Opinnäytetyö (AMK)

Tietotekniikka

Ohjelmistoliiketoiminta

2016

Teemu Kuutti

WEB-SOVELLUSTEN TESTIYMPÄRISTÖN KEHITTÄMINEN SEKÄ TESTIAUTOMAATIO

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Ohjelmistoliiketoiminta

2016 | 37 Sivua

Teemu Kuutti

WEB-SOVELLUSTEN TESTIYMPÄRISTÖN KEHITTÄMINEN SEKÄ TESTIAUTOMAATIO

Tämän opinnäytetyön tavoitteena oli kehittää ja tehostaa PerkinElmer-konserniin kuuluvan Wallac Oy:n sovelluskehityksen yksikkötestausprosessin automaatiota selvittämällä onko mahdollista tehdä yksikkötestien rinnakkaisajoa. Nykyinen testausympäristö koostuu Microsoftin tuoteperheen tuotteista, kuten Windows Server, Team Foundation Server, SQL Server, System Center sekä avoimeen lähdekoodiin perustuva selaimen automatisointityökalu nimeltä Selenium WebDriver.

Työn keskeisenä aiheena on System Centerin komponentti nimeltä Virtual Machine Manager, jolla pystytään luomaan perinteisten testiympäristöjen lisäksi SCVMM-testiympäristöjä. Lisäämällä monta virtuaalikonetta SCVMM-testiympäristöön mahdollistuu yksikkötestien ajo monessa virtuaalikoneessa yhtä aikaa säästäen näin aikaa ja rahaa.

Työssä luotiin kaksi eri SCVMM-testiympäristöä, joihin molempiin luotiin yksi virtuaalikone. Testaus tehtiin rinnakkaisajona ja se suoritettiin onnistuneesti. Testi oli hyvin minimaalinen ja tehtiin tätä opinnäytetyötä varten, eikä se liity mitenkään Wallacin valmistamiin ohjelmistoihin. Se todisti vain teoriatasolla, että testiajot voidaan suorittaa monessa virtuaaliympäristössä ja -koneessa yhtä aikaa. Koska Wallacin valmistamat ohjelmistot ovat niin suuria, SCVMM -ympäristön integroiminen osaksi automatisoitua testiajoa vaatii tarkempaa suunnittelua ja valmistelua. Myös fyysisen palvelimen järjestelmävaatimukset tulee ottaa huomioon otettaessa käyttöön suurempia testiympäristökokonaisuuksia.

ASIASANAT:

SCVMM, TFS, Virtual Machine Manager, Yksikkötesti, Virtuaalikone, Palvelin, Hyper-V, SQL

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Software Business

2016 | 37 pages

Teemu Kuutti

UPGRADING A WEB APPLICATION TESTING ENVIRONMENT AND TEST AUTOMATION

The aim of this study was to develop and enhance the automated unit testing process in software development in Wallac Oy which belongs to PerkinElmer corporated. This was achieved by investigating is it possible to carry out parallel unit testing. The current testing environment consists of the Microsoft family of products, such as Windows Server, Team Foundation Server, SQL Server, System Center as well as an open-source browser automation tool called Selenium WebDriver.

The essential focus of this work is a component of System Center called Virtual Machine Manager which allows the creation of SCVMM test environments in addition to standard environments. Parallel testing can be achieved by adding multiple virtual machines in SCVMM environment. Thus saving time and money.

In this work two different SCVMM environments with one virtual machine were created. Testing was done in parallel and it ran successfully. The test itself was very minimal and it was created specifically for this thesis and it has nothing to do with the softwares that Wallac manufactures. It proved only in theory that tests can be run in multiple virtual environments and virtual machines at the same time. Integrating an SCVMM environment as part of automated testing needs a more detailed planning and preparation because the softwares Wallac manufactures are so large. The hardware requirements of a physical server should also be taken into consideration when deploying larger test environments.

KEYWORDS:

SCVMM, TFS, Virtual Machine Manager Unit test, Server, Hyper-V, SQL

SISÄLTÖ

LIITTEET

KOODIT

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET TAI SANASTO

Build	Ohjelmakooste, Koontiversio
DevOps	Development & Operation - ketterä toimintamalli sähköisten palveluiden tuottamiseen.
Domain	Verkkotunnus, esim perkinelmer.net
Host	Isäntä
Host group	Isäntäryhmä
HTTP	Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon (HyperText Transfer Protocol)
HTTPS	Suojattu tiedonsiirtoprotokolla (Hypertext Transfer Protocol Secure)
Hyper-V	Microsoftin virtualisointiohjelma
SCVMM	System Center Virtual Machine Manager
Server	Palvelin
SQL	Structured Query Language
TFS	Team Foundation Server
Unit test	Yksikkötesti
.NET	Microsoftin kehittämä ohjelmointikehys

1 JOHDANTO

Ohjelmistokehittäjät tekevät ohjelmiston, ja testaajat todentavat laadun. Mutta mitä laatu oikeastaan edes on? Software Testing Fundamentalsin mukaan ohjelmiston laadulla on monia ulottuvuuksia, joista esimerkkeinä mainittakoon toiminnallisuus, eli ohjelmiston kyky suorittaa annetut tehtävät määritelmänsä mukaisesti; luotettavuus, eli ohjelmiston kyky suorittaa vaaditut toiminnot asetetuissa olosuhteissa määrätyn ajan ilman virheitä; sekä käytettävyys, eli ohjelmiston helppokäyttöisyys. Myös testattavuutta, eli miten helppo ohjelmistoa on testata, pidetään yhtenä laadun ulottuvuutena, kuten myös ohjelmiston ylläpidettävyyttä. (Software Testing Fundamentals, 2016a.) Laadunvalvontaan ja ohjelmiston toimivuuden seurantaan käytetään usein yksikkötestejä. Yksikkötestit ovat paras tapa korjata virheet mahdollisimman aikaisessa vaiheessa. (Software Testing Fundamentals, 2016b.)

Ongelmaksi web-sovelluksissa muodostuu yleisimpänä web-selainten tiuha noin 6 viikon jaksottainen päivitystahti (The Chromium Projects, 2016), joka saattaa saada toimivaksi todetun sovelluksen toimimaan väärällä tavalla. Pitääkseen ohjelmistot laadukkaana, täytyy niitä testata usein.

Tämän opinnäytetyön tavoitteena on kehittää ja tehostaa PerkinElmer-konserniin kuuluvan Wallac Oy:n sovelluskehityksen yksikkötestausprosessin automaatiota selvittämällä onko mahdollista tehdä yksikkötestien rinnakkaisajoa. Työn keskeisenä aiheena on System Centerin komponentti nimeltä Virtual Machine Manager (VMM), jolla pystytään luomaan perinteisten testiympäristöjen lisäksi SCVMM-testiympäristöjä. Lisäämällä monta virtuaalikonetta SCVMM-testiympäristöön mahdollistuu yksikkötestien ajo samanaikaisesti monessa virtuaalikoneessa säästäen näin aikaa ja rahaa. Nykyinen testausympäristö koostuu Microsoftin tuoteperheen tuotteista, kuten Windows Server, Team Foundation Server, SQL Server, System Center sekä avoimeen lähdekoodiin perustuva selaimen automatisointityökalu nimeltä Selenium WebDriver.

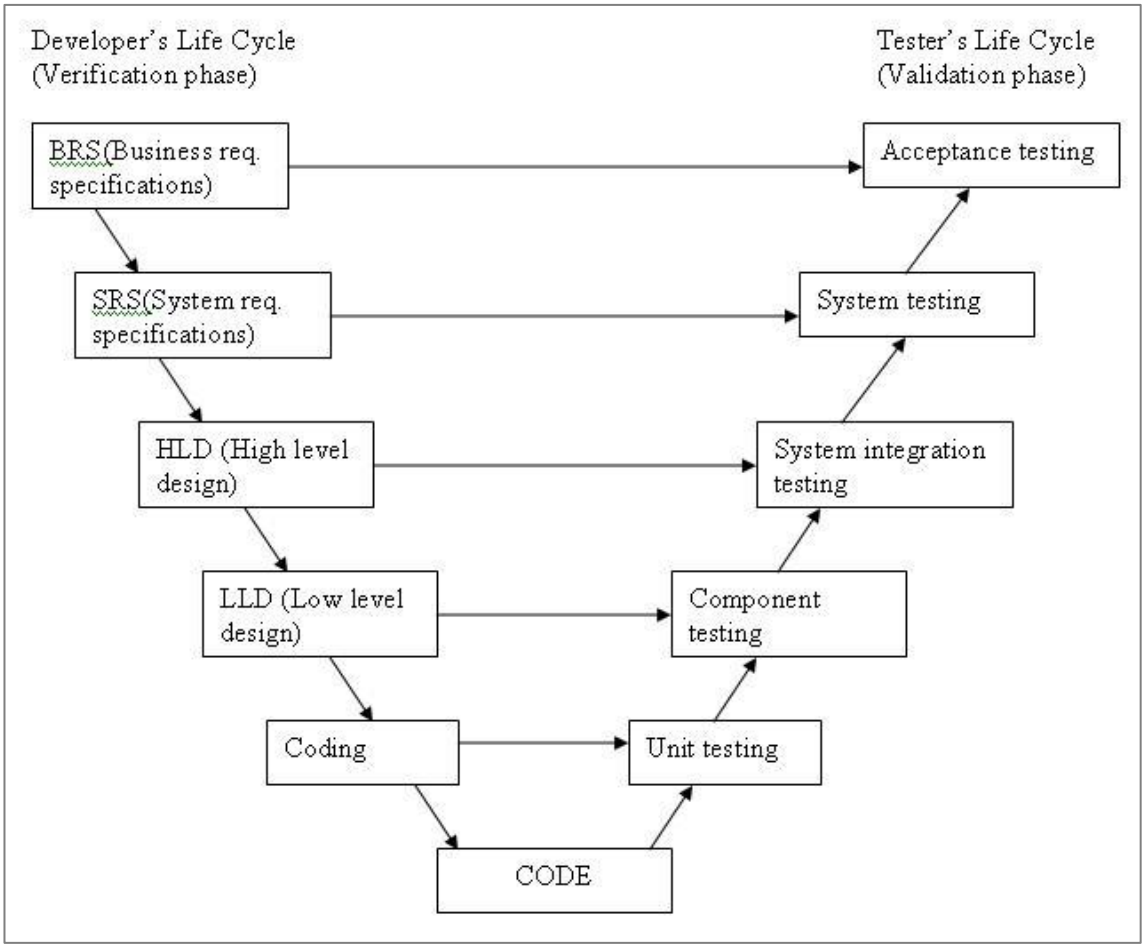
Wallacissa sovelluskehitys tapahtuu muunnellun Scrum-nimisen ketterän kehitysmallin mukaan, jossa kehitys tapahtuu noin 3 viikon mittaisissa jaksoissa, joita kutsutaan sprinteiksi. Jokaisen sprintin päätteeksi käydään läpi, mitä ollaan saatu valmiiksi, ja mitä seuraavan sprintin aikana tehdään. Jokaisen sprintin aikana on myös tarkoitus saada aloitettu ominaisuus valmiiksi ja saada se testattua. Wallacissa ohjelmistot ovat niin suuria ja hitaita testattavia, että niitä voidaan nykyisellä testiympäristöllä testata

vain kerran vuorokaudessa. Ajamalla yksikkötestit rinnakkain monella virtuaalikoneella yhtä aikaa, voidaan testausaikaa vähentää, mikä helpottaa ohjelmistokehittäjien työtä.

Työssä kerrotaan ensiksi testauksen ja testiautomaation teoriaa, että saadaan perusymmärrys testauksen toiminnasta ja tarkoituksesta. Tämän jälkeen tutustutaan Virtual Machine Manageriin ja sen ominaisuuksiin. Sitten suoritetaan Wallacin testiympäristön kehitystoimet asentamalla VMM Team Foundation Serverille (TFS:lle). Sen jälkeen luodaan VMM:llä kaksi virtuaalikonetta ja näitä virtuaalikoneita hyödyntäen luodaan Test managerilla kaksi SCVMM-testiympäristöä. Tämän jälkeen konfiguroidaan build-palvelin, jotta voidaan suorittaa koonti testikoodille ja sen yksikkötestille. Sitten tehdään testikoodi ja sille yksikkötesti voidaksemme testata ympäristöä. Lopuksi ajetaan testit rinnakkain kahdessa SCVMM-testiympäristössä käyttäen TFS:n web-portaalia.

2 TESTAUS JA TESTIAUTOMAATIO

Ohjelmistotestaus voidaan jakaa kahteen yläluokkaan, joista toinen on suorituskyvyn testaamista ja toinen toiminnallisuuden testaamista. Tämä työ keskittyy ainoastaan toiminnallisuuden testaamiseen. Toiminnallisuuden testaaminen voidaan jakaa useaan tasoon, joista ns. alin taso on yksikkötestaus. Kuvassa 1. näkyy sovelluksen elinkaarta aina suunnittelusta toteutukseen asti kuvaava V-malli, jota käytetään yleisesti sovelluksen tarkistamiseen ja validointiin. (ISTQB exam certification, 2016.)



Kuva 1. Verification and Validation model eli V-malli. (ISTQB exam certification 2016.)

2.1 Yksikkötestaus

Yksikkötestaus on ohjelmistotestauksen taso, jolla yksittäiset ohjelmiston komponentit testataan. Tarkoituksena on varmistaa, että ohjelmiston jokainen yksikkö toimii, kuten

on suunniteltu. Yksikkötesti on ohjelmiston pienin testattava osa. Yleensä se sisältää yhdestä muutamaan syötteen (input) ja vain yhden tulosteen (output). Prosessuaalisessa ohjelmoinnissa yksikkö voi olla yksittäinen ohjelma, funktio tai toiminto. Olio-ohjelmoinnissa pienin yksikkö on metodi, joka voi kuulua perus- tai super-luokkaan, perittyyn tai abstraktiin luokkaan tai pääluokan lapsiluokkaan. Yksikkötestausta tehdään ennen integraatiotestausta, jossa testataan yksikköjen toimivuutta ryhmänä. (Software Testing Fundamentals, 2016b.)

2.2 Testaamisen tärkeys

Ohjelmistojen testaus on välttämätöntä, koska ihminen tekee virheitä. Jotkin näistä virheistä ovat merkityksettömiä, mutta jotkin niistä ovat vaarallisia ja voivat koitua kalliiksi. Varsinkin silloin, kun kyseessä on terveysalan ohjelmistot. Jotkin virheet voivat tulla vääristä oletuksista, eli työtä tarkistaessa voi erittäin helposti tehdä samoja virheitä kuin työtä tehdessä. Ihminen on niin sanotusti sokea omille virheilleen, eikä välttämättä huomaa puutteita, joita on tehty tai jäänyt tekemättä. Parasta onkin, kun joku toinen henkilö tarkistaa toisen henkilön työn, koska näin virheet tulevat todennäköisemmin ilmi. (ISTQB exam certification, 2016.)

