamalla komento *docker pull jenkins/jenkins:jdk17*.



5. Kun Image on haettu, voidaan Jenkins-kontti käynnistää komennolla *docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:jdk17.* Komentorivi tulee jättää taustalle auki, jotta yhteys Dockerin ja Jenkinsin välillä ei katkea.

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Juho>docker pull jenkins/jenkins:jdk17
jdk17: Pulling from jenkins/jenkins
17c9e6141fdb: Already exists
278b525db3fe: Pull complete
2146be0183f9: Pull complete
ccdc8b87afc5: Pull complete
6ed61b50246b: Pull complete
5f5bc91ee65a: Pull complete
8b7a975a1400: Pull complete
33cea6d18350: Pull complete
9b356167e67b: Pull complete
62c640a839df: Pull complete
fa330906a07c: Pull complete
8215cb7d2f1e: Pull complete
5e133804d2b9: Pull complete
Digest: sha256:c358a681ce635420c8f0d490c37f8c5189defe75eceda76b23a7c3a7b9fa2683
Status: Downloaded newer image for jenkins/jenkins:jdk17
docker.io/jenkins/jenkins:jdk17

C:\Users\Juho>docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:jdk17
```

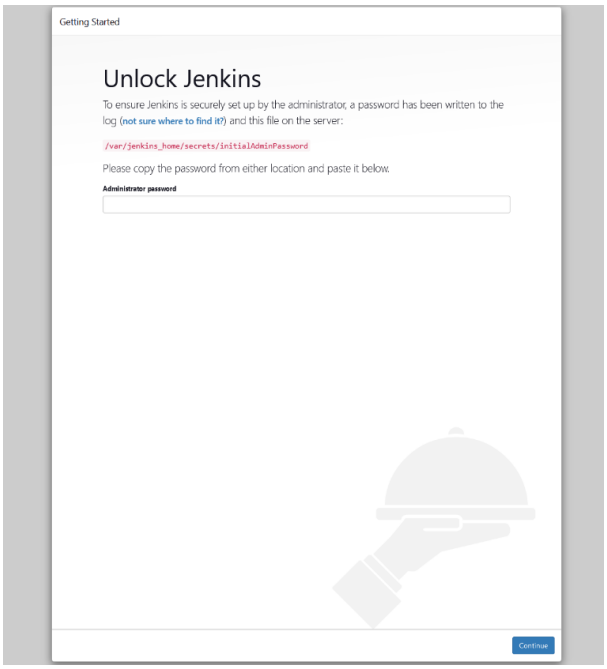
Jenkins

Jenkinsin asennus jatkuu suoraan Docker asennuksen viimeisestä vaiheesta. Jenkins pystytetään Dockeria hyödyntäen.

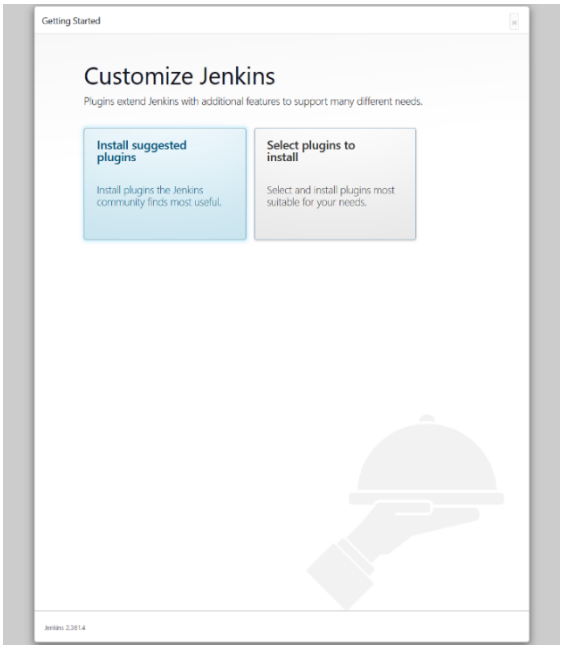
1. Ensin kopioidaan talteen komentorivillä näkyvä järjestelmänvalvojan salasana.

```
Command Prompt - docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk11
2022-11-20 14:01:43.397+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started Server@32c8e539{STARTING}[10.0.11,sto=0] @
1581ms
2022-11-20 14:01:43.402+0000 [id=35]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2022-11-20 14:01:43.565+0000 [id=42]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2022-11-20 14:01:43.582+0000 [id=61]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2022-11-20 14:01:44.133+0000 [id=69]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2022-11-20 14:01:44.138+0000 [id=49]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2022-11-20 14:01:44.146+0000 [id=55]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2022-11-20 14:01:44.327+0000 [id=57]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2022-11-20 14:01:44.328+0000 [id=65]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2022-11-20 14:01:44.329+0000 [id=47]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2022-11-20 14:01:44.331+0000 [id=50]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2022-11-20 14:01:44.346+0000 [id=84]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$1: Started Download metadata
2022-11-20 14:01:44.356+0000 [id=84]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1 (file:/var/jenkins_home/war/WEB-INF/lib/groovy-all-2.4.21.jar
) to constructor java.lang.invoke.MethodHandles$Lookup(java.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2022-11-20 14:01:44.692+0000 [id=63]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