ISTQB exam certificationin mukaan tärkeimmät syyt suorittaa testaus ovat:

1. Ohjelmistoa testatessa havaitaan puutteita ja virheitä, jotka tehtiin kehitysvaiheessa.
2. Sillä voidaan varmistaa, että asiakas voi luottaa tuotteeseen ja on siihen tyytyväinen.
3. Testaaminen todentaa ja varmistaa tuotteen laadun.
4. Testaus on tarpeen, jotta voidaan palvella asiakkaita korkealaatuisilla tuotteilla tai sovelluksilla, jotka vaativat vähemmän huoltokustannuksia ja johtavat näin tarkempiin, johdonmukaisempiin ja luotettavampiin tuloksiin.
5. Testausta tarvitaan sovelluksen tai tuotteen tehokkaan suorituskyvyn varmistamiseksi.
6. On tärkeää varmistaa, että sovellus ei johda virheisiin, koska se voi koitua kalliiksi tulevaisuudessa tai myöhemmissä kehitysvaiheissa.
7. Tiivistettynä: Testausta tarvitaan, jotta yritys pysyy mukana liiketoiminnassa.

Ohjelmistojen testaus auttaa viimeistelemään sovelluksen tai tuotteen liiketoiminnan ja käyttäjien vaatimukset. On tärkeää saada hyvä kattavuus testeille, jotta ohjelmisto voidaan testata ja varmistaa, että se toimii hyvin jokaista vaatimusta kohden. (ISTQB exam certification, 2016.)

2.3 Testiautomaation hyödyt ja haitat

Saadakseen kaiken hyödyn irti yksikkötesteistä tulisi testit ajaa jokaisen buildin jälkeen. Termi ”build” tarkoittaa ohjelmiston osien eli lähdekoodin koostamista suoritettavaksi kokoonpanoksi ja se on tehtävä ennen ohjelmiston käyttöä ja testaamista. (Microsoft 2016a.) Testien automatisoinnilla tarkoitetaan yleensä ajoitettua buildia, joka ajaa siinä samalla myös yksikkötestit. Suosituksen mukaan build ja yksikkötestit tulisi ajoittaa suoritettavaksi 1–2 kertaa päivässä. Näin tehtynä testaus tapahtuu usein ja aikaisessa vaiheessa, joka helpottaa virheiden löytämistä ja korjaamista, sekä pitää lähdekoodin siistinä ja toimivana. (Marick, 2013.)

Automatisoidut testit säästävät huomattavasti työntekijän aikaa ja yrityksen rahaa. Kun mietitään esimerkiksi tilannetta, jossa joudutaan ajamaan 500 yksikkötestiä, joista jokainen vie aikaa noin 5 minuuttia, veisi kaikkien testien ajaminen yli 41 tuntia eli yli yhden työviikon.

Testiautomaatiota suositellaan käytettävän pääasiassa silloin, kun tehdään ohjelmistoja, joita tullaan kehittämään ja ylläpitämään jatkossakin useita vuosia. Saatu hyöty kasvaa myös silloin, kun tiedetään että sama testi joudutaan ajamaan useita kertoja, esimerkiksi eri selaimilla tai eri käyttöjärjestelmillä. (Marick, 2013.)

Jos tiedetään jo etukäteen, että kehitettävä sovellus tulee muuttumaan paljon lähitulevaisuudessa, ei testien automatisoimista kannatta jättää tekemättä, vaan siirtää myöhemmäksi. Tapauksissa, joissa testi ajetaan vain muutamia kertoja ja sen ajaminen manuaalisesti olisi nopeampaa kuin sen automatisointi, ei automaatiota kannata käyttää. (Software Testing Fundamentals, 2016b.)

2.4 Selenium WebDriver

Selenium WebDriver on avoimeen lähdekoodiin perustuva verkkoselaimen automatisointityökalu, jota käytetään pääasiassa web-sovellusten testaamiseen ja sen automa-

tisointiin. Selenium WebDriver koostuu kahdesta eri komponentista, Seleniumista ja WebDriverista. WebDriver korjaa Seleniumin puutteita tämän javascript-hiekkalaatikon suhteen keskustelemalla suoraa selaimen kanssa, ja Selenium korjaa WebDriverin puutteita tukemalla laajemmin eri selaimia. (Selenium, 2016.)

3 TESTIYMPÄRISTÖ

Testiympäristö koostuu osista, jotka tukevat testaamista ohjelmiston, laitteiston ja tietoverkon kannalta. Testiympäristön asetusten tulee vastata tuotannossa olevaa ympäristöä mahdollisimman hyvin, jotta saadaan eriteltyä ympäristöön ja asetuksiin koostuvat virheet mahdollisimman hyvin. (Tutorialspoint, 2016.)

3.1 Virtual Machine Manager

Virtual Machine Manager on Microsoft System Centerin komponentti virtuaalisten ympäristöjen luontiin, konfigurointiin ja hallintaan. Se mahdollistaa muun muassa virtuaalisten SCVMM-ympäristöjen luonnin, mikä nopeuttaa testaamista mahdollistamalla rinnakkaistestiajot useassa testiympäristössä ja -koneessa. Testien kannalta fyysisen palvelinkoneen on oltava tarpeeksi tehokas, vaikka VMM ei suuria tehoja vaadi. Kuvissa 2. ja 3. on esitetty VMM:n suositellut järjestelmä- ja tehovaatimukset, kun se pyörittää virtuaalikoneita 150:een kappaleeseen asti tai enemmän. Suositellut vaatimukset prosessorin tehossa ja RAM-muistin määrässä ovat samaa luokkaa tai pienemmät kuin uusimmissa tietokoneissa nykypäivänä. Minimivaatimukset ovat tästä vielä alhaisemmat. (Microsoft, 2016f.)

Managing up to 150 hosts		
Hardware component	**Minimum**	**Recommended**
Processor	Pentium 4, 2 gigahertz (GHz) (x64)	Dual-processor, dual-core, 2.8 GHz (x64) or greater
RAM	4 gigabytes (GB)	4 GB
Hard disk space, without a local VMM database	2 GB	40 GB
Hard disk space, with a local, full version of Microsoft SQL Server	80 GB	150 GB

Kuva 2. Virtual Machine Manager -järjestelmävaatimukset 150 isäntäkoneeseen asti.

Managing more than 150 hosts		
Hardware component	**Minimum**	**Recommended**
Processor	Pentium 4, 2.8 GHz (x64)	Dual-processor, dual-core, 3.6 GHz or greater (x64)
RAM	4 GB	8 GB
Hard disk space	10 GB	50 GB

Kuva 3. Virtual Machine Manager -järjestelmävaatimukset yli 150 isäntäkoneella.

System Center 2012 R2 -versiosta alkaen Virtual Machine Managerilla pystyy luomaan ja hallinnoimaan kahdentyyppisiä virtuaalikoneita. Aiemmin tavallisina tunnetut virtuaalikoneet ovat nyt saaneet sukupolven 1 (generation 1) virtuaalikone -nimityksen. Uudemmat 2012 R2 Hyper-V:n mukana esitellyt virtuaalikoneet ovat saaneet sukupolven 2 (generation 2) virtuaalikone -nimityksen. Virtuaalikoneen mallia luotaessa täytyy valita näistä toinen. Suurin ero näiden kahden tyypin välillä havaitaan silloin, kun katsotaan, mitä käyttöjärjestelmää isäntäpalvelin käyttää, ja mitä käyttöjärjestelmää virtuaalikoneelle ollaan asentamassa. Tällä hetkellä sukupolven 2 virtuaalikoneita tukee vain Windows Server 2012 R2 -versio. (Microsoft 2016h; 2016i.)

VMM tarvitsee tietokantapalvelimen, ja siksi vaatii jonkin Microsoft SQL Serverin asentamista Windows serverille, ennen kuin VMM voidaan asentaa. VMM:n asentaminen vaatii myös, että tietokoneelle on asennettu .NET 3.5 -ohjelmointikehys. Se on Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoft Visual Studio -ympäristössä kehitetyt ohjelmistot käyttävät. VMM vaatii myös, että asennettuna on Windows Assessment and Deployment Kit (Windows ADK) for Windows 8.1. Se on kokoelma työkaluja, joiden avulla voidaan muokata, määrittää ja ottaa käyttöön Windows-käyttöjärjestelmiä uusiin tietokoneisiin, ja sen saa ladattua ilmaiseksi Microsoftin omilta nettisivuilta. (Microsoft 2016b; 2016j.)

Virtuaalikone (virtual machine, VM) toimii kuin oikea fyysinen tietokone, mutta sen toiminta-alusta on joko virtualisointiohjelma tietokoneella tai palvelimella. Virtuaalikoneen malli, eli template, puolestaan on kuin virtuaalikone, jonka tunnistetiedot on riisuttu. Sitä voidaan käyttää nopeuttamaan uuden virtuaalikoneen luomisprosessia. (Microsoft 2009.)

3.2 Team Foundation Server

Team Foundation Server (TFS) on Microsoftin tuote, joka tarjoaa erilaisia työkaluja lähdekoodin hallintaan (esimerkiksi raportointi, vaatimusmäärittelyt, projektinhallinta, automatisoidut buildit, testiympäristön hallinta, testauksen- ja julkaisun hallintaominaisuudet). Se kattaa koko sovelluksen elinkaaren ja mahdollistaa Development & Operation (DevOps) -ominaisuudet.

Team explorer on osa TFS:n asennusta, mutta se on myös ladattavissa erillisenä lisäosana Visual Studioon. Team exploreria käytetään TFS:n ominaisuuksien integroimiseksi Visual Studioon. Sen avulla voidaan tarkastella ja hallita lähdekoodin yksittäisiä osia sekä virheitä, työtehtäviä ja dokumentteja sekä luoda TFS-tilastoja. (Microsoft, 2016g.)

3.3 Test manager

Test manager on Microsoftin kehittämä lisäosa lähdekoodin testaamista varten, joka voidaan asentaa vain Visual Studio 2015 Enterprise -version tai Visual Studio Test Professionalin mukana. Test manager on jaettu kahteen eri näkymään: Testing centeriin ja Lab centeriin. Testing Centeriä käytetään testien ja testisuunnitelmien luomiseen ja hallinnoimiseen sekä tulosten tarkasteluun. Lab centerillä luodaan ja hallinnoidaan testiympäristöjä, joissa test centerillä luodut testit ajetaan. (Microsoft, 2012b.)

SCVMM-testiympäristön luomiseen vaaditaan Team project collectioniin vähintään yhden test controllerin asentamista ja konfiguroimista. Test controllerin voi asentaa Windows 7- tai Windows Server 2008 -käyttöjärjestelmille, sekä kaikille niitä uudemmille. Kone, jolle test controller asennetaan, voi olla joko fyysinen tai virtuaalinen, mutta sen täytyy olla samassa domainissa kuin testiympäristö on. Se ei kuitenkaan saa olla samalla koneella, jolla domain controller on. Test controller suositellaan asennettavan koneelle, joka on aina päällä, koska sitä tarvitaan testiympäristöä luotaessa sekä testejä ajettaessa. (Microsoft, 2016h.)

3.4 Build-palvelin

Esivaatimukset Build-palvelimen konfigurointiin ovat PowerShell tai komentorivityökalu, sekä palvelimella jokin .NET-versio. Jos build-palvelimen käyttöjärjestelmänä käytetään esimerkiksi Windows Server 2012 R2:aa, tulevat nämä käyttöjärjestelmässä vakiona. Build-palvelimen konfigurointi tehdään vain kerran, jonka jälkeen se on kyseisillä asetuksilla käytössä kaikissa tulevissa projekteissa. (Microsoft 2012a; 2016a.)

4 TESTIYMPÄRISTÖN KEHITTÄMINEN

Wallacin testiympäristö koostuu Build-, Hyper-V- ja TFS-palvelimesta. Näihin kaikkiin tehdään muokkauksia asetuksiin, jotta testejä voidaan ajaa rinnakkain. Testiympäristön valmistelussa Virtual Machine Manageria varten muutetaan järjestelmäasetuksia järjestelmänvalvojan tasoisilla oikeuksilla taulukon 1. mukaisissa virtuaalikoneissa ja -palvelimissa:

Kone	**Käyttötarkoitus**
Oma työkone	Tähän tietokoneeseen on asennettu Visual Studio, joten koodi ja testikoodi kirjoitetaan myös tältä koneelta. Koneelta suoritetaan myös palvelimien konfiguroinnit etätyöpöytäyhteyden avulla.
Hyper-V-palvelin	Palvelimen käyttöjärjestelmänä toimii Windows Server 2012 R2, ja sen nimeksi on asetettu vm-manager.
Team Foundation Server -palvelin	Palvelimen käyttöjärjestelmänä toimii Windows Server 2012 R2, ja sen nimeksi on asetettu TFS-PALVELIN.
Build-palvelin	Tämä palvelin on tavallinen domainiin liitetty 64-bittinen Windows 10 -virtuaalikone.

Taulukko 1. Testiympäristön valmistelut VMM:ää varten

4.1 SQL Server 2014:n asennus

Avataan etätyöpöytäyhteys vm-manageriin, jolle SQL Serverin asennus suoritetaan. Microsoft SQL Server 2014 Developer 64-bit version asennusohjelman (liite 1) toimintolistauksesta valitaan ”Asennus” ja ”Uusi erillinen SQL palvelimen asennus tai lisää uusia ominaisuuksia olemassaolevaan asennukseen” (New SQL Server stand-alone installation or add new features to an existing installation).

Asetetaan SQL palvelimen rooliksi ”SQL Server ominaisuuksien asennus” (SQL Server feature installation) (liite 2), jotta voidaan asentaa palvelimen päätoimintoja, joita ovat muun muassa tietokantamoottori, analyysipalvelut, raportointipalvelut, sekä integraatiopalvelut. Tämän jälkeen valitaan tarkemmin toiminnot, joita tarvitaan (liite 3). Jotta palvelimen toiminta ei kävisi liian raskaaksi, valitaan vain tarvittavat toiminnot, jotka ovat:

- Database Engine Services

- Client Tools Connectivity
- Client Tools Backwards Compatibility
- Management Tools - Basic
- Management Tools – Complete

Tätä seuraa Instanssin konfigurointi (liite 4). Valitaan oletusinstanssi (Default instance) ja asetetaan tunnisteeksi (instance ID): MSSQLSERVER. Palvelimen konfigurointi (liite 5) Collation-välilehdestä valitaan ”muokkaa” ja asetetaan database enginen tietojenkeräämisasetukseksi ”SQL Collation used for backwards compatibility”, johon asetetaan “SQL_Latin1_General_cp1_CI_AS”

Database enginen konfigurointikohdassa (liite 6) authentikaatiomoodiksi valitaan ”Mixed mode”, ja sitten valitaan ”Lisää nykyinen käyttäjä” (Add Current User), joka asettaa asennusta suorittavalle tunnukselle järjetelmänvalvojan oikeudet sekä sallii kirjautumisen ja pääsyn tietokantaan.

Asennusohjelman näkymä näyttää esikatseluna asennettavat toiminnot ja tiedostosijainnin, johon konfiguraatiotiedosto tallennetaan (liite 7). Valitaan ”Asenna” (Install), jolloin valitut toiminnot asennetaan. Lopuksi ilmestyy yhteenveto, joka ilmoittaa jos tiettyjä toimintoja ei syystä tai toisesta pystytty asentamaan. Kaikki toiminnot asentuivat onnistuneesti (liite 8).

4.2 VMM-palvelimen asennus

Ennen Virtual Machine Managerin asentamista vm-manager -palvelimelle, asennetaan Windows ADK:n. Windows ADK:n asennusohjelmasta asennetaan 2 VMM:n asennuksen kannalta tarpeellista toimintoa, jotka ovat Deployment Tools sekä Windows Preinstallation Environment (Windows PE).

Myös Team Foundation Serverille asennetaan VMM, mutta vain Admin console. Tämä käy nopeammin kuin VMM-palvelimen asentaminen, sillä minkäänlaista konfigurointia eikä ADK:n asennusta tarvitse tehdä. Käynnistetään VMM:n asennusohjelma ja valitaan ”admin console”.

4.3 Virtuaalikoneen ja sen mallin luonti

Tehdään virtuaalikoneen malli VMM:llä. Ensiksi avataan kirjasto-työtila, sen jälkeen koti-välilehdestä (home) valitaan ”Luo VM malli” (Create VM Template), minkä jälkeen erilliseen ikkunaan aukeaa ohjatun luomisen avustaja. Tämän jälkeen valitaan ”luo malli esiasetuksista” ja asetetaan malliksi ”pieni tyhjä asema” (Blank Disk - Small.vhd). Sen jälkeen kirjoitetaan mallille nimeksi Windows7_Template ja valitaan sukupolveksi 1. (Microsoft, 2016e.)

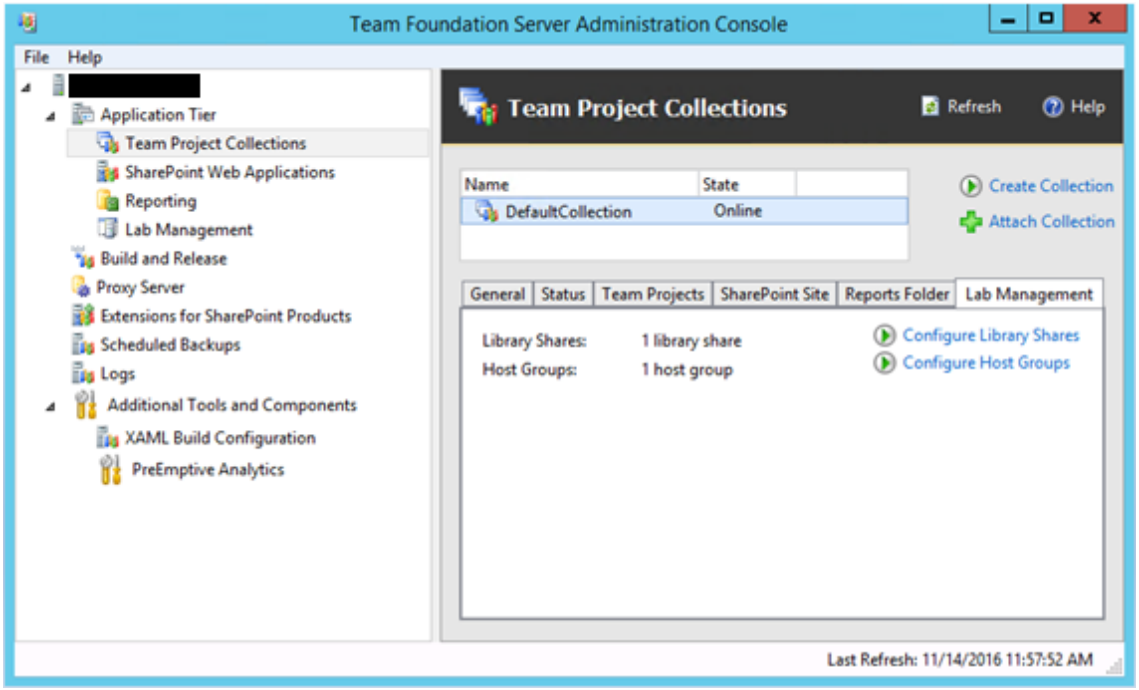
Tämän jälkeen virtuaalikoneen mallille määritellään laitteistoasetukset. Laitteistoasetuksissa määritellään tärkeimpänä prosessorin- ja muistinkäyttö. Asetetaan malli käyttämään muistia 2GB ja yhtä prosessoria.

Seuraavaksi määritetään mallille käyttöjärjestelmäasetukset. Käyttöjärjestelmäasetuksissa tärkeimpänä määritellään käyttöjärjestelmä, sen kirjautumistiedot ja kotiryhmä (home-group).

Mallin pohjalta luodaan VMM:llä virtuaalikone ja asennetaan siihen käyttöjärjestelmäksi Windows 7. Virtuaalikoneen laitteisto- ja käyttöjärjetelmäasetukset tulevat luodusta virtuaalikoneen mallista.

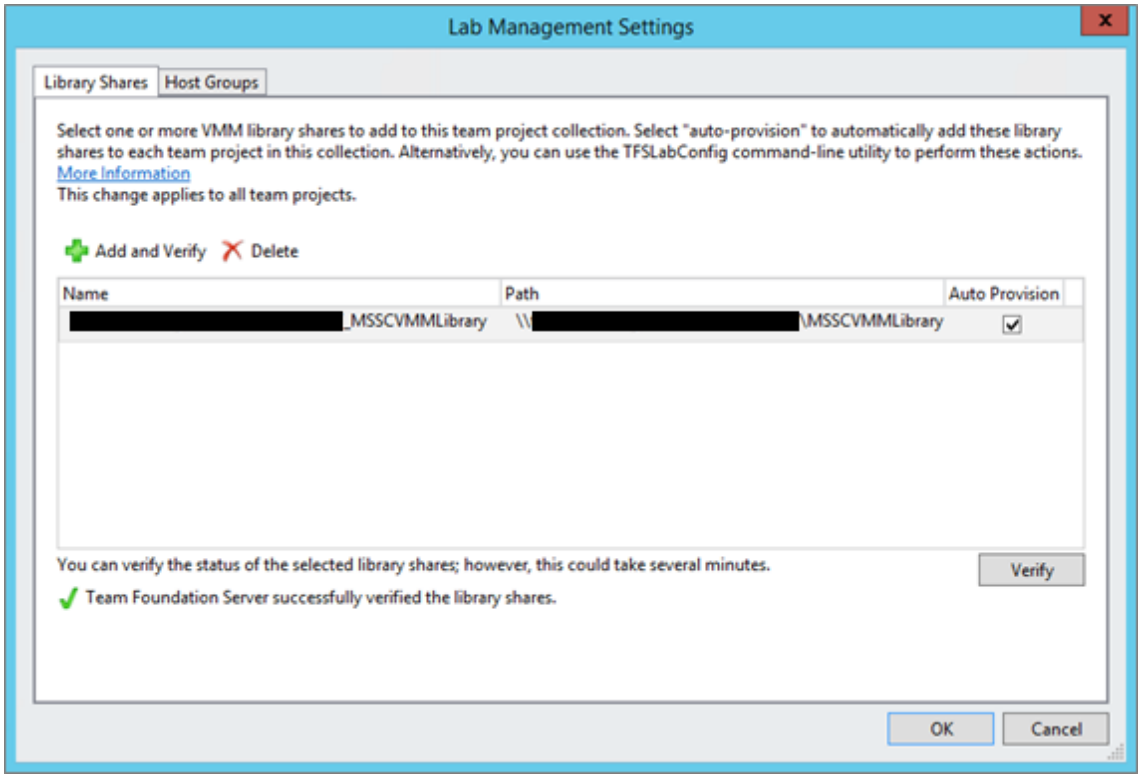
4.4 VMM-palvelimen konfigurointi

Toteutetaan yksinkertaisin testiympäristöasettelu, jossa Hyper-V, Virtual Machine Manager ja VMM:n mukana asennettava Kirjastopalvelin (LibraryMachine) toimivat kaikki yhdellä fyysisellä tietokoneella. Aloitetaan VMM:n asetusten konfigurointi TFS:ssä avaamalla etätyöpöytäyhteys Team Foundation Serverille. Kuvassa 4. näkyvästä Team Foundation Server Administration Consolesta hallitaan TFS:n asetuksia. Avataan valikosta Sovellus-taso (Application tier). Tältä tasolta löytyy Team Project Collection, josta löytyy välilehdeltä Lab managementin asetukset. Lab managementissa on konfiguroidaan ensiksi kirjastopalvelimet (library share) ja sitten isäntäryhmät (host groups). Valitaan ensin ”Configure library shares”.



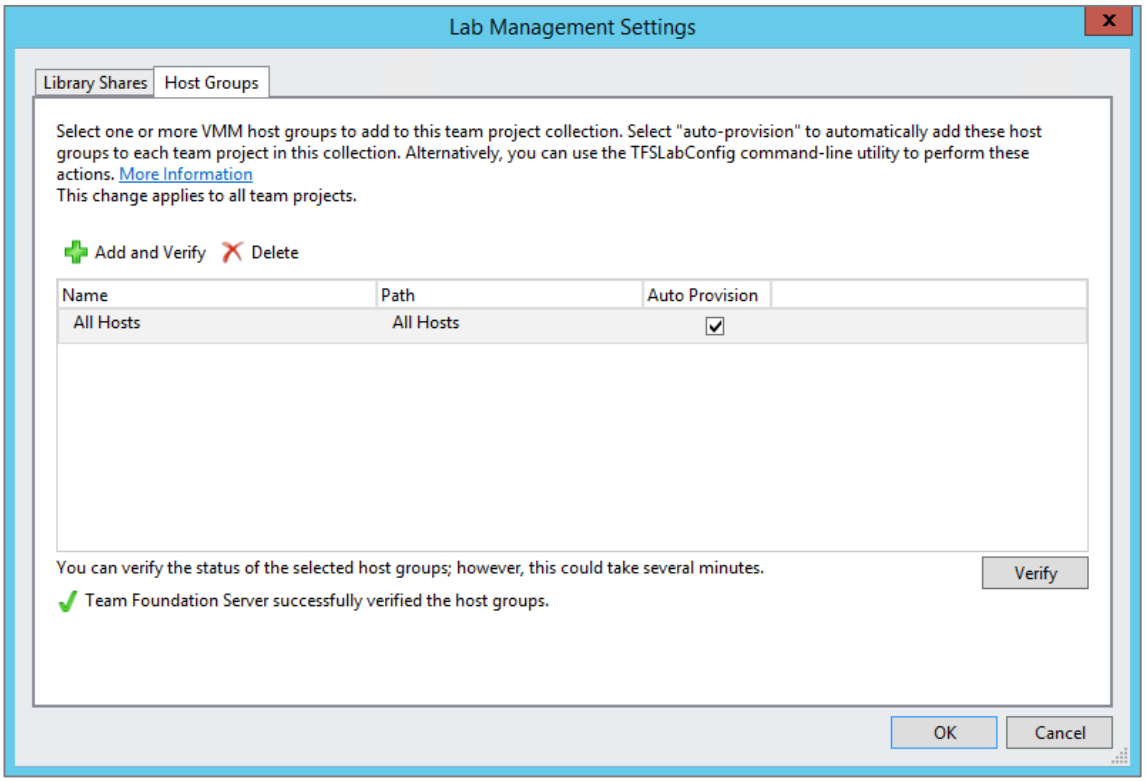
Kuva 4. Team Foundation Server Administration Console TFS-palvelimella.

Lisätään kirjastopalvelin TFS:ään valitsemalla “Lisää ja Verifioi” (Add and Verify) (kuva 5). Aukeaa uusi ikkuna, joka etsii domainista kirjastopalvelimia. Haku löytää vain vm-manageriin luodun MSSCVMMLibraryn. Painetaan “Lisää” (Add), jolloin kyseinen polku lisätään TFS:n tietoihin. Lisäys myös automaattisesti verifioi palvelimen toimivaksi ja palauttaa näkyviin tekstin: ”Team Foundation Server successfully verified the library shares” merkiksi onnistuneesta kirjastopalvelimen lisäyksestä. Tämä asetus kertoo TFS:lle sen, mistä polusta VMM:ssä määritetty kirjastopalvelin löytyy. Tämä tehdään, jotta myöhemmin SCVMM-ympäristöä luodessa TFS ja Test manager löytävät VMM:llä luodut virtuaalikoneet ja virtuaalikonemallit.



Kuva 5. Kirjastopalvelimen konfigurointi TFS:ään.

Avataan samasta konfigurointinäkymästä (kuva 6) toiselta välilehdeltä isäntäryhmät (Host groups). Valitaan ”add and verify”, mikä lisää kaikki isännät kyseiseen project collectioniin. Asetetaan päälle ”Auto Provision”, jolloin kaikki host groupit liitetään jatkossa jokaiseen ryhmäprojektiin tämän collectionin sisällä. Merkiksi onnistuneesta lisäyksestä ilmestyy teksti ”Team Foundation Server successfully verified the host groups”.



Kuva 6. Isäntäryhmien konfigurointi TFS:ään.

4.5 Test managerin konfigurointi

Test managerissa luodaan ensin yhteys TFS-palvelimelle hakemalla se nimellä, minkä jälkeen aukeaa näkymä eri ryhmäprojekteista (kuva 7). Yhdistetään ryhmäprojektiin nimeltä ScvmmTest.

Kuva 7. Test managerin yhdistäminen Team projectiin.

4.5.1 SCVMM-ympäristön luonti Lab centerissä

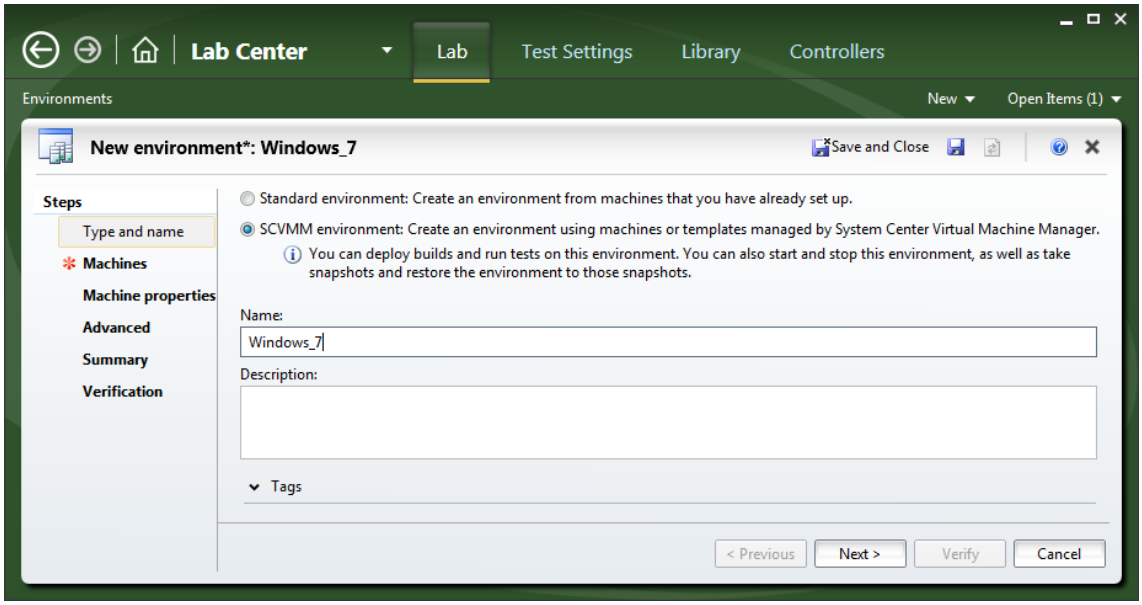
Ennen uuden SCVMM-ympäristön luontia, täytyy ympäristöön lisättäviin koneisiin tehdä esivalmisteluja, jotta ympäristön luonti onnistuu. Otetaan käyttöön tiedostojen ja tulostimien jako sekä Internet Information Services (IIS), josta valitaan käyttöön Windows Authentication.