9b589b4419954d59af64ddd384e53160

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

2022-11-20 14:01:57.550+0000 [id=57]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2022-11-20 14:01:57.570+0000 [id=32]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2022-11-20 14:01:57.978+0000 [id=84]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Mave
n.MavenInstaller
2022-11-20 14:01:57.979+0000 [id=84]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the at
tempt #1
2022-11-20 14:01:57.981+0000 [id=84]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$1: Finished Download metadata. 13,633 ms
```

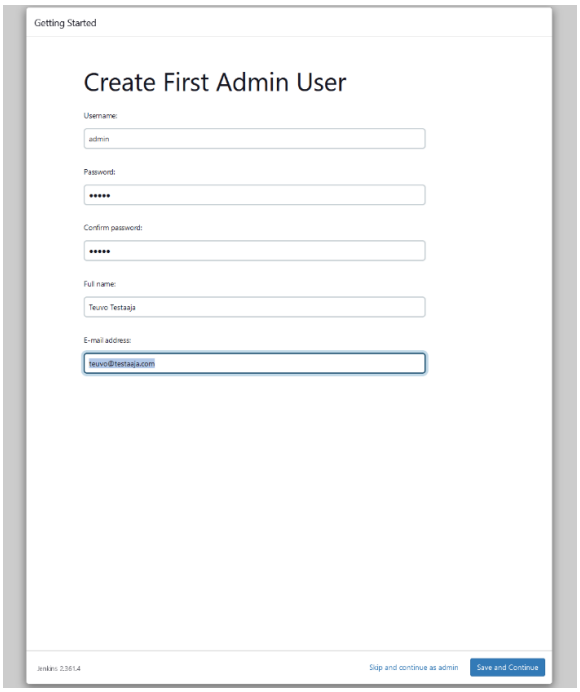
2. Avataan selain osoitteeseen localhost:8080, jolloin selaimeen avautuu kenttä, johon kopioitu salasana voidaan syöttää.



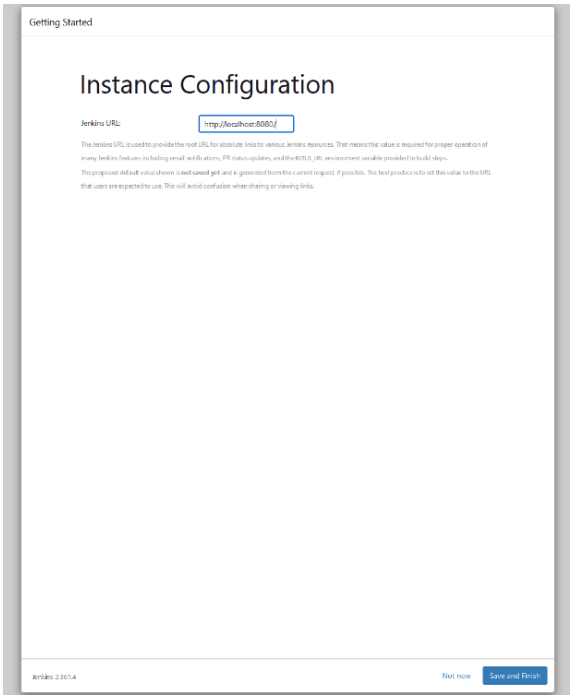
3. Salasanan syöttämisen jälkeen avautuu Jenkinsiin asennettavien laajennuksien valinta. Tässä kohtaa valitaan suositeltujen laajennuksien asennus. Tämä vaihtoehto asentaa yleisimmin käytetyt laajennukset.



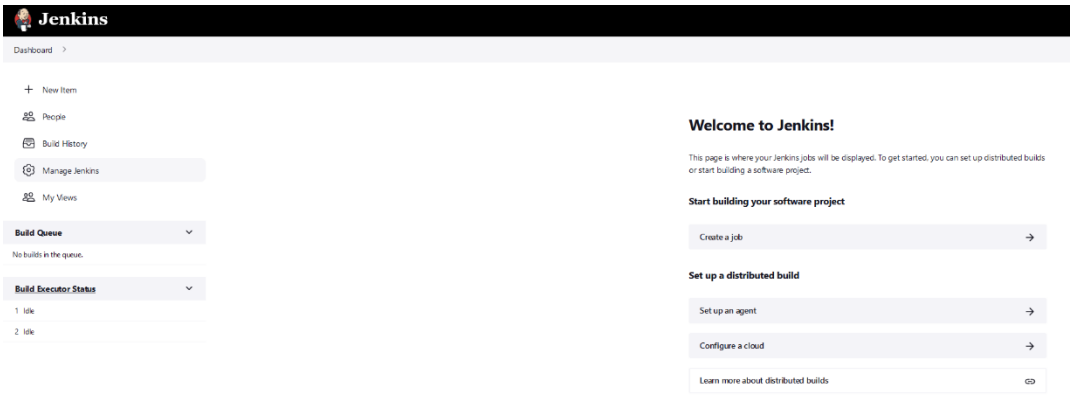
4. Laajennusten asennusten jälkeen Jenkins pyytää käyttäjää luomaan järjestelmänvalvojatilin.



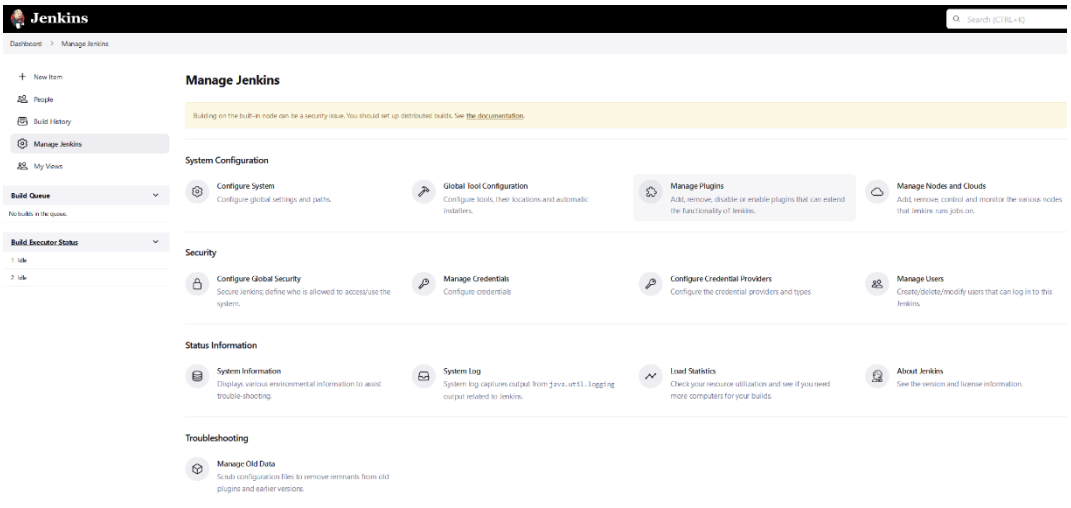
5. Viimeisessä asennus vaiheessa on mahdollista muuttaa Jenkinsin osoitetta, mutta tässä tapauksessa se jätetään oletusarvoonsa.



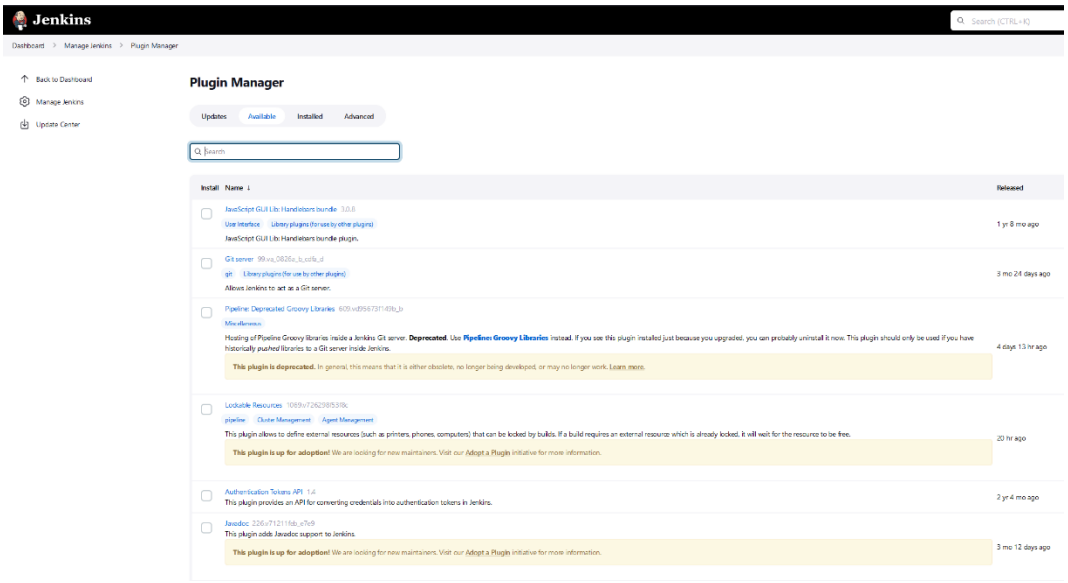
6. Tämän jälkeen Jenkins on käyttövalmis. Tässä vaiheessa on ottaa käyttöön Jenkinsin Robot Framework -laajennus. Laajennuksia voidaan lisätä Jenkinsin hallintanäkymän kautta. Sinne pääsee vasemmalla olevasta Manage Jenkins painikkeesta tai kirjoittamalla osoitekenttään http://localhost:8080/manage/.



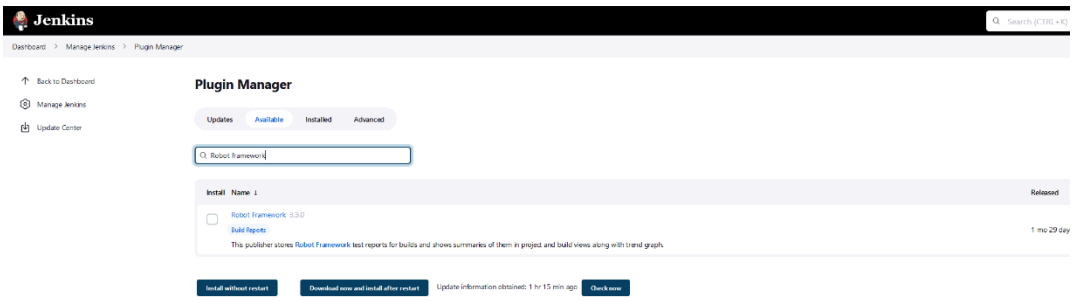
7. Hallintapaneelin kautta voidaan hallita erinäisiä Jenkins asetuksia, kuten laajennuksia. Laajennusten hallinta löytyy Manage Plugins -painiketta painamalla tai kirjoittamalla osoitekenttään http://localhost:8080/manage/pluginManager/.



8. Laajennusten hallintaosiosta saa selville mitä laajennuksia kyseiseen Jenkinsin instassiin on asennettu, mitä laajennuksia on saatavilla ja mihin asennettuihin laajennuksiin on päivityksiä. Saatavilla olevat laajennukset löytyvät Available -valikon alta.



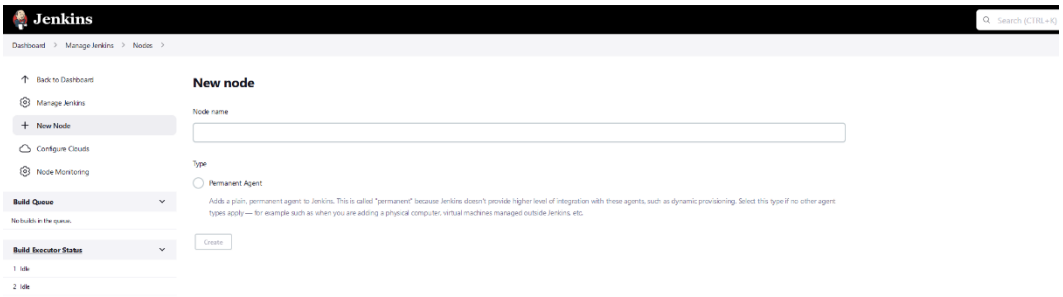
9. Kirjoittamalla haku kenttään Robot Framework löytyy tarvittava laajennus. Laajennus asennetaan valitsemalla haluttu asennus ja sen jälkeen painamalla Install without restart.



10. Tämän jälkeen laajennus asentuu ja se on tämän jälkeen heti käytettävissä.

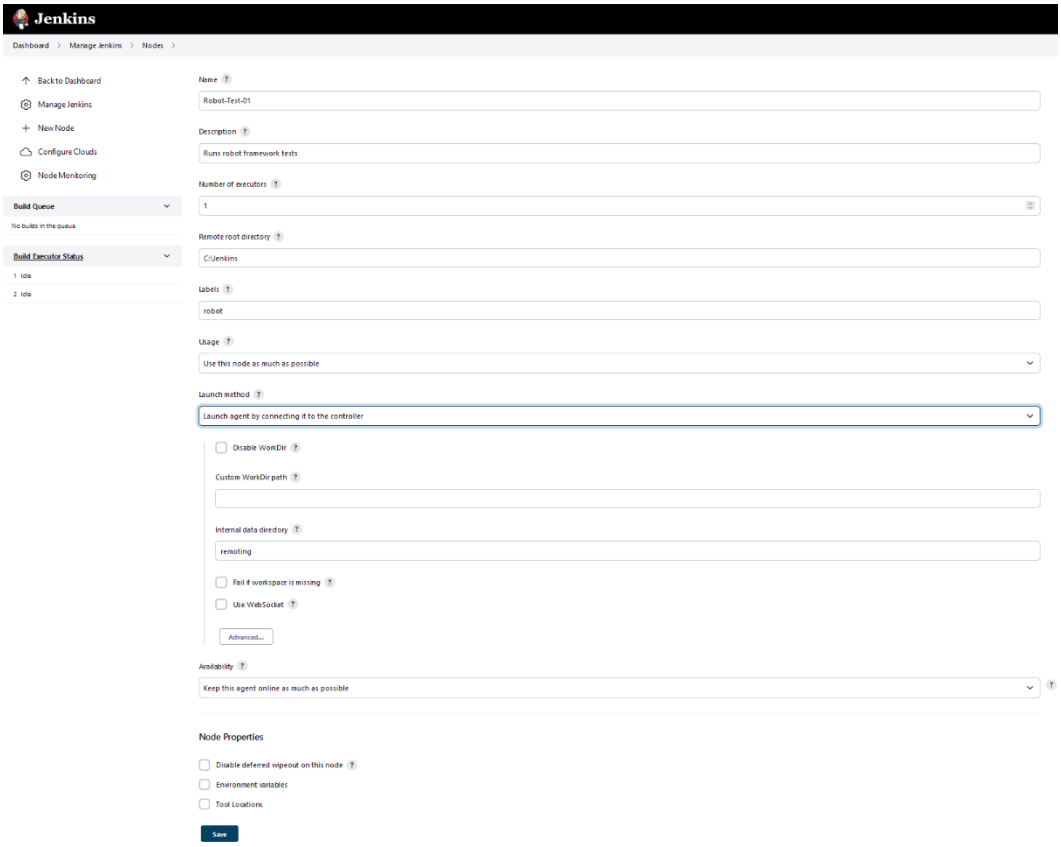
Jenkins noodin luonti

1. Avataan selaimessa localhost:8080, jos se ei ole jo auki.
2. Jenkinsin etusivulla siirrytään Jenkinsin hallintapaneliin vasemman sivuvalikon Manage Jenkins -painikeella tai osoitteella http://localhost:8080/manage.
3. Hallinta paneelista painetaan Manage Nodes and Clouds -painiketta. Lopuksi painetaan New Node -painiketta tai osoitteella tai siirrytään osoitteeseen http://localhost:8080/manage/computer/new.
4. Seuraavaksi annetaan noodille kuvaava nimi esimerkiksi Robot-Test-01 ja valitaan Permanent Agent -optio.



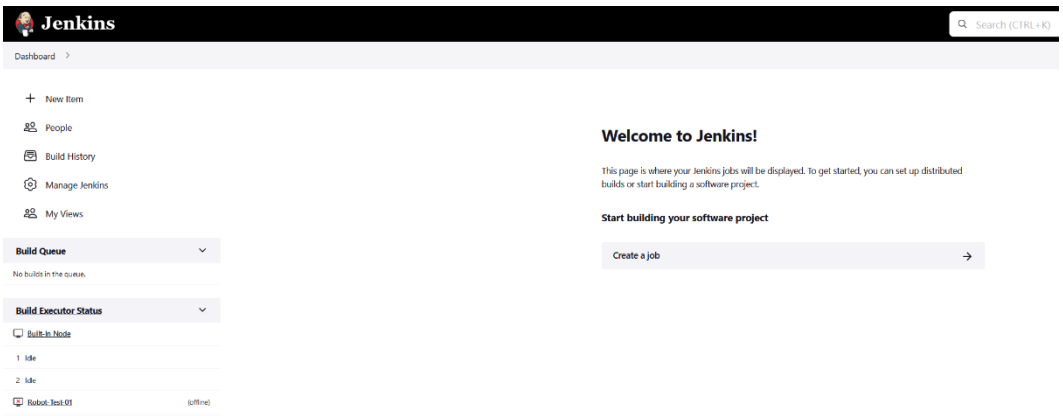
5. Tämän jälkeen avautuu näkymä, jossa määritetään noodille asetukset kuten kuvaus, polku ja suorittajien määrä.
6. Noodille asetetaan seuraavat arvot:
   - Description: Runs robot tests
   - Number of executors: 1
   - Remote root directory: C:\Jenkins
   - Labels: robot.

   Kun arvot on asetettu, noodi voidaan tallentaa.

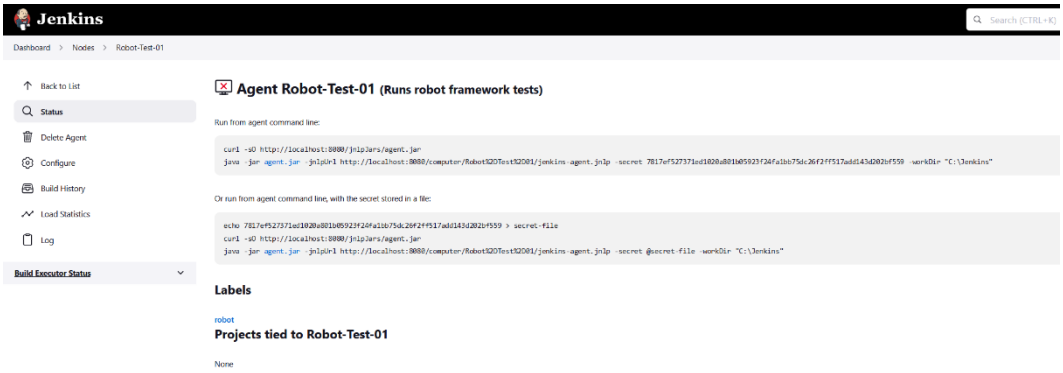


Jenkins noodin käynnistys

1. Avataan selaimessa localhost:8080, jos se ei ole jo auki.
2. Aiemmin luotu noodi on valittavissa vasemman laidan Build Executor Status valikossa.

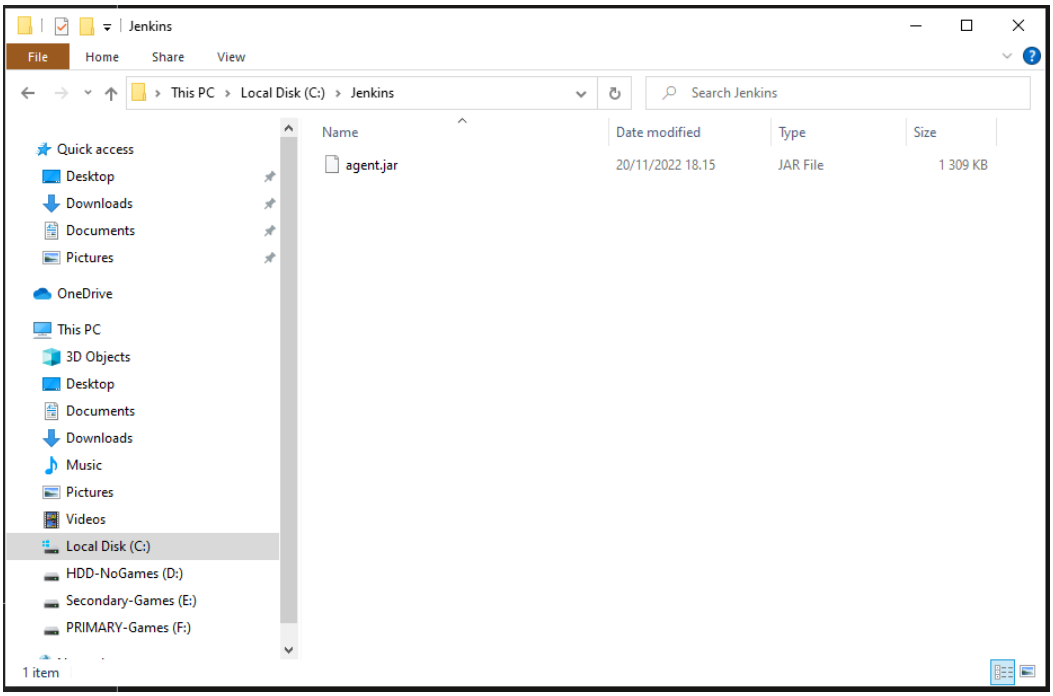


3. Avataan noodi-näkymä joko klikkaamalla aiemmin luotua noodia Build Executor Status valikossa tai kirjoittamalla osoitekenttään http://localhost:8080/computer/Robot%2DTest%2D01/.

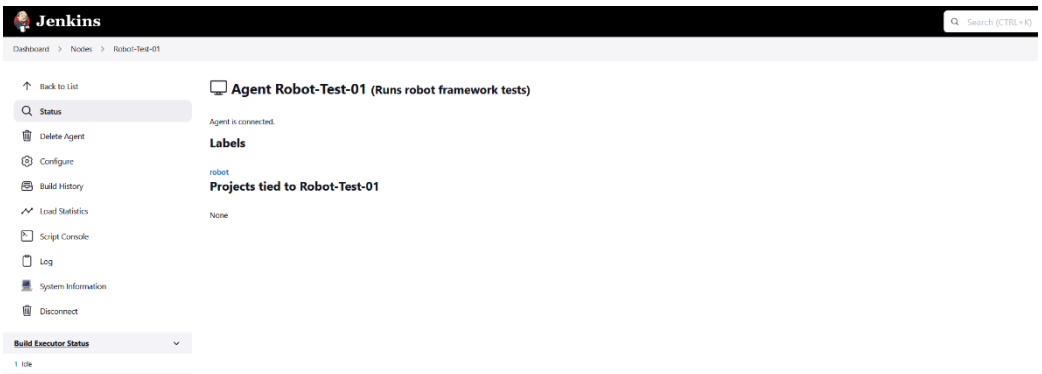


4. Noodi on tällä hetkellä offline-tilassa. Jotta se saadaan online-tilaan, täytyy koneelle, joka pyörittää noodia, ladata agent.jar tiedosto. Tiedoston voi ladata näkymästä painamalla agent.jar -linkkiä tai menemällä osoitteeseen http://localhost:8080/jnlpJars/agent.jar.

5. Luodaan kansio nimeltä Jenkins polkuun C:\Jenkins ja siirretään agent.jar kyseiseen kansioon.



6. Avataan komentorivi ja siirrytään komennolla *cd C:\Jenkins* luotuun kansioon.
7. Tämän jälkeen ajetaan komentorivillä noodi-näkymästä löytyvä komento java -jar agent.jar -jnlpUrl http://localhost:8080/computer/Robot%2DTest%2D01/jenkins-agent.jnlp -secret 7817ef527371ed1020a801b05923f24fa1bb75dc26f2ff517add143d202bf559 -workDir "C:\Jenkins".
8. Kun komento on ajettu, päivitetään noodi-näkymä ja noodi on yhdistetty.

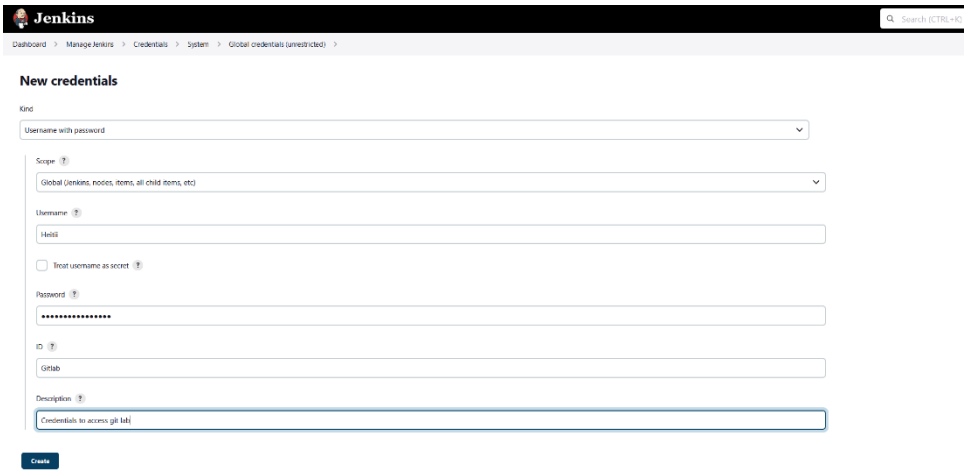


9. Noodi pysyy yhdistettynä niin kauan kun komentorivi, jolla komento ajettiin, pidetään auki. Noodi voidaan myös asentaa pyörimään Windows-palveluna, jolloin noodi pyörii aina koneen ollessa käynnissä.

Jenkins Git tunnusten luonti

1. Avataan selaimessa localhost:8080, jos se ei ole jo auki.
2. Jenkinsin etusivulla siirrytään Jenkinsin hallintapaneliin vasemman sivuvalikon Manage Jenkins -painikeella tai osoitteella http://localhost:8080/manage.
3. Hallintapaneeli-näkymässä painetaan Manage Credentials -painiketta
4. Käyttäjätunnus-näkymässa valitaan System
5. System-näkymän alta painetaan Global credentials (unrestricted).
6. Tässä näkymässä oikealta löytyy Add Credentials -painike, painetaan sitä.

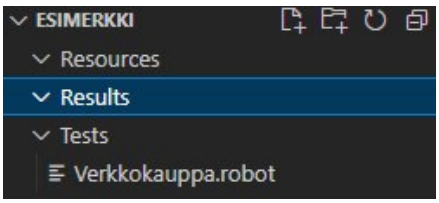
7. New credentials näkymässä lisätään kirjautumistunnukset GitLab palveluun. Kenttiin syötetään palvelun käyttäjätunnus, salasana, id tunnuksille ja kuvaus.



Testit

Ympäristön pystytyksen jälkeen voidaan siirtyä testien pariin.

1. Alkuun kannattaa suunnitella testausta, eli tuotteeseen tulee tutustua ensin.
2. Seuraavaksi voidaan kirjoittaa ylös testattavia tapahtumia ja ominaisuuksia, eli esimerkiksi ostoskoriin lisääminen tai vaikka lisää ostoskoriin -painike. Testauksen hyvä suunnittelu on tärkeää, jotta mitään oleellisia testejä ei jää pois tai vastaavasti jotain tehdään kahdesti tai muuten turhaan.
3. Seuraavassa suunnittelun vaiheessa testien vaiheita voidaan kirjata ylös. Esimerkiksi kirjautumistapahtumassa vaiheita on selaimen avaaminen, halutulle sivustolle meneminen, kirjautumissivulle meneminen, tunnuksen ja salasanan syöttö, kirjautumispainikkeen painaminen sekä kirjautumistapahtuman onnistumisen varmentaminen oikean sivun auetessa. Tämän suunnitteluvaiheen avulla on helpompi lähteä kirjoittamaan testejä.
4. Seuraavaksi voidaan aloittaa testien kirjoittaminen. Testisarjan luominen voidaan aloittaa hyvin yksinkertaisella tiedostorakenteella ja parilla testitapauksella. Tässä vaiheessa koko koodi sijaitsee yhdessä tiedostossa eli projektin ajettavassa tiedostossa, jota kutsutaan nimellä Verkkokauppa.robot. Testien kirjoitusvaihetta mallintamassa on kuvakaappauksia tiedostorakenteesta ja koodista testisarjassa, jossa on vain yksi testi. Alla on nähtävissä tämän vaiheen tiedostorakenne sekä Verkkokauppa.robot -tiedosto.

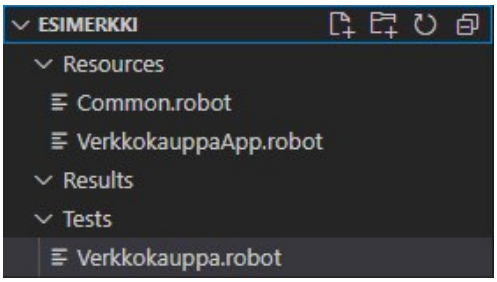