Ohjauspaneelin ”Network and Internet” -näkymästä mennään kotiryhmä- eli homegroup-asetuksiin ja valitaan ”Advanced sharing settings”. Kytketään ominaisuudet ”Network discovery” sekä ”file and printer sharing” päälle. Nyt kyseinen virtuaalikone löytää homegroupin, johon se voi liittyä.

Siirrytään ohjauspaneelin päänäkymästä Ohjelmat -valikkoon, josta valitaan ”ota käyttöön tai poista käytöstä Windowsin ominaisuuksia”. Otetaan käyttöön IIS, jonka jälkeen navigoidaan IIS:n alavalikkoon ”World Wide Web Services” -> ”Security” ja otetaan käyttöön ”Windows authentication”. Nyt virtuaalikoneiden asetukset ovat kunnossa ja SCVMM-ympäristön luontia voidaan jatkaa.

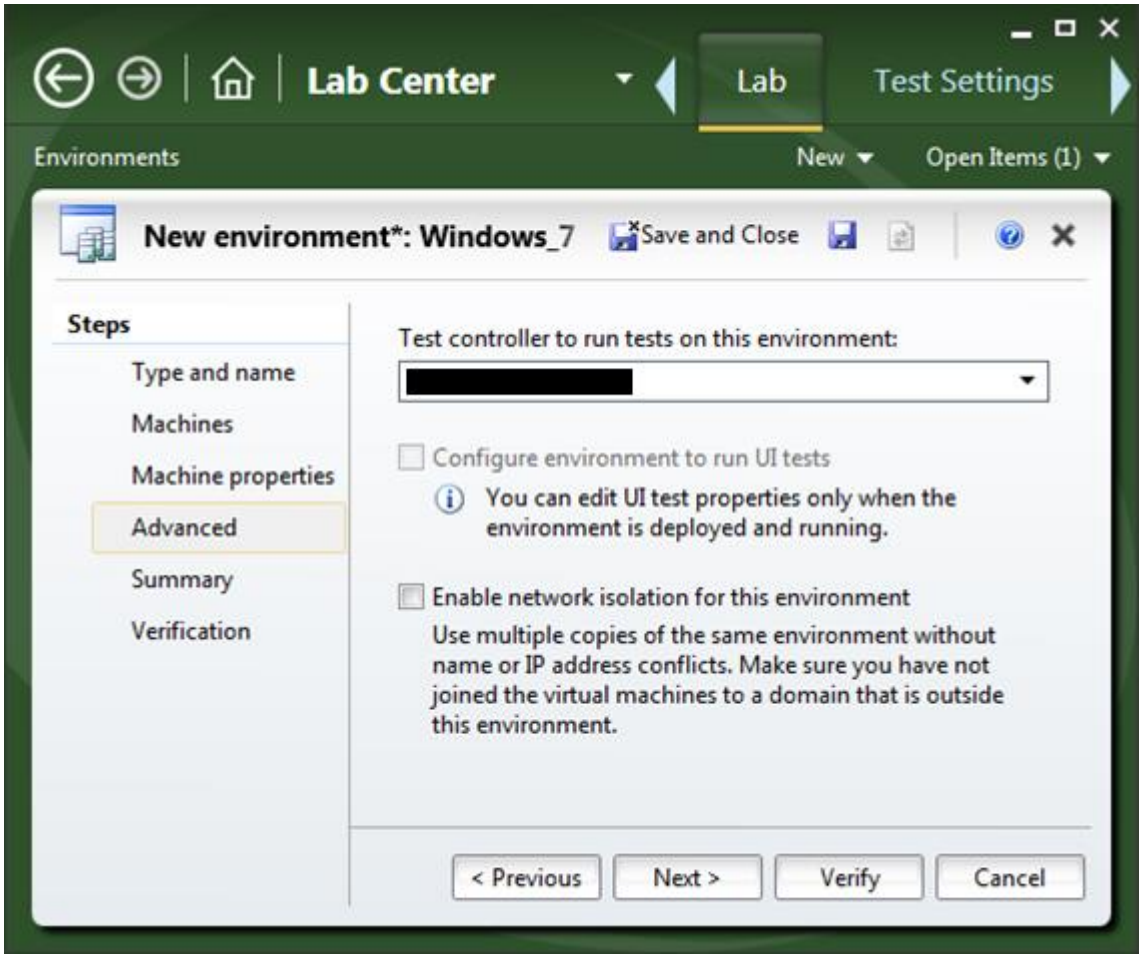
Aloitetaan SCVMM-ympäristön luonti valitsemalla Lab center -näkymästä ”Uusi” (New), jolloin uuteen ikkunaan aukeaa näkymä testiympäristön luomiseen. Lab centerillä voidaan luoda kahdenlaisia testiympäristöjä: standard environment ja SCVMM environment. Tässä työssä keskittytään vain SCVMM-ympäristöihin.

Aloitetaan SCVMM-testiympäristön luonti valitsemalla Test managerin Lab -välilehden näkymästä ”Uusi” (New). Luontinäkymän aloitussivulla määritellään luotavan testiympäristön tyyppi, nimi ja kuvaus (kuva 8). Asetetaan tyypiksi SCVMM environment ja annetaan nimeksi Windows_7.



Kuva 8. Uuden testiympäristön luontinäkymä.

Machines-välilehdellä lisätään ympäristössä toimivat virtuaalikoneet. Koska kyseessä on minimaalinen testi, lisätään ympäristöön vain yksi virtuaalikone. Seuraavaksi Advanced-välilehdellä asetetaan ympäristö käyttämään jo olemassa olevaa test controlleria: SGLV-EKLUNDMA05 (kuva 9).

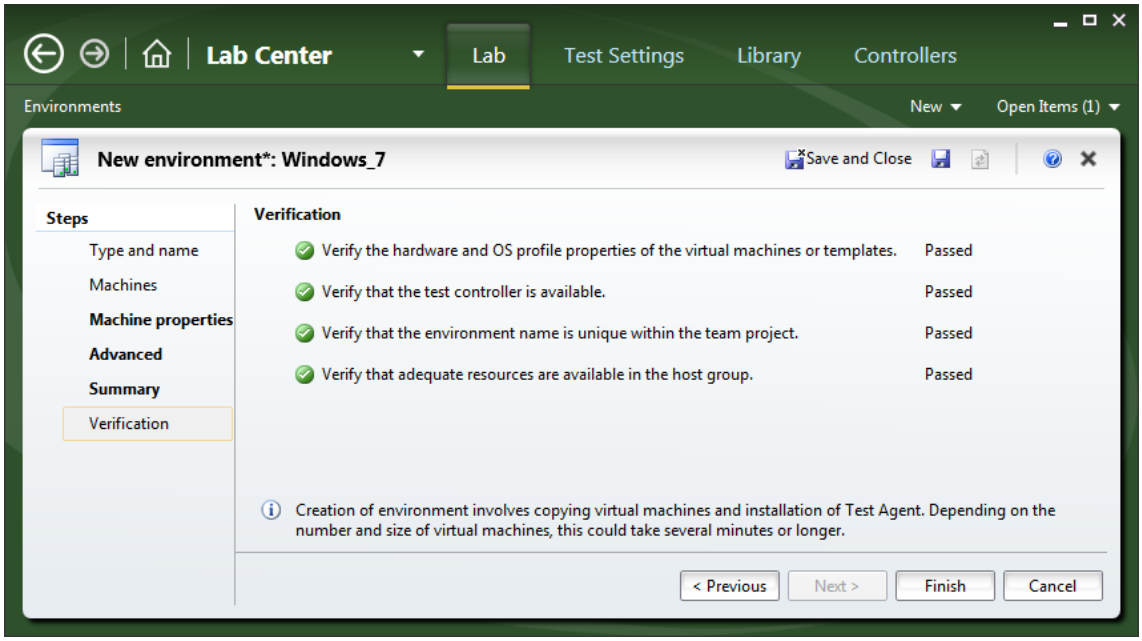


Kuva 9. Test controllerin asettaminen.

Tarkistetaan ja verifioidaan (Verify), että ympäristöön lisättävien virtuaalikoneiden:

- Laitteisto- ja käyttöjärjestelmäprofiilit ovat kunnossa.
- Test controller on online-tilassa.
- Luotavan ympäristön nimi on uniikki tiimiprojektin sisällä
- Riittävät resurssit ovat saatavilla isäntäryhmässä.

Kuvasta 10. nähdään, että ympäristö on luotu onnistuneesti.



Kuva 10. Uuden testiympäristön luonnin verifiointi.

4.5.2 SCVMM-ympäristön valmistelu testaamista varten

SCVMM-ympäristön, eli tässä tapauksessa yhden virtuaalikoneen, luonnin jälkeen tarkistetaan, että sen .NET-ohjelmistokehys on päivitetty versioon 4.5. Tämän jälkeen päivitetään PowerShell versioksi 4.0.

Kolmas edellytys testien ajamiselle on, että virtuaalikone on konfiguroitu sallimaan etäkäskyjen vastaanotto. Tehdään se avaamalla PowerShell Järjestelmänvalvojana, ja sen jälkeen kirjoittamalla ” Enable-PSRemoting”, jonka jälkeen PowerShell kertoo neljä eri suoritettavaa toimintoa;

1. Käynnistetään WinRM. Windows Remote Management (WinRM) -palvelu on osa Microsoftin Web Services Management -protokollaa, joka hallinnoi etäkäskyjä.
2. Asetetaan WinRM-palvelun käynnistymistyyppi automaattiseksi. Tämä on suositeltavaa sallia, jotta etätyöyhteydet toimivat aina kun virtuaalikone on päällä.
3. Luodaan kuuntelija (listener) hyväksymään tulevia pyyntöjä mistä tahansa IP-osoitteesta.

4. Sallitaan poikkeus Windowsin palomuurin poikkeuksiin saapuvalle WS-Management liikenteelle (vain http-liikenteelle).

Hyväksytään kaikki asetukset. PowerShell ilmoittaa, että WinRm-palvelu on jo käynnissä ja sallii etäyhteydet. Tämän jälkeen PowerShell vielä varmistaa sallitaanko konfiguroinnit. Näin on taattu, ettei testejä ajettaessa tule ongelmia etäyhteyksissä WinRM-palvelun ja palomuurin estojen takia. (Microsoft, 2016d.)

Vasta sitten, kun yhdellä ympäristöllä on saatu onnistunut testiajo, luodaan toinen testiympäristö. Ensimmäisessä testissä käytetystä virtuaalikoneesta valitaan VMM:lla ”clone” ja toistetaan samat askeleet kuin ensimmäisen ympäristön luonnin kanssa. Virtuaalikone nimetään uusiksi, jottei workgroupissa tule epäselvyyksiä edellisen virtuaalikoneen kanssa.

4.6 Build-palvelimen konfigurointi

Visual Studion Team Explorer -valikosta löytyy linkki TFS:n Web-portaaliin projektin hallintanäkymään (http://TFS-PALVELIN:8080/tfs/DefaultCollection/ScvmmTest/_dashboards), jossa ”TFS-PALVELIN” on TFS-palvelimen nimi, ”DefaultCollection” on oletuskokoelma kaikille ryhmäprojekteille, ja ”ScvmmTest” on ryhmäprojekti. Ladataan portaalista testi-agentti, joka asennetaan ja konfiguroidaan build-palvelimelle.

Agentti latautuu pakattuna .zip-tiedostona, joten puretaan se lähelle C:\-aseman juurta, jolloin siihen navigoiminen komentorivi-työkalulla on hieman helpompaa. Purettu paketti sisältää Agent-nimisen alikansion sekä RunAgent.cmd- ja ConfigureAgent.cmd -nimisen konfigurointitiedoston.

Käynnistetään komentorivityökalu järjestelmänvalvojana ja navigoidaan kansioon C:\Temp\Agent. Tämän jälkeen käynnistetään Agentin asennus ja konfigurointi. Tätä seuraa sarja kysymyksiä, joista jokaiseen tulee samalla myös suluissa oleva oletusvastaus (taulukko 2). Jos jättää vastauksen tyhjäksi ja painaa ”Enter”, konfiguroidaan build-palvelimelle oletusvastaus.

Asetus	Oletusvastaus	Toiminto
Aseta agentin nimi	Tietokoneen nimi	Tämä on build-palvelimella toimivan agentin nimi. Koska näitä saattaa olla useampi kuin yksi, kannattaa agentti nimetä mahdollisimman kuvaavasti. Testit nimettiin Visual Studiossa ”Dummy”-etuliitteellä, joten tässä asetettiin agentin nimeksi ”DummyAgent”.
Aseta TFS- palvelimen URL	-	Tässä oletusvastaus on tyhjä. Tässä haetaan TFS-palvelimen osoitetta, josta koottavat projektit löytyy, joka tässä tapauksessa on: http://PALVELIMEN_NIMI:8080/tfs.
Mitä Agent poolia vasten tämä agentti konfiguroidaan?	Agent pool on ”default”	Agent pooleja voi olla projektin sisällä useampia, mutta tässä tapauksessa sillä ei ole juurikaan merkitystä, koska testien ja ympäristön koko tulee olemaan hyvin minimaalinen, joten asetetaan Agent pooliksi ”default”.
Aseta agentille tiedostopolku työskentelykansioon	Oletus on C:\Temp\Agent_work	Oletuksena on aiemmin puretun agent.zipin alikansio, joka on tässä tapauksessa sopiva.
Haluatko asentaa agentin Windows servicenä Kyllä/Ei	Ei	Koska ei aiota tehdä koodattuja käyttöliittymätestejä tai interaktiivisia lataustestejä, valitaan "kyllä".
Anna käyttäjänimi, joka käyttää tätä palvelua	NT AUTHORITY\NETWORK SERVICE	Jos testit ajetaan domain-ympäristössä, kuten tässä tapauksessa, tulee tähän syöttää Domainin nimi sekä Build-palvelimen käyttäjätunnus eli: DOMAIN\KÄYTTÄJÄNIMI
Anna käyttäjän DOMAIN\ KÄYTTÄJÄNIMEN	-	Syötetään KÄYTTÄJÄNIMEN-käyttäjän salasana

Taulukko 2. Build-palvelimen konfigurointi.

Onnistuneen konfiguroinnin merkiksi komentorivi palauttaa seuraavan tekstin:

> Installing service vsoagent.TFS-PALVELIN.DummyAgent…
> Service vsoagent.TFS-PALVELIN.DummyAgent has been successfully installed.
> Creating EventLog source vsoagent.TFS-PALVELIN.DummyAgent in log Application…

Nyt Build-agentti on asennettu ja konfiguroitu onnistuneesti ja valmis käyttöä varten (Microsoft 2016c; 2016d). Seuraavaksi päästään luomaan testi ja suorittamaan testiajo.