```
Verkkokauppa.robot ×
Tests > Verkkokauppa.robot > ...
1   *** Settings ***
2   Library  SeleniumLibrary
3
4   *** Test Cases ***
5   Tuotteen lisääminen koriin selaamalla onnistuu
6     [Documentation]  Tässä jotain infoa
7     [Tags]  Testi
8
9     Open Browser  https://verkkokauppa.[redacted]  chrome
10    Wait Until Page Contains  Kuvapankki
11    Input Text  xpath=/html/body/div[1]/div/section[2]/div/div[3]/div/div/div/form/div/input  Lumi
12    Press Keys  None  ENTER
13    Wait Until Page Contains  Lumiaura
14    Click Link  xpath=/html/body/main/div/h2/a
15    Wait Until Page Contains  Lisää ostoskoriin
16    Click Button  xpath=/html/body/div[2]/main/div[2]/div[2]/form/button
17    Wait Until Page Contains  "Lumiaura" on lisätty ostoskoriin.
18    Click Link  xpath=/html/body/main/div/div[1]/div[2]/div/div/a
19    Wait Until Page Contains  Kassa
20    Close Browser
```

5. Seuraavassa vaiheessa testitapaus jaetaan avainsanoihin, eli tiedostoon lisätään ***Keywords*** -osio. Avainsanoja ovat siis esimerkiksi Aloita Testi, Etsi Tuotteita ja Valitse Tuote Hakutuloksista. Näiden avainsanojen alle tulevat testitapauksen vaiheet, jotka olivat näkyvissä jo edellisessä vaiheessa. Samalla testitapauksen alle tulee nyt ainoastaan avainsanoja, kuten alla olevassa kuvassa.

```
Verkkokauppa.robot ×
Tests > Verkkokauppa.robot > Tuotteen lisääminen koriin selaamalla onnistuu
1   *** Settings ***
2   Library  SeleniumLibrary
3
4   *** Test Cases ***
5   Tuotteen lisääminen koriin selaamalla onnistuu
6     [Documentation]  Tässä jotain infoa
7     [Tags]  Testi
8     Aloita Testi
9     Etsi Tuotteita
10    Valitse Tuote Hakutuloksista
11    Lisää Tuote Ostoskoriin
12    Siirry Kassalle
13    Lopeta Testi
14
15  *** Keywords ***
16  Aloita Testi
17    Open Browser  about:blank  chrome
18
19  Etsi Tuotteita
20    Go To  https://verkkokauppa.saimaanwebpalvelut.zoner.dev/
21    Wait Until Page Contains  Kuvapankki
22    Input Text  xpath=/html/body/div[1]/div/section[2]/div/div[3]/div/div/div/form/div/input  Lumi
23    Press Keys  None  ENTER
24    Wait Until Page Contains  Lumiaura
25
26  Valitse Tuote Hakutuloksista
27    Click Link  xpath=/html/body/main/div/h2/a
28    Wait Until Page Contains  Lisää ostoskoriin
29
30  Lisää Tuote Ostoskoriin
31    Click Button  xpath=/html/body/div[2]/main/div[2]/div[2]/form/button
32    Wait Until Page Contains  "Lumiaura" on lisätty ostoskoriin.
33
34  Siirry Kassalle
35    Click Link  xpath=/html/body/main/div/div[1]/div[2]/div/div/a
36    Wait Until Page Contains  Kassa
37
38  Lopeta Testi
39    Close Browser
```

6. Seuraavassa vaiheessa luodaan Resources -kansion alle tiedostot Common.robot sekä VerkkokauppaApp.robot, kuten alla olevassa kuvassa.



7. Aiemmin kokonaan Verkkokauppa.robot -tiedostoon edellisessä vaiheessa luodut avainsanat on sijoitetaan Common.robot sekä VerkkokauppaApp.robot -tiedostoihin. Ajettavaan Verkkokauppa.robot -tiedostoon lisätään Test Setup ja Test Teardown, jotka osoitetaan Common.robot -tiedostoon, jossa on Aloita Testi ja Lopeta Testi -avainsanat. Testitapauksen alla avainsanojen eteen laitetaan VerkkokauppaApp, sillä siinä tiedostossa kyseiset avainsanat sijaitsevat. Lisäksi Verkkokauppa -tiedoston alkuun lisätään resurssien linkitykset.

```
Verkkokauppa.robot ×   VerkkokauppaApp.robot   Common.robot
Tests > Verkkokauppa.robot > Tuotteen lisääminen koriin selaamalla onnistuu
1  *** Settings ***
2  Resource  ../Resources/VerkkokauppaApp.robot
3  Resource  ../Resources/Common.robot
4  Test Setup  Common.Aloita Testi
5  Test Teardown  Common.Lopeta Testi
6
7  *** Test Cases ***
8  Tuotteen lisääminen koriin selaamalla onnistuu
9    [Documentation]  Tässä jotain infoa
10   [Tags]  Testi
11   VerkkokauppaApp.Etsi Tuotteita
12   VerkkokauppaApp.Valitse Tuote Hakutuloksista
13   VerkkokauppaApp.Lisää Tuote Ostoskoriin
14   VerkkokauppaApp.Siirry Kassalle
```

8. Esimerkiksi Etsi Tuotteita löytyy VerkkokauppaApp.robot -tiedostosta. Kyseinen avainsana sisältää esimerkiksi verkkokaupan sivustolle siirtymisen, eli Go To.

```
Verkkokauppa.robot   VerkkokauppaApp.robot ×   Common.robot
Resources > VerkkokauppaApp.robot > Siirry Kassalle
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Etsi Tuotteita
6    Go To  https://verkkokauppa.[redacted]
7    Wait Until Page Contains  Kuvapankki
8    Input Text  xpath=/html/body/div[1]/div/section[2]/div/div[3]/div/div/div/form/div/input  Lumi
9    Press Keys  None  ENTER
10   Wait Until Page Contains  Lumiaura
11
12 Valitse Tuote Hakutuloksista
13   Click Link  xpath=/html/body/main/div/h2/a
14   Wait Until Page Contains  Lisää ostoskoriin
15
16 Lisää Tuote Ostoskoriin
17   Click Button  xpath=/html/body/div[2]/main/div[2]/div[2]/form/button
18   Wait Until Page Contains  "Lumiaura" on lisätty ostoskoriin.
19
20 Siirry Kassalle
21   Click Link  xpath=/html/body/main/div/div[1]/div[2]/div/div/a
22   Wait Until Page Contains  Kassa
```

9. Common.robot puolestaan sisältää testitapauksen aloituksen ja lopetuksen.

```
Verkkokauppa.robot    VerkkokauppaApp.robot    Common.robot ×
Resources > Common.robot > Lopeta Testi
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Aloita Testi
6    Open Browser  about:blank  chrome
7
8  Lopeta Testi
9    Close Browser
```

10. Seuraavaksi luodaan page object eli sivun elementti -tiedostot PO-kansioon, eli aloitetaan luomalla PO-kansio Resources-kansion alle.
11. Jokainen testitapauksessa esiintyvä sivu sekä navigointipalkki saavat oman sivun elementti -tiedostonsa, eli esimerkkitapauksessa sivun elementit ovat Etusivu, Hakutulos, Kassa, Navigointi, Ostoskori sekä Tuote. Luodaan siis tässä vaiheessa tarvittavat tiedostot.

```
ESIMERKKI
  Resources
    PO
      Etusivu.robot
      Hakutulos.robot
      Kassa.robot
      Navigointi.robot
      Ostoskori.robot
      Tuote.robot
    Common.robot
    VerkkokauppaApp.robot
  Results
  Tests
    Verkkokauppa.robot
```

12. VerkkokauppaApp.robot -tiedostossa viime vaiheessa sijainneet testien vaiheet siirretään oikeisiin sivun elementti -tiedostoihin sen mukaan millä sivulla nämä tapahtumat ovat. Esimerkiksi avainsana Lataa löytyy Etusivu.robot -tiedostosta, ja tämä avainsana sisältää verkkokaupan sivun avaamisen.

```
kauppaApp.robot    Common.robot    Ostoskori.robot    Kassa.robot    Etusivu.robot ×
Resources > PO > Etusivu.robot > Varmenna Sivun Lataus
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Lataa
6    Go To  https://verkkokauppa.[redacted]
7
8  Varmenna Sivun Lataus
9    Wait Until Page Contains  Kuvapankki
```

13. VerkkokauppaApp.robot -tiedostoon sijoitetaan avainsanojen alle uusia avainsanoja sekä sen tiedoston nimi, mistä tämä avainsana löytyy. Esimerkiksi

Etsi Tuotteita avainsanan alla on Etusivu.Lataa, joka löytyy Etusivu.robot -tiedostosta. Lisäksi asetuksiin lisätään resurssina linkitys jokaiseen aiemmin luotuun sivun elementti -tiedostoon.