5 TESTIN LUONTI JA TESTIAJO

Aloitetaan testin luonti tekemällä uusi ryhmäprojekti valitsemalla Visual Studiossa ”Tiedosto -> Uus -> Ryhmäprojekti” (File -> New -> Team Project). Asetetaan projektille nimeksi ScvmmTest ja jätetään kuvaus tyhjäksi. Tämän jälkeen TFS luo projektille hallintanäkymän osoitteeseen http://TFS-PALVELIN:8080/tfs/defaultcollection/ScvmmTest.

Koodi 1. on Visual Studiossa luotu lyhyt testikoodi, joka on kirjoitettu C#-ohjelmointikielellä. Koodin 1. toimivuutta testaa koodi 2, joka on yksikkötesti, ja se on kirjoitettu myös C#-ohjelmointikielellä. Nämä kaksi koodia kuvastavat normaalia kehitys- ja testausympäristöä, mutta paljon pienemmässä mittakaavassa. Tällä voidaan testata tämän työn testiympäristön toimivuutta vastaten oikeita olosuhteita.

DummyClass.cs | UnitTest1.cs

DummyLib | DummyLib.DummyClass

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace DummyLib
8   {
        1 reference | Kuutti, Teemu, 17 days ago | 1 author, 1 change
9       public class DummyClass
10      {
            1 reference | Kuutti, Teemu, 17 days ago | 1 author, 1 change
11          public int Add(int a, int b) {
12              return a + b;
13          }
14      }
15  }
```

Koodi 1. Testikoodi.

UnitTest1.cs | DummyClass.cs

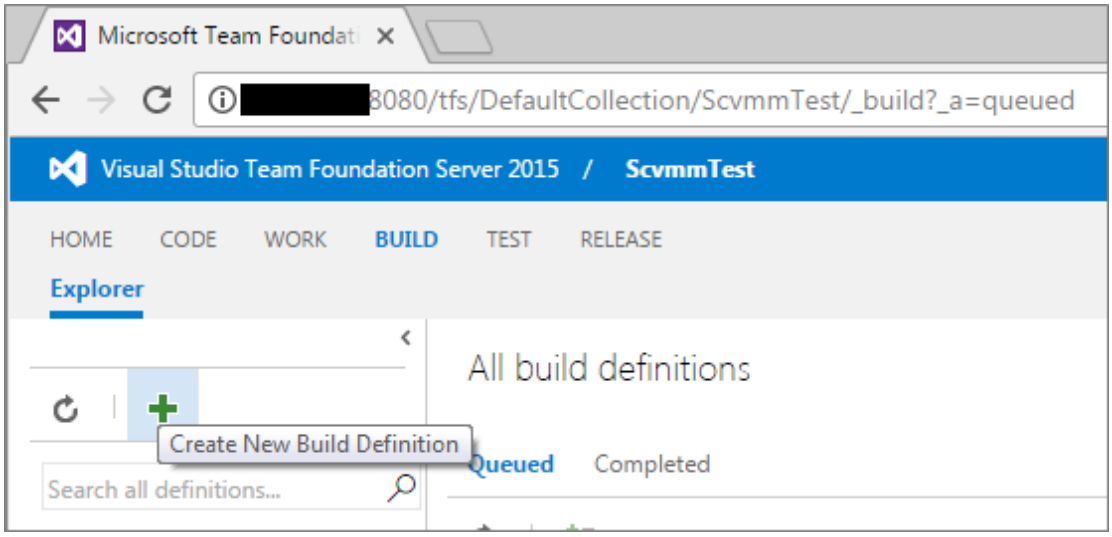
DummyLibTests | DummyLibTests.UnitTest1