```
VerkkokauppaApp.robot ×   Common.robot   Ostoskori.robot
Resources > VerkkokauppaApp.robot > ...
1   *** Settings ***
2   Library  SeleniumLibrary
3   Resource  ../Resources/PO/Etusivu.robot
4   Resource  ../Resources/PO/Hakutulos.robot
5   Resource  ../Resources/PO/Kassa.robot
6   Resource  ../Resources/PO/Navigointi.robot
7   Resource  ../Resources/PO/Ostoskori.robot
8   Resource  ../Resources/PO/Tuote.robot
9
10  *** Keywords ***
11  Etsi Tuotteita
12    Etusivu.Lataa
13    Etusivu.Varmenna Sivun Lataus
14    Navigointi.Anna hakusana
15    Navigointi.Lähetä hakusana
16    Hakutulos.Varmenna Sivun Lataus
17
18  Valitse Tuote Hakutuloksista
19    Hakutulos.Paina Hakutulos Linkkiä
20    Tuote.Varmenna Sivun Lataus
21
22  Lisää Tuote Ostoskoriin
23    Tuote.Paina Lisää Ostoskoriin Painiketta
24    Ostoskori.Varmenna Sivun Lataus
25
26  Siirry Kassalle
27    Ostoskori.Paina Kassa Linkkiä
28    Kassa.Varmenna Sivun Lataus
```

14. Lisätään Verkkokauppa.robot -tiedostoon variables- eli muuttujatosio, johon tässä esimerkkitapauksessa lisättiin selaimelle, sivuston urlille sekä hakutermille muuttujat ja niiden arvot. Tällöin ne on helppo muuttaa halutessaan.

```
Verkkokauppa.robot ×   Common.robot   Etusivu.robot   Navigointi.robot
Tests > Verkkokauppa.robot > ${HAKUTERMI}
1   *** Settings ***
2   Resource  ../Resources/VerkkokauppaApp.robot
3   Resource  ../Resources/Common.robot
4   Test Setup  Common.Aloita Testi
5   Test Teardown  Common.Lopeta Testi
6
7   *** Variables ***
8   ${SELAIN} =  Chrome
9   ${URL} =  https://verkkokauppa.[redacted]
10  ${HAKUTERMI} =  Lumi
11
12  *** Test Cases ***
13  Tuotteen lisääminen koriin selaamalla onnistuu
14    [Documentation]  Tässä jotain infoa
15    [Tags]  Testi
16    VerkkokauppaApp.Etsi Tuotteita
17    VerkkokauppaApp.Valitse Tuote Hakutuloksista
18    VerkkokauppaApp.Lisää Tuote Ostoskoriin
19    VerkkokauppaApp.Siirry Kassalle
```

15. Seuraavaksi muutetaan muuttujat oikein myös tiedostoihin, joissa ne sijaitsevat, eli selainmuuttujan tapauksessa Common.robot -tiedostoon tehdään muutos. Sen sijaan että koodirivillä lukee selaimen nimi, esimerkiksi Chrome, asetetaan siihen muuttuja ${SELAIN}, johon on annettu arvo Verkkokauppa.robot -tiedostossa.

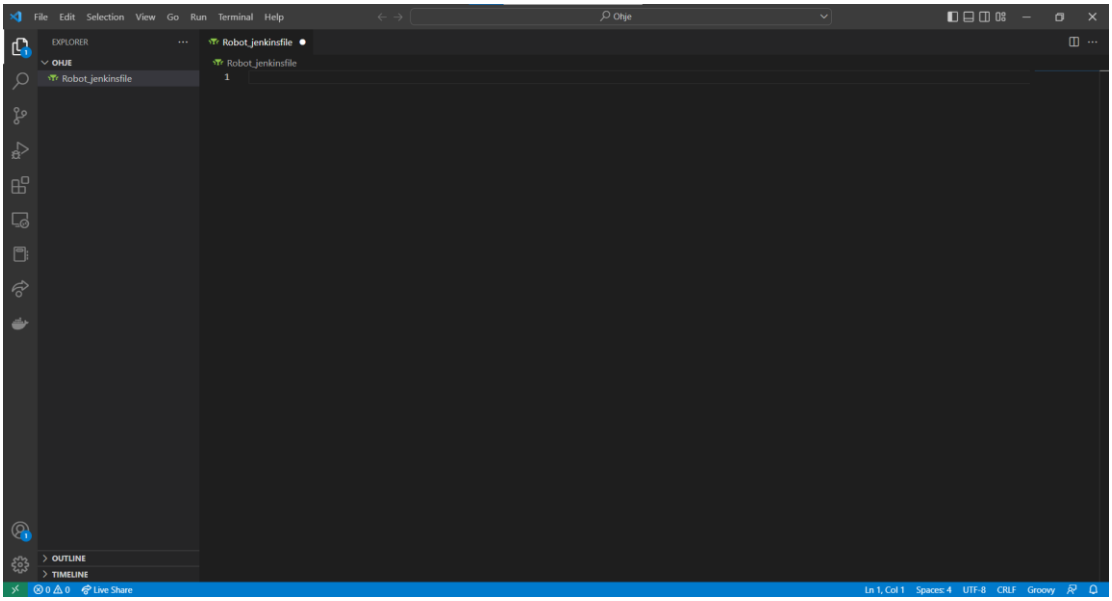
```
Verkkokauppa.robot   Common.robot ×   Etusivu.robot
Resources > Common.robot > Aloita Testi
1  *** Settings ***
2  Library  SeleniumLibrary
3
4  *** Keywords ***
5  Aloita Testi
6    Open Browser  about:blank  ${SELAIN}
7
8  Lopeta Testi
9    Close Browser
```

Testien ajo CI-ympäristössä

Ennen kuin testejä aletaan ajamaan, on testisarjan versionhallinta oltava ajantasalla jollakin koodinhallinta-alustalla, joka tässä tapauksessa on GitLab. Lisäksi luodaan niitä varten jenkinsfile, johon määritetään mitä putkessa ajetaan.

Jenkinsfilen luonti

1. Luodaan tiedosto nimeltä robot_jenkinsfile Visual Studio Codessa ja asetetaan Language Mode -asetus groovyksi.



2. Tämän jälkeen lisätään tiedostoon Pipeline {} -osio, jonka sisään määritetään putken määritelmä ja asetukset.

```
Robot_jenkinsfile
1  pipeline {
2
3  }
```

3. Ensimmäisenä määritetään millä noodilla putkea ajetaan määrittämällä sille agent osio ja lisäämällä halutun noodin nimi tai nimike. Tässä tapauksessa käytetään aiemmin luodun noodin nimikettä, joka on Robot.

```
Robot_jenkinsfile
1  pipeline {
2      agent {
3          label 'Robot'
4      }
5  }
```

4. Noodin määrityksen jälkeen määritetään options -osio, jossa voidaan määrittää putken asetuksia, kuten esimerkiksi säilytettävien ajojen määrä ja maksimi ajoaika.

```
Robot_jenkinsfile
1  pipeline {
2      agent {
3          label 'Robot'
4      }
5      options {
6          buildDiscarder(logRotator(numToKeepStr: '10', artifactNumToKeepStr: '10'))
7          timeout(time: 1, unit: 'HOURS')
8      }
9  }
```

5. Määritysten jälkeen lisätään Stages -osio, jonka sisään määritellään putken työvaiheet.

```
Robot_jenkinsfile
1  pipeline {
2      agent {
3          label 'Robot'
4      }
5      options {
6          buildDiscarder(logRotator(numToKeepStr: '10', artifactNumToKeepStr: '10'))
7          timeout(time: 1, unit: 'HOURS')
8      }
9      stages {
10
11     }
12 }
```

6. Stages -osion alle määritetään yksittäisiä stage -osia, jotka sisältävät steps -osion johon määritetään mitä kyseisessä vaiheessa suoritetaan. Ensimäisessä vaiheessa työkansio siivotaan ja testit haetaan versionhallinasta

```
Robot_jenkinsfile
1  pipeline {
2      agent {
3          label 'Robot'
4      }
5      options {
6          buildDiscarder(logRotator(numToKeepStr: '10', artifactNumToKeepStr: '10'))
7          timeout(time: 1, unit: 'HOURS')
8      }
9      stages {
10         stage('Git checkout') {
11             steps {
12                 // Siivoaa edellisen ajon tulokset työkansiosta
13                 cleanWs()
14                 // Hakee Gitlabistä main haarasta verkkokaupan testit
15                 git branch: 'main', credentialsId: 'Gitlab' url: 'https://gitlab.com/tiiaheino/verkkokauppa.git'
16             }
17         }
18     }
19 }
```

7. Ensimmäisen vaiheen jälkeen lisätään toinen vaihe joka suorittaa versionhallinnasta haetut testit.