```
1   using System;
2   using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4   namespace DummyLibTests
5   {
6       [TestClass]
        0 references | Kuutti, Teemu, 17 days ago | 1 author, 1 change
7       public class UnitTest1
8       {
9           [TestMethod]
            0 references | Kuutti, Teemu, 17 days ago | 1 author, 1 change
10          public void AddTest()
11          {
12              var d = new DummyLib.DummyClass();
13              Assert.AreEqual(3, d.Add(1, 2));
14          }
15      }
16  }
```

Koodi 2. Yksikkötesti testikoodille.

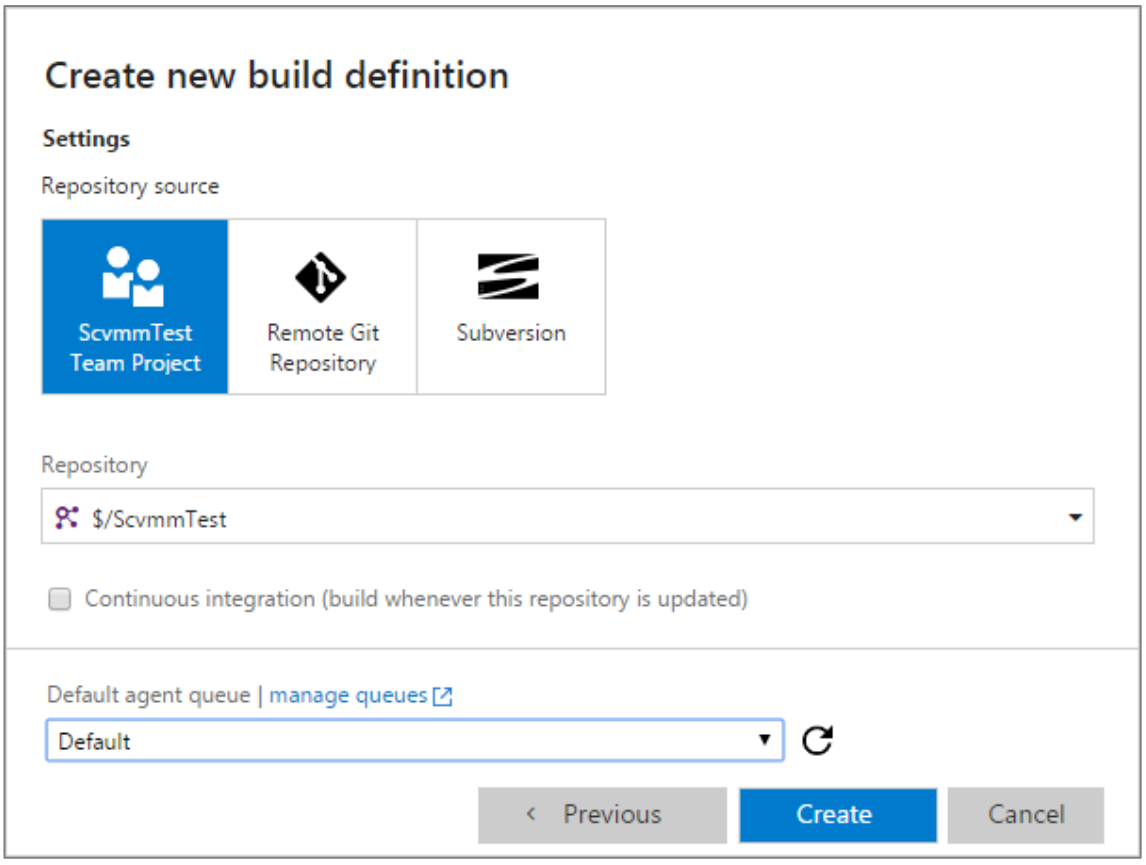
5.1 Testiajon asetukset

Web-portaalissa hallitaan testejä ja testiajoja. Ennen testien ajamista projektille luodaan build-määritelmä, jonka mukaan ohjelmistojen osat kootaan ja testataan. Aloitetaan build-määritelmän luonti web-portaalin Explorer-näkymästä ”Luo uusi build-määritelmä” (Create New Build Definition) (kuva 11).



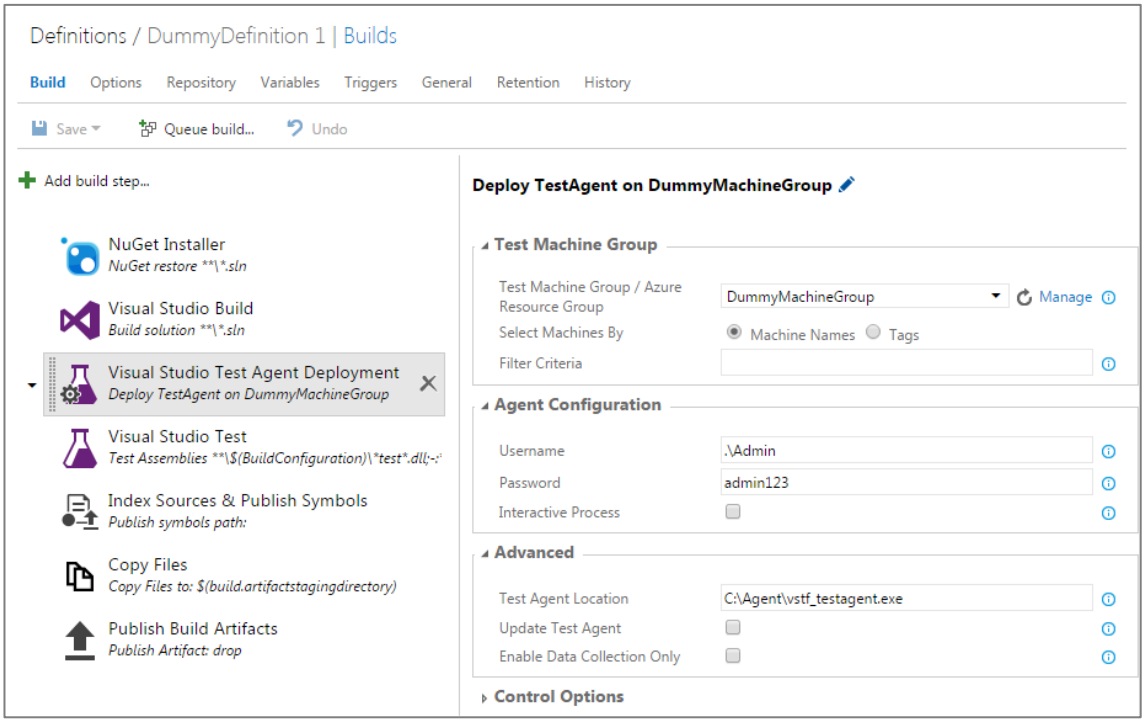
Kuva 11. Uuden build-määritelmän luonti.

Uutta build-määritelmää luodessa määritetään sen sijainniksi (repository) aiemmin luotu ScvmmTest-ryhmäprojekti (kuva 12). Tässä yhteydessä jätetään asettamatta päälle jatkuvan integraation asetus, joka suorittaa projektin buildin ja testauksen jokainen kerta, kun projektin sijaintiin tulee muutoksia tai päivityksiä.



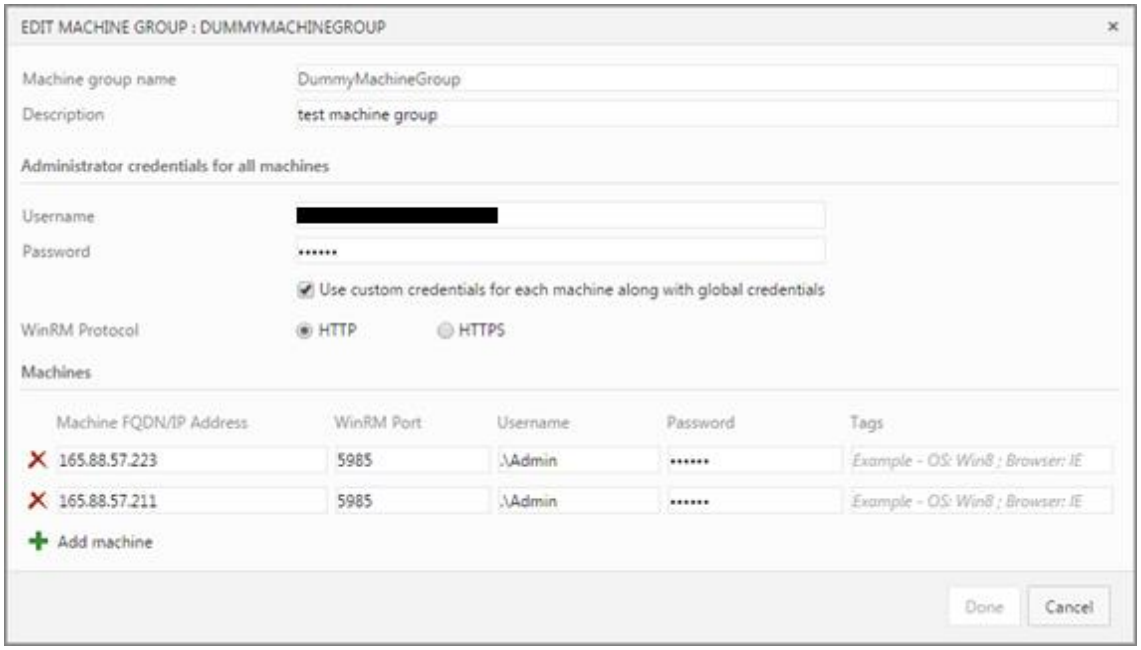
Kuva 12. Uuden build-määritelmän sijainti.

Web-portaalin build-määritelmänäkymässä on valmis perusrunko sille, miten projektin eri vaiheet eli buildaus (koonti) ja testaus suoritetaan. Perusrunko sisältää 6 eri vaihetta. Lisätään yksi uusi vaihe painamalla vihreää plus-ikonia, jossa lukee ”Add build step”. Lisätään vaihe ”Visual Studio testi-agentin käyttöönotto” ja asetetaan se vaiheiden ”Build” ja ”Test” väliin (kuva 13). Kuvasta 13. nähdään lisätyn vaiheen eri asetukset, jotka konfiguroidaan testattavalle projektille sopivaksi. Luodaan uusi testikoneiden ryhmä (Test Machine Group) asettamalla sille nimeksi DummyMachineGroup.



Kuva 13. Testi-agentin käyttöönotto.

Painamalla Kuvan 13. ”Hallitse”-näppäintä (Manage) päästään asettamaan testikoneiden ryhmän asetukset (kuva 14). Syötetään käyttäjänimi ja salasana, jotka pystyvät kirjautumaan kaikkiin testikoneryhmän koneisiin. Määritetään etäyhteysprotokollaksi HTTP, ja sen alle testikoneryhmässä olevien koneiden IP-osoitteet. Testit ajautuvat testikoneisiin näiden IP-osoitteiden perusteella. Testikoneisiin asetetaan myös paikallisen järjestelmävalvojan käyttäjänimi ja salasana. WinRM-porttinumero määrittyy testikoneille automaattisesti sen mukaan onko valittu HTTP vai HTTPS. HTTP käyttää oletuksena porttia numero 5985 ja HTTPS porttia numero 5986, mutta tarvittaessa käyttäjä voi määrittää ne itse testiympäristön mukaan.



Kuva 14. Testikoneryhmän asetukset.

5.2 Testiajo

Asetusten määrittämisen jälkeen voidaan suorittaa testiajo. Liitteestä 11 nähdään testiajon suorittamisen käynnistysvaiheet. Testivaiheet menevät build-määritelmän mukaisessa järjestyksessä. Ensiksi asennetaan projektin vaatimat NuGetit, ja sen jälkeen suoritetaan ohjelmiston osien koonti.

Liitteessä 12. nähdään web-portaalin näkymä testiajon vaiheista siirryttäessä itse testi-agenttien käyttöönottoon. Liitteestä 13. nähdään hieman selvemmin tämän tapahtuman eri vaiheet. Ensiksi build-kone ottaa yhteyttä testikoneisiin, ja sen jälkeen tarkistaa, onko niissä testi-agentti jo valmiiksi asennettuna. Huomataan että ei ole, joten testiajo lähtee itse konfiguroimaan testi-agenttia testikoneisiin. Tässä kestää jonkin aikaa riippuen verkkoyhteyden nopeudesta. Tässä työssä testi-agenttien siirtämiseen testikoneille ja niiden konfiguroimiseen kului noin 30 sekuntia. Kun testi-agentit on konfiguroitu ja testikoneet ovat valmiita testausta varten, tulee näytölle teksti ”Testagents are configured and ready to run tests, make sure you use the 'Visual Studio Test using Test Agent' task in the Build Definition after this task”.

Testiajon suoritus valmistui melkein heti sen jälkeen, kun testiagentit oltiin saatu konfiguroitua. Tämä johtui siitä, että luodut testitapaukset olivat niin pieniä, ettei niiden aja-

misen mennyt kauaa. Yhteensä testiajon kesto oli noin yksi minuutti. Liitteestä 14. nähdään testiajon suorittamisen viimeistelyvaiheet, jossa testiajo ilmoittaa tulokset sekä eri tiedostosijainnit, joihin se tallentaa testiajon lokin ja tulokset. Lopuksi web-portaalin näkymään teksti ”Build Succeeded”, eli koonti onnistui ja testit menivät läpi testikoneissa onnistuneesti.

6 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli kehittää ja tehostaa PerkinElmer-konserniin kuuluvan Wallac Oy:n sovelluskehityksen yksikkötestausprosessin automaatiota selvittämällä onko mahdollista tehdä yksikkötestien rinnakkaisajoa.

Työn tavoitteet saavutettiin ja rinnakkaistestiajo saatiin suoritettua onnistuneesti. Työn valmis toteutus osoitti, että testausta voidaan tehostaa ja testausaikaa lyhentää ottamalla käyttöön useampi testikone, jossa testit ajetaan. Työ osoitti myös, että testejä on mahdollista ajaa yhtä aikaa eri ympäristöissä. Näin ollen voidaan testata ohjelmiston toimivuutta esimerkiksi Windows- ja Linux-käyttöjärjestelmissä samanaikaisesti.

Työn tavoitteet ja vaatimukset eivät työn aikana muuttuneet, mutta tarkentuivat aina hieman työn edetessä. Työ osoitti, että yksikkötestejä voidaan ajaa monessa eri virtuaalikoneessa yhtä aikaa, lyhentäen näin testaukseen menevää aikaa. Työn tuloksia voidaan Virtual Machine Managerin osalta soveltaa missä tahansa testiympäristössä, jossa on käytössä Microsoftin palvelinympäristö, se täytyy vain ottaa käyttöön oman testiympäristönsä asetusten mukaisesti.

Opinnäytetyön valmis toteutus sisältää kaksi eri SCVMM-testiympäristöä, joihin molempiin luotiin yksi virtuaalikone. Testaus tehtiin rinnakkaisajona ja se suoritettiin onnistuneesti. Testi oli hyvin minimaalinen ja tehtiin tätä opinnäytetyötä varten, eikä se liity mitenkään Wallacin valmistamiin ohjelmistoihin. Se todisti vain teoriatasolla, että testiajot voidaan suorittaa monessa virtuaaliympäristössä ja -koneessa yhtä aikaa. Koska Wallacin valmistamat ohjelmistot ovat niin suuria, SCVMM -ympäristön integroiminen osaksi automatisoitua testiajoa vaatii tarkempaa suunnittelua ja valmistelua. Myös fyysisen palvelimen järjestelmävaatimukset tulee ottaa huomioon otettaessa käyttöön suurempia testiympäristökokonaisuuksia.

Tämän työn pohjalta on tarkoitus luoda virtuaaliset testausympäristöt Wallacin ohjelmistoille, jotta saadaan oikea käytännön hyöty yksikkötestaukseen sekä ohjelmistojen toimivuuden ylläpitoon. Tämä tapahtuu lisäämällä virtuaalikoneiden määrää SCVMM-ympäristössä, sekä luomalla build-määritelmät jokaiselle ohjelmistolle ja testiympäristölle sopivaksi.

LÄHTEET

ISTQB exam certification 2016. Why is software testing necessary? Viitattu 14.11.2016 http://istqbexamcertification.com/why-is-testing-necessary/

Marick, B. 2013. When Should a Test Be Automated? Viitattu 25.11.2016 http://www.exampler.com/testing-com/writings/automate.pdf

Microsoft 2009. Creating Virtual Machines from a Template. Viitattu 10.11.2016 https://technet.microsoft.com/en-us/library/cc764306.aspx?f=255&MSPPError=-2147217396

Microsoft 2012a. Deploy and Configure Build Agents. Viitattu 11.11.2016 https://msdn.microsoft.com/en-us/library/bb399135(v=vs.110).aspx

Microsoft 2012b. Running Tests in Microsoft Test Manager. Viitattu 15.11.2016 bhttps://msdn.microsoft.com/en-us/library/dd286680(v=vs.110).aspx

Microsoft 2016a. Building and Debugging (Visual C#). Viitattu 25.11.2016 https://msdn.microsoft.com/en-us/library/ms173083(v=vs.90).aspx

Microsoft 2016b. Configure Lab Management for SCVMM environments. Viitattu 12.10.2016 https://msdn.microsoft.com/en-us/library/dd380687.aspx?f=255&MSPPError=-2147217396

Microsoft 2016c. Deploy an agent on Windows. Viitattu 11.11.2016 https://www.visualstudio.com/en-gb/docs/build/admin/agents/v2-windows