```
Robot_jenkinsfile
pipeline {
    agent {
        label 'Robot'
    }
    options {
        buildDiscarder(logRotator(numToKeepStr: '10', artifactNumToKeepStr: '10'))
        timeout(time: 1, unit: 'HOURS')
    }
    stages {
        stage('Git checkout') {
            steps {
                // Siivoaa edellisen ajon tulokset työkansiosta
                cleanWs()
                // Hakee Gitlabistä main haarasta verkkokaupan testit
                git branch: 'main', credentialsId: 'Gitlab' url: 'https://gitlab.com/tiiaheino/verkkokauppa.git'
            }
        }
        stage('Run tests') {
            steps {
                script {
                    // Vaihtaa kansion työkansion juuresta Tests kansioon
                    dir("Tests") {
                        // Ajaa haetut robotti testit
                        bat "robot -d ${workspace}\\Results Verkkokauppa.robot"
                    }
                }
            }
        }
    }
}
```

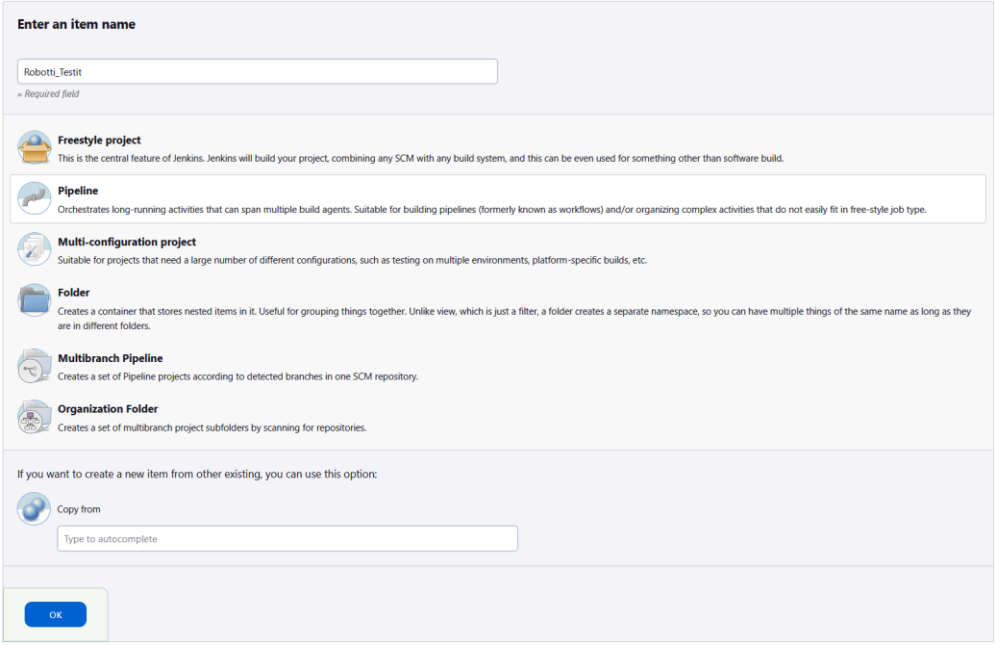
8. Tämän jälkeen lisätään vielä yksi vaihe ja post -osio. Viimeisessä vaiheessa testiajon tulokset varastoidaan Jenkinsiin. Jälkiosio suoritetaan vain jos testiajossa tapahtuu virhe ja se suorittaa saman varastoinnin kuin viimeinen vaihe, mutta vain silloin kun tapahtuu jokin virhe ja ajo keskeytyy sen takia.

```
Robot_jenkinsfile
Robot_jenkinsfile
        stage('Git checkout') {
            steps {
                // Siivoaa edellisen ajon tulokset työkansiosta
                cleanWs()
                // Hakee Gitlabistä main haarasta verkkokaupan testit
                git branch: 'main', credentialsId: 'Gitlab' url: 'https://gitlab.com/tiiaheino/verkkokauppa.git'
            }
        }
        stage('Run tests') {
            steps {
                script {
                    // Vaihtaa kansion työkansion juuresta Tests kansioon
                    dir("Tests") {
                        // Ajaa haetut robotti testit
                        bat "robot -d ${workspace}\\Results Verkkokauppa.robot"
                    }
                }
            }
        }
        stage('Archive artifacts') {
            steps {
                script {
                    // Varastoi testi tulokset ajon päätteeksi
                    archiveArtifacts artifacts: 'Results/*.*', excludes: 'Results/*.log'
                }
            }
        }
    }
    post {
        failure {
            archiveArtifacts artifacts: 'Results/*.*', excludes: 'Results/*.log'
        }
    }

}
```

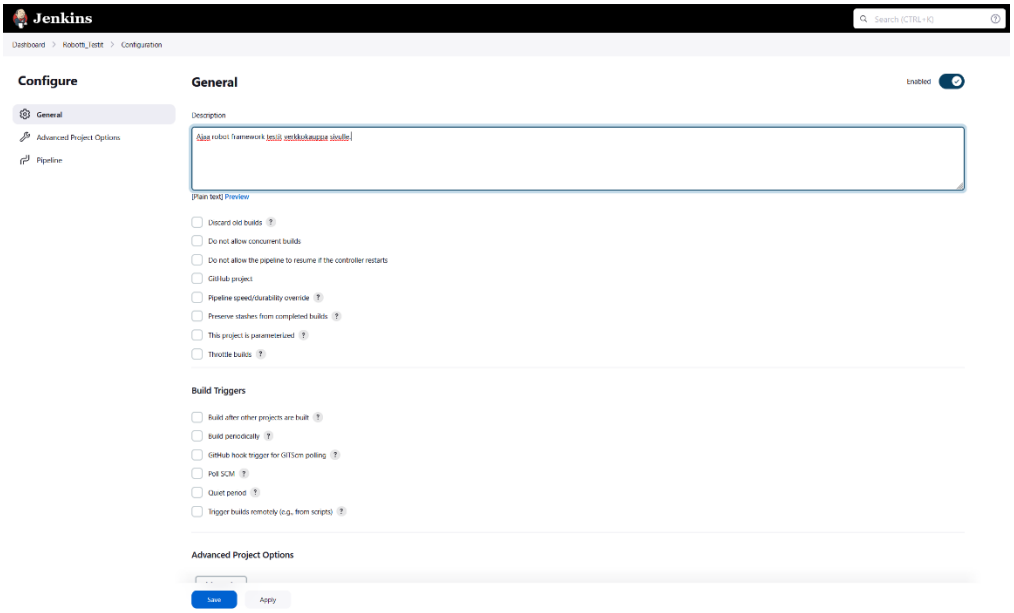
9. Luotu jenkinsfile tulee tallentaa versionhallintaan, josta Jenkins hakee sen kun sitä käytetään.

Jenkins Pipelinen luonti

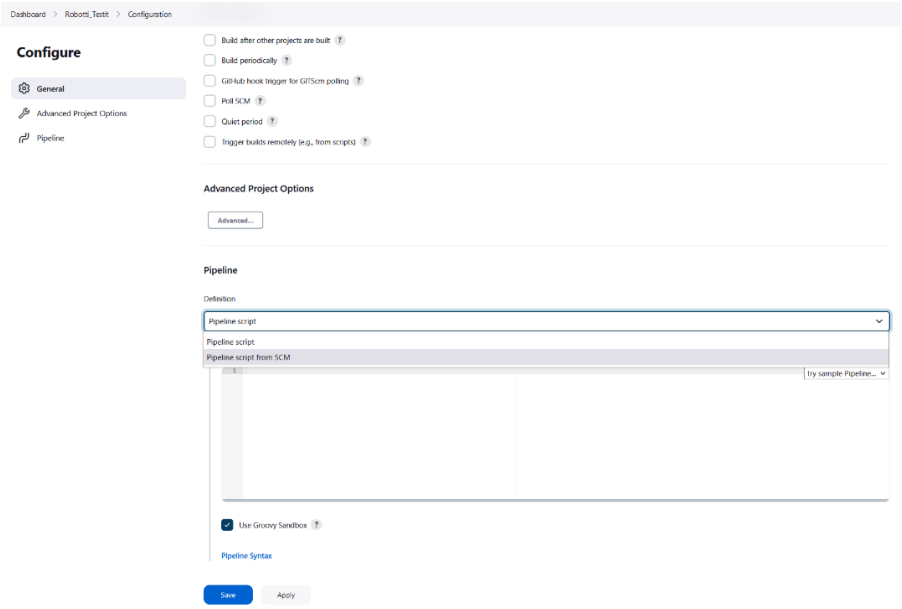
1. Avataan selaimessa localhost:8080, jos se ei ole jo auki.
2. Jenkinsin etusivun vasemmasta valikosta valitaan New Item -painike.
3. New Item -näkymässä annetaan nimi Robotti_Testit ja valitaan Pipeline -vaihtoehto. Tämän jälkeen painetaan OK painiketta.



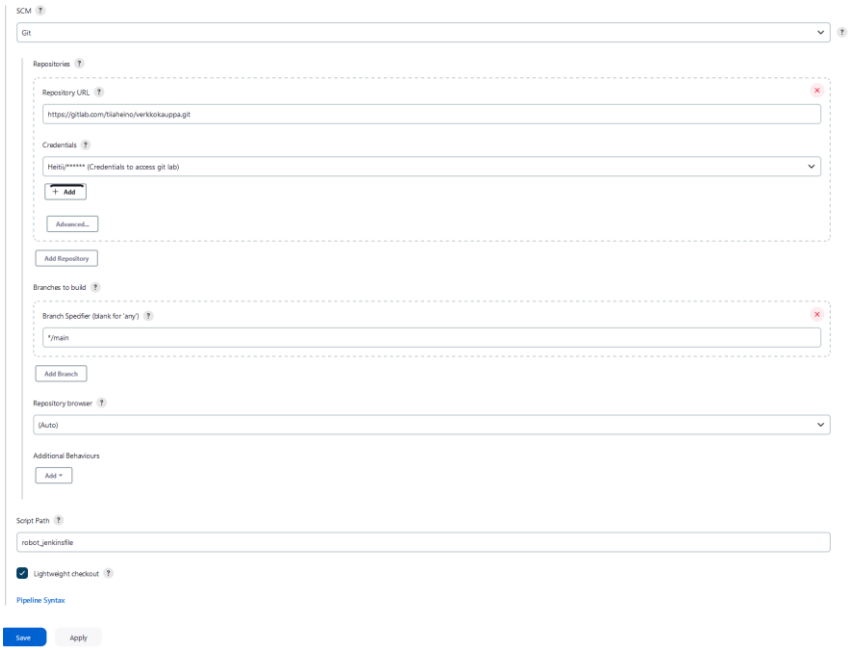
4. Putken asetusten määrityssivun aukeaa, jolloin voidaan määrittää putkelle asetuksia, kuten esimerkiksi mistä putken jenkinsfile haetaan. Putkelle voi myös asettaa kuvauksen, joka kertoo mitä putki tekee.



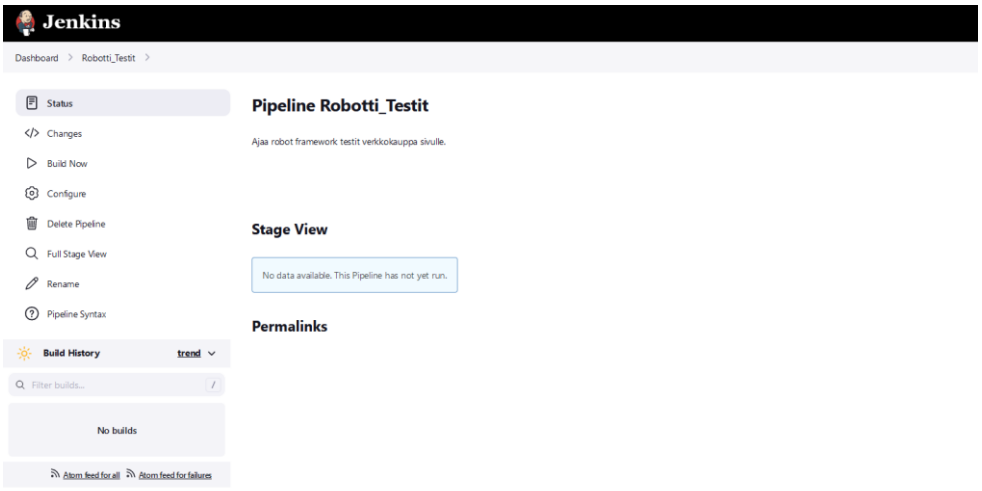
5. Pipeline -osion alta voi vaihtaa mistä putken määritys noudetaan. Oletuksena putkeen on määritetty, että se ajetaan Script -kentästä löytyvällä koodilla, mutta Definition valikosta se voidaan vaihtaa hakemaan määritelmä versionhallinasta valitsemalla Pipeline script from SCM.



6. Tämän jälkeen valitaan SCM valikosta Git ja asetetaan tarvittavat arvot, että Jenkins voi ottaa yhteyden versionhallintaan.

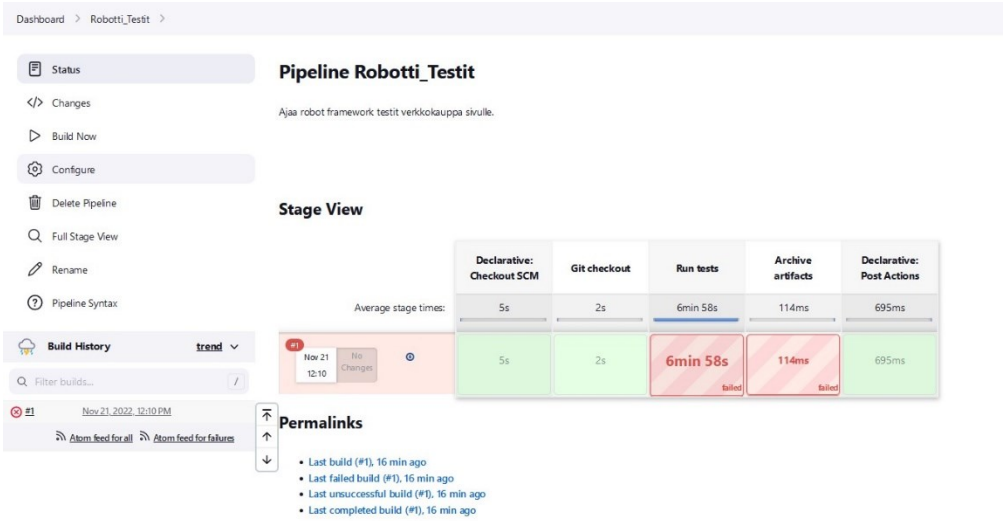


7. Kun asetukset on asetettu voidaan painaa Save-painiketta ja aloittaa testien ajo painamalla Build Now-painiketta avautuvan näkymän vasemmanlaidan valikosta. Kun käännös on valmis, on mahdollista tarkastella tuloksia testiloki- ja -raporttitiedostojen avulla. Mikäli tiedostot eivät aukea suoraan Jenkinsin kautta, ne on mahdollista avata tietokoneen tiedostoista.

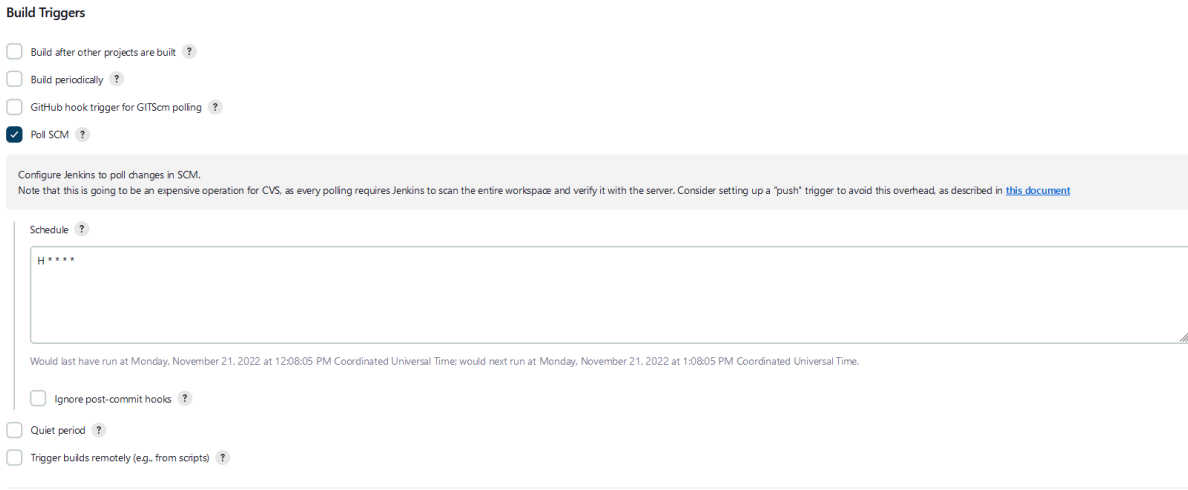


Muutosten automaattinen seuraus

1. Avataan aiemmin luodon putken asetukset painamalla Configure-painiketta putken näkymästä.



2. Asetusnäkymässä valitaan Build Triggers -osion alta Poll SCM ja annetaan avautuvaan kenttään arvoksi H * * * *. Tällöin Jenkins tarkistaa kerran tunnissa onko ajettavaan versionhallintaan tullut muutoksia. Jos muutoksia on tullut, testiajo käynnistyy.



3. Kun asetukset on asetettu paina Save -painiketta.