Microsoft 2016d. Deploy ASP.NET apps to workgroup-joined machines using WinRM. Viitattu, 10.11.2016, https://www.visualstudio.com/en-us/docs/release/examples/other-servers/net-to-workgroup-vm

Microsoft 2016e. How to Create a Virtual Machine Template. Viitattu 9.11.2016 https://technet.microsoft.com/en-us/library/hh427282%28v=sc.12%29.aspx?f=255&MSPPError=-2147217396

Microsoft 2016f. System Requirements: VMM Management Server in System Center 2012 and in System Center 2012 SP1 Viitattu 10.11.2016 https://technet.microsoft.com/en-us/library/gg610562(v=sc.12).aspx

Microsoft 2016g Team Foundation Server. Viitattu 10.11.2016 https://msdn.microsoft.com/en-us/library/ms181238(v=vs.90).aspx

Microsoft 2016h. Test: Visual Studio Test Agent Deployment. Viitattu 7.11.2016 https://www.visualstudio.com/en-us/docs/build/steps/test/visual-studio-test-agent-deployment

Microsoft 2016i. Understanding Generation 1 and Generation 2 Virtual Machines in VMM. Viitattu, 9.11.2016 https://technet.microsoft.com/en-us/library/dn440675%28v=sc.12%29.aspx?f=255&MSPPError=-2147217396#Host

Microsoft 2016j. Windows Assessment and Deployment Kit (Windows ADK) for Windows 8.1 Update. Viitattu 16.11.2016 https://www.microsoft.com/en-US/download/details.aspx?id=39982

Richard Fennel 2012. TFS Test agent cannot connect to test controller – Part 2. Viitattu 8.11.2016 http://blogs.msmvps.com/richardfennell/2012/10/01/tfs-test-agent-cannot-connect-to-test-controller-part-2/

Selenium 2016. Brief History of The Selenium Project. Viitattu 10.11.2016 http://www.seleniumhq.org/docs/01_introducing_selenium.jsp#test-automation-for-web-applications

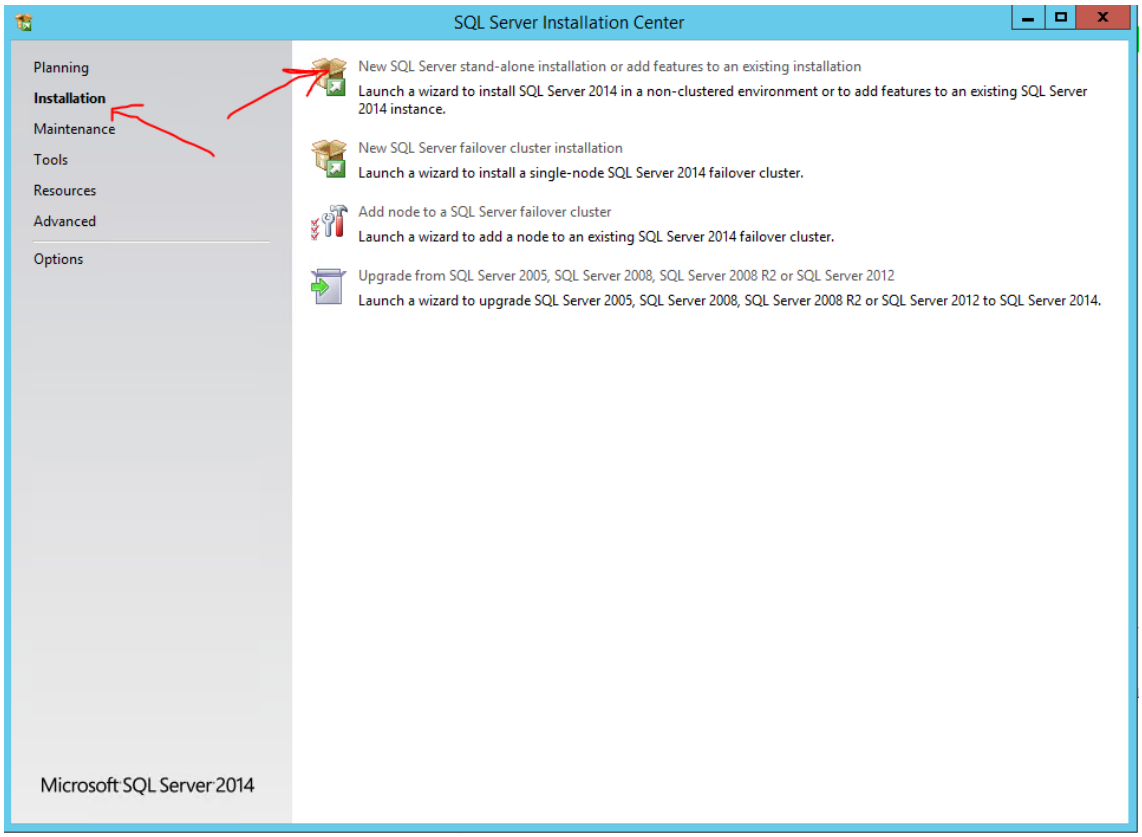
Software Testing Fundamentals 2016a. Software quality dimensions. Viitattu 11.12.2016 http://softwaretestingfundamentals.com/dimensions-of-software-quality/

Software Testing Fundamentals 2016b. Unit Testing. Viitattu 7.11.2016 http://softwaretestingfundamentals.com/unit-testing/

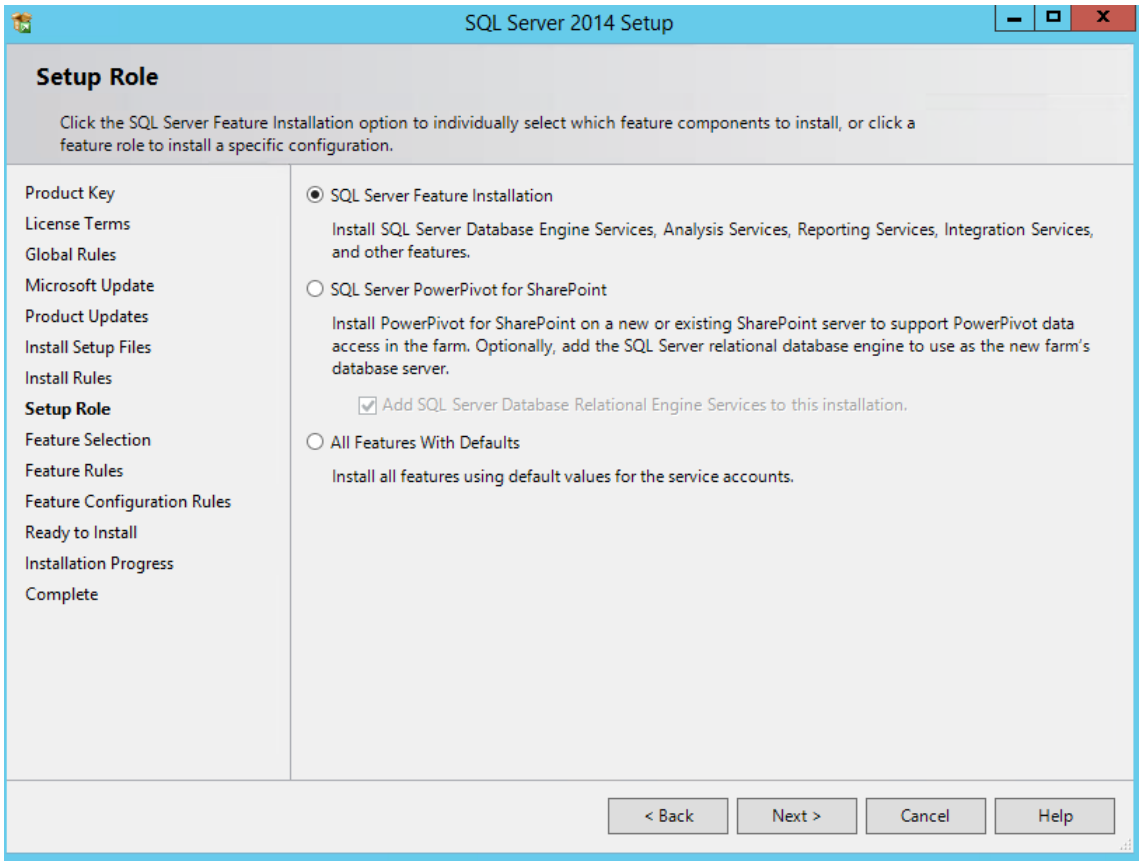
Tutorialspoint 2016. What is Test Environment? Viitattu 11.12.2016 https://www.tutorialspoint.com/software_testing_dictionary/test_environment.htm

The Chromium Projects 2016. Chrome Release Channels. Viitattu 25.11.2016 http://www.chromium.org/getting-involved/dev-channel

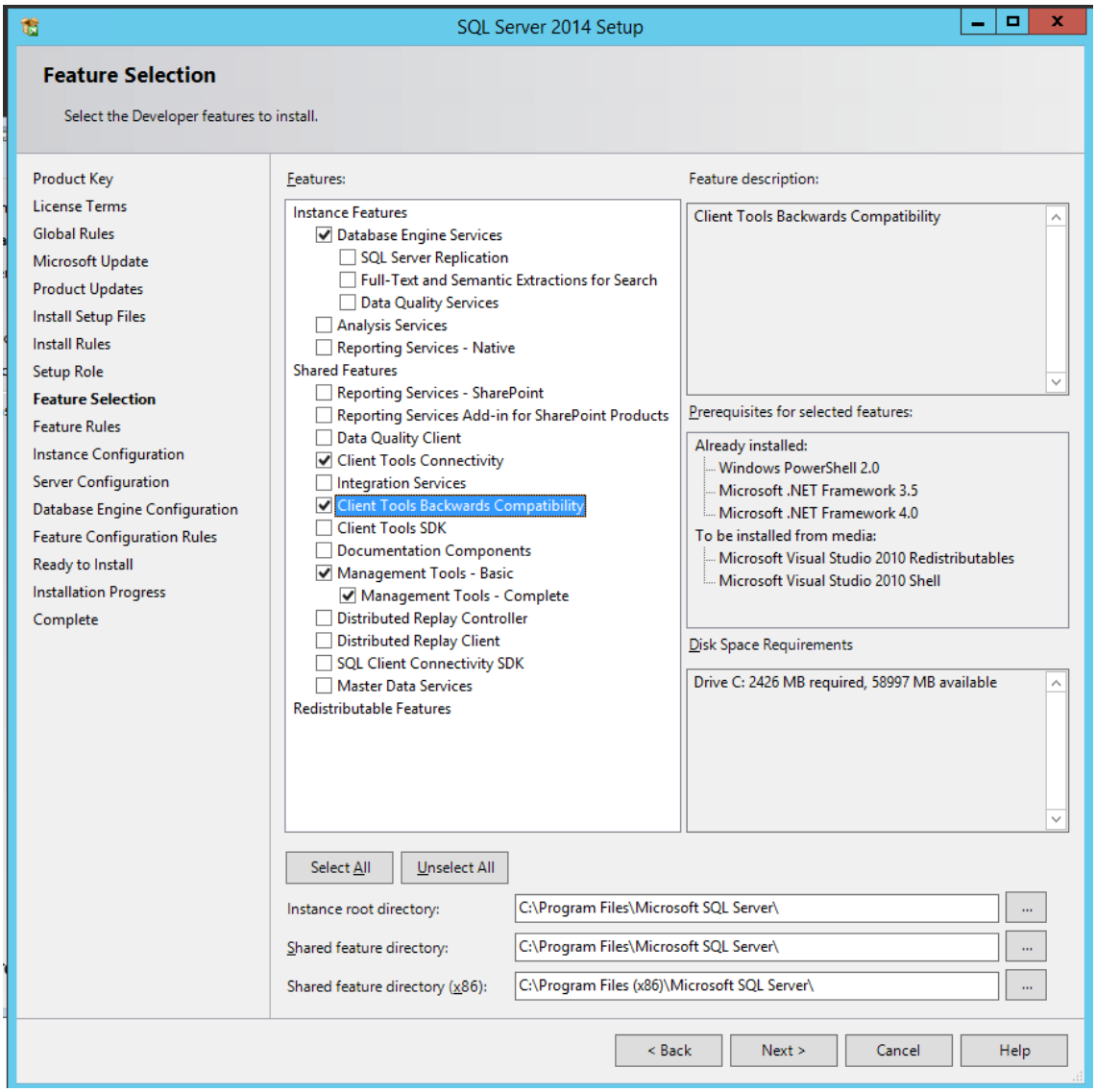
SQL Server 2014 -asennusohjelma



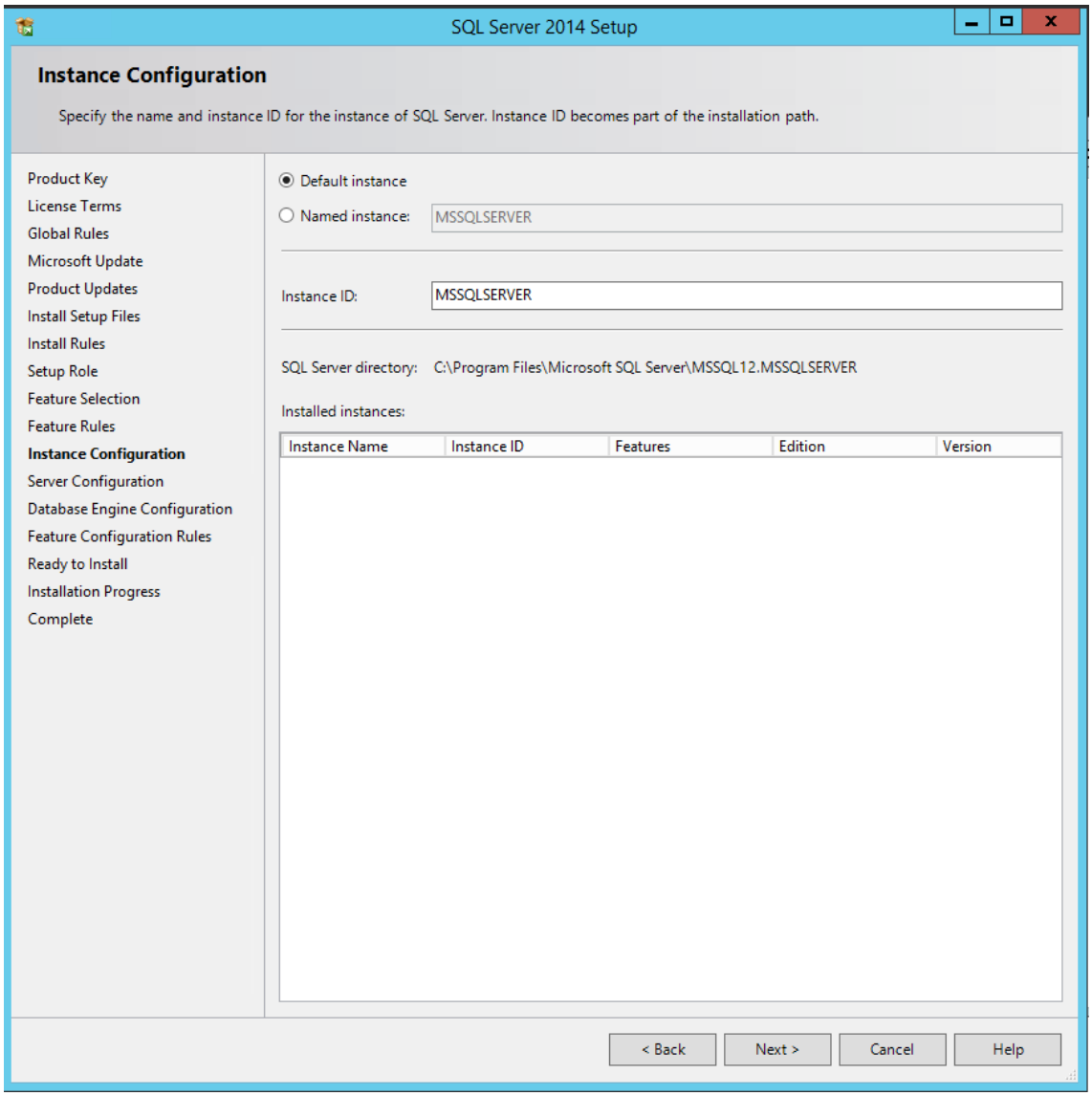
SQL Palvelimen roolin asetus



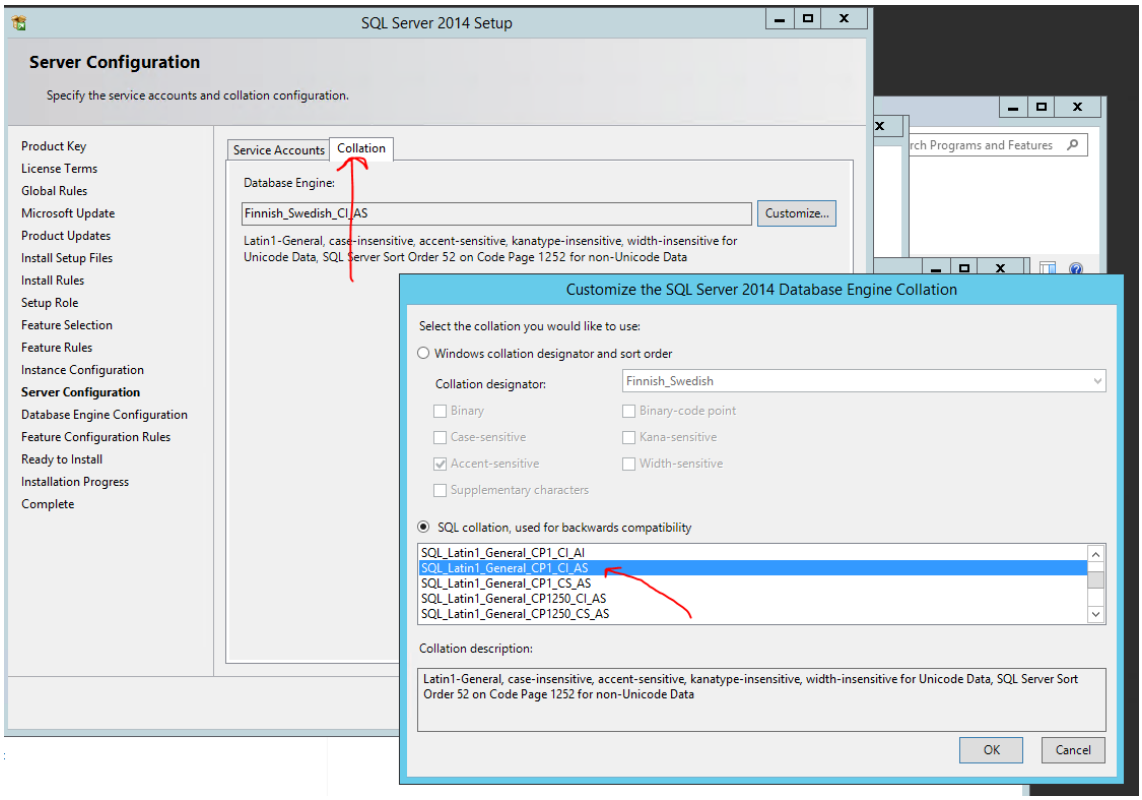
SQL Palvelimen ominaisuuksien asettaminen



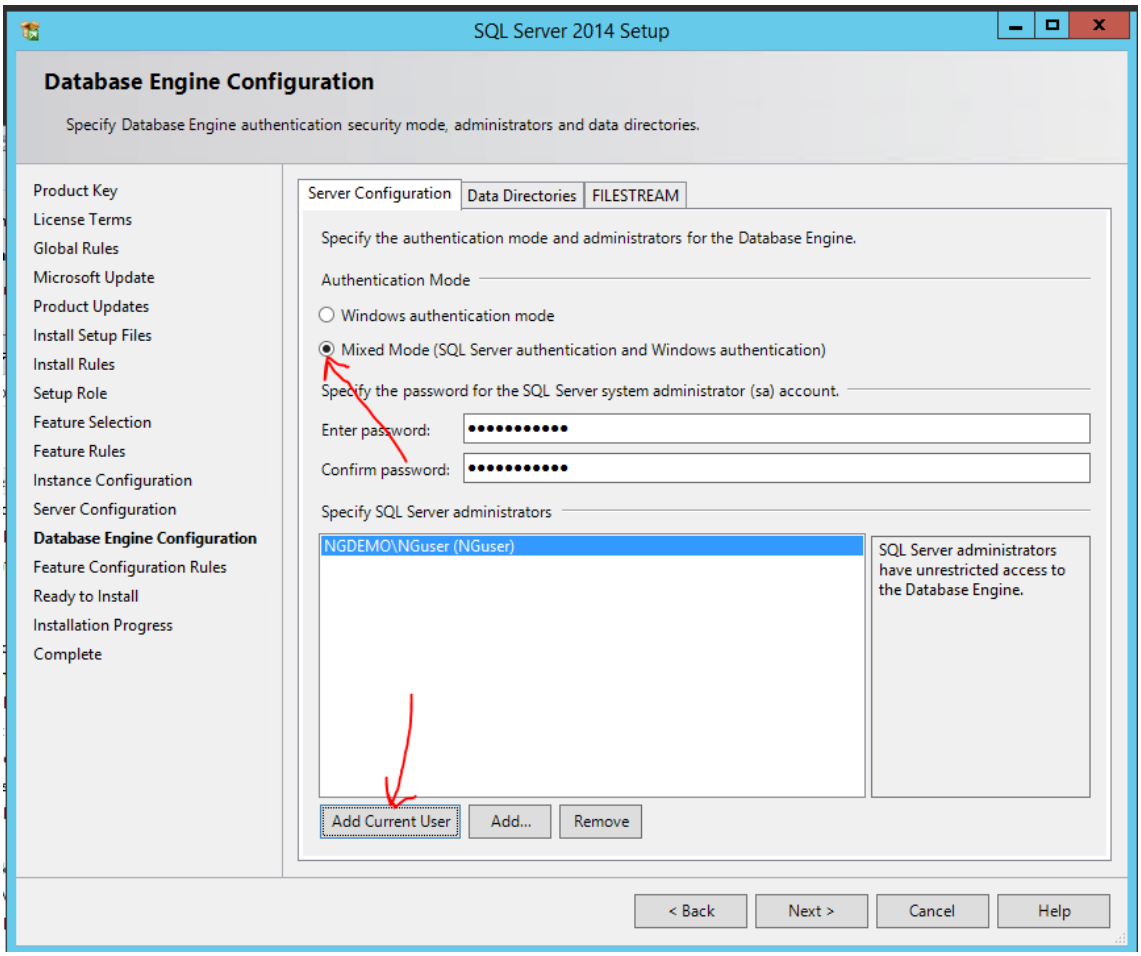
SQL Palvelimen instanssin konfigurointi



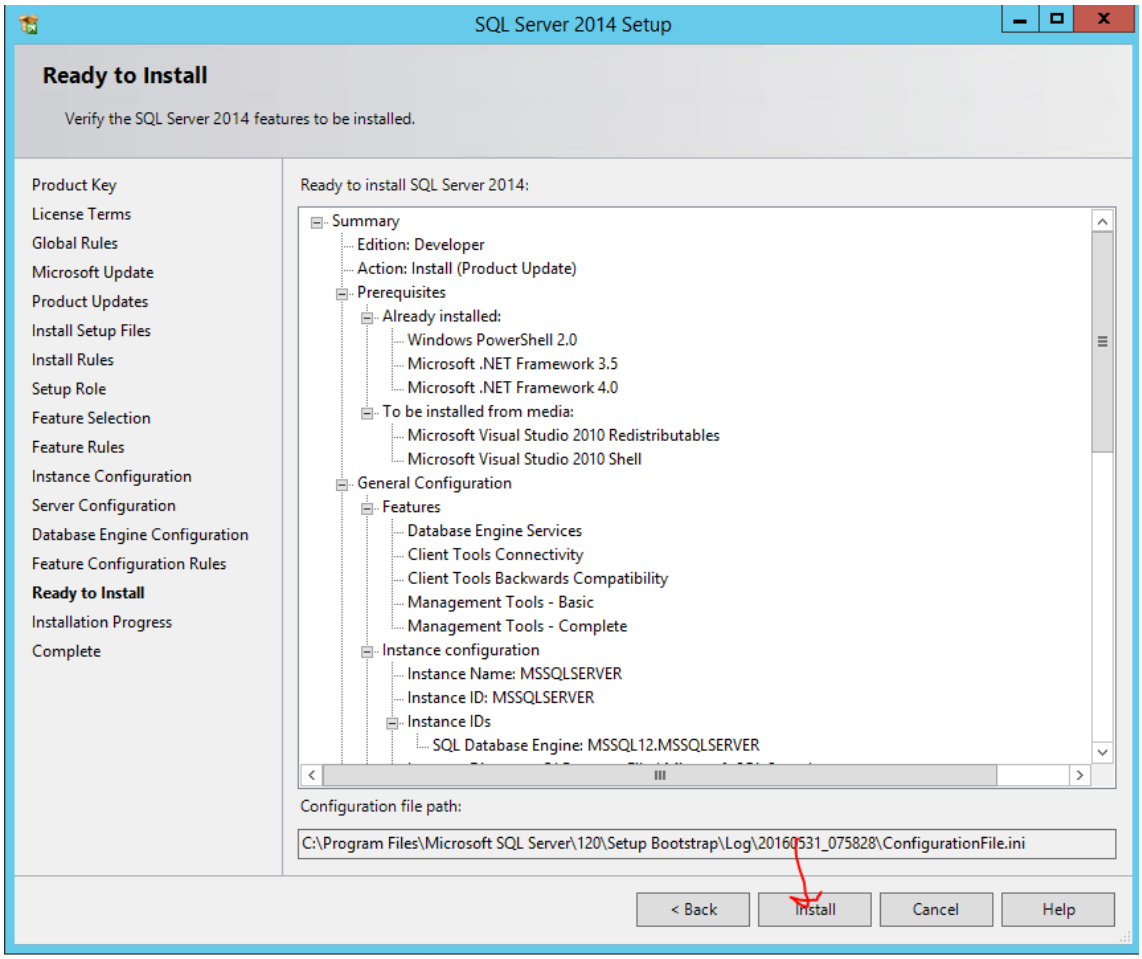
SQL Palvelimen kollaatioasetus



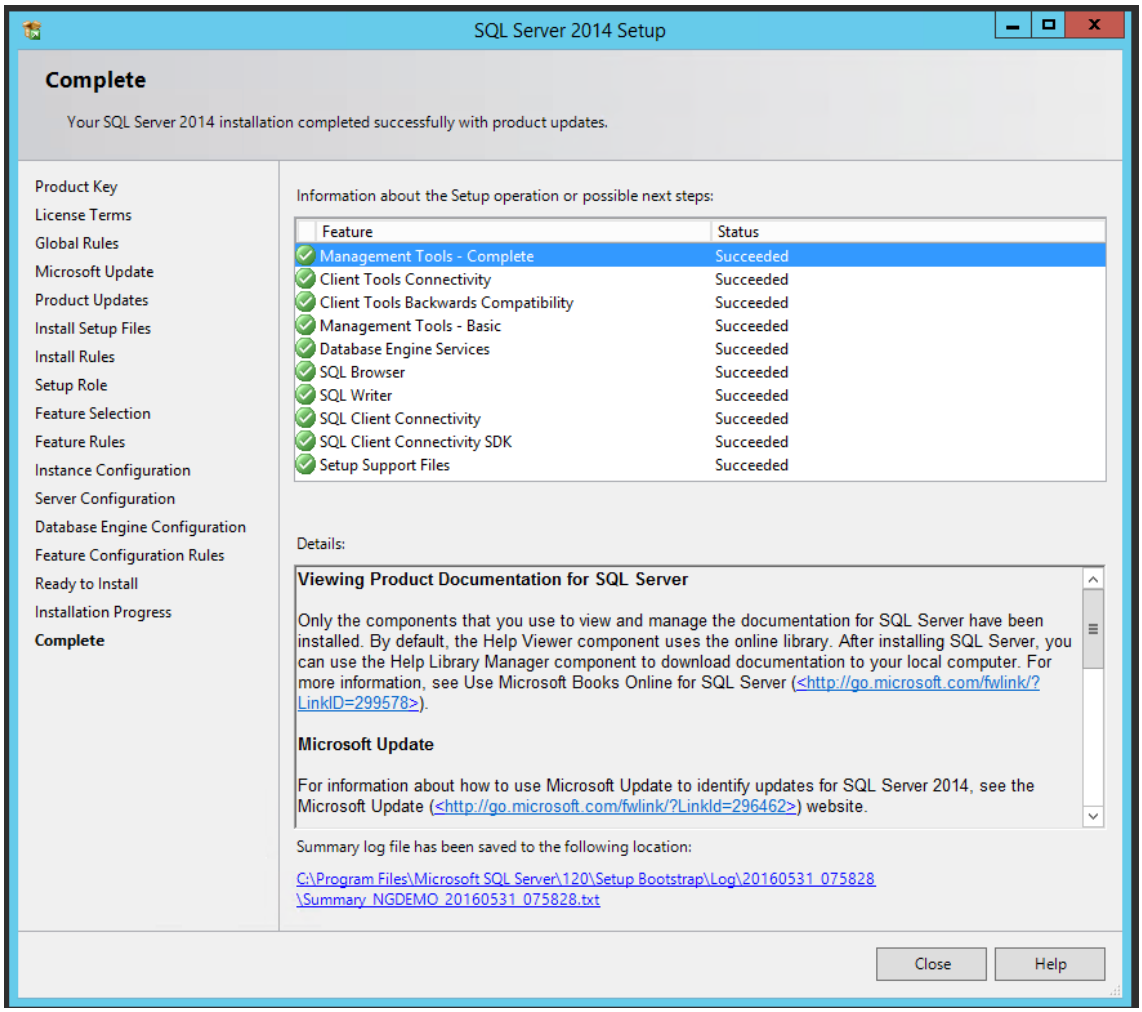
SQL Palvelimen tietokanta-asetukset



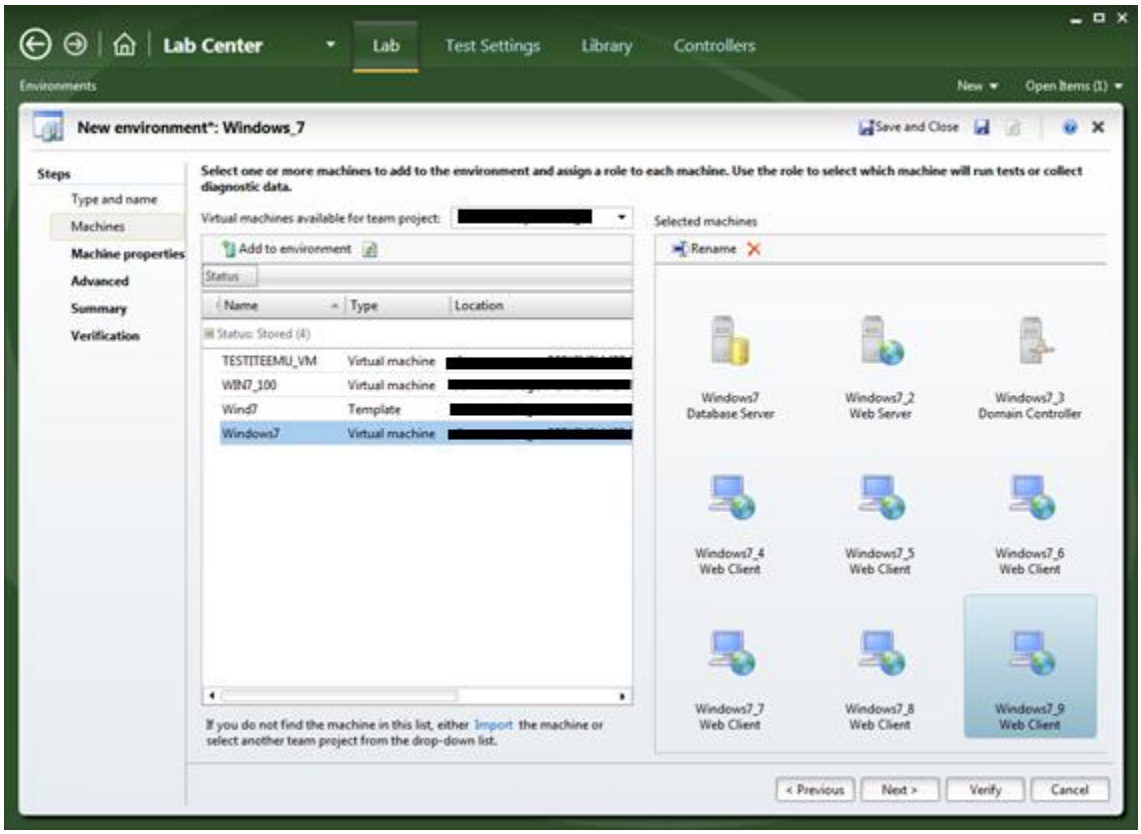
SQL Palvelimen asennuksen viimeistely



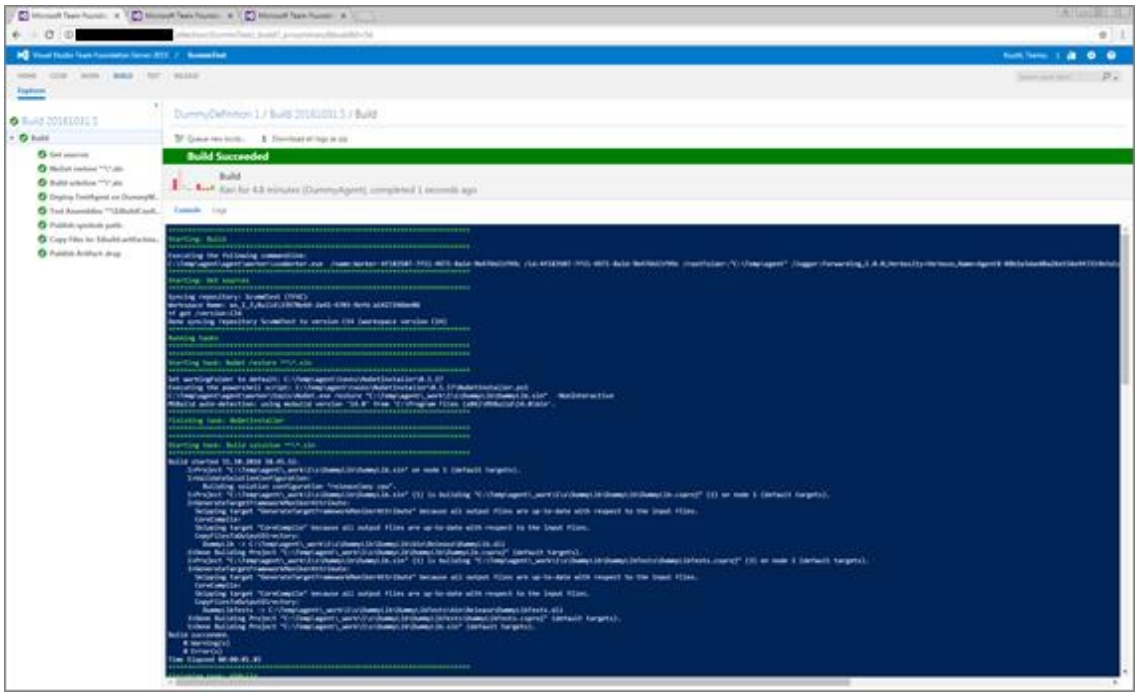
SQL Palvelimen asennus onnistui



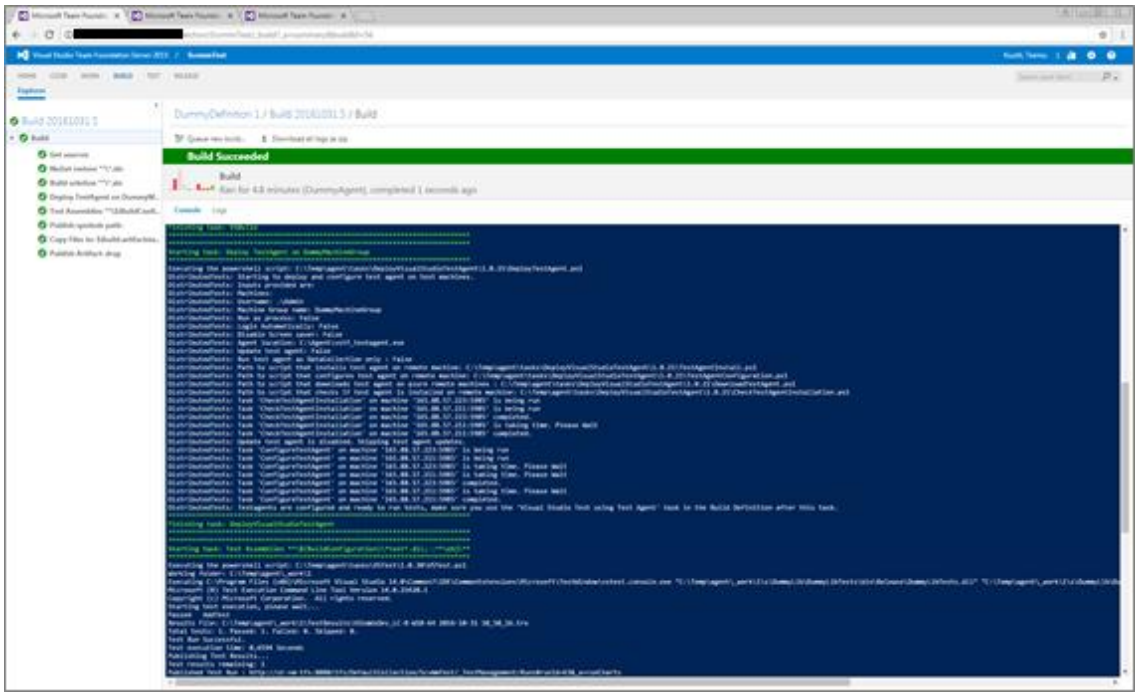
Virtuaalikoneiden lisääminen ympäristöön



Testiajon käynnistys



Testiajon suoritus



Testiajon loki: testi-agenttien käyttöönotto testiympäristöissä

2016-10-31T08:45:54.7313941Z Executing the powershell script: C:\Temp\agent\tasks\DeployVisualStudioTestAgent\1.0.21\DeployTestAgent.ps1
2016-10-31T08:45:54.8404416Z ##[debug]resourceFilteringMethod = machineNames
2016-10-31T08:45:55.2314863Z DistributedTests: Starting to deploy and configure test agent on test machines.
2016-10-31T08:45:55.2314863Z DistributedTests: Inputs provided are:
2016-10-31T08:45:55.2314863Z DistributedTests: Machines:
2016-10-31T08:45:55.2314863Z DistributedTests: Username: .\Admin
2016-10-31T08:45:55.2314863Z DistributedTests: Machine Group name: DummyMachineGroup
2016-10-31T08:45:55.2314863Z DistributedTests: Run as process: False
2016-10-31T08:45:55.2314863Z DistributedTests: Login Automatically: False
2016-10-31T08:45:55.2314863Z DistributedTests: Disable Screen saver: False
2016-10-31T08:45:55.2314863Z DistributedTests: Agent location: C:\Agent\vstf_testagent.exe
2016-10-31T08:45:55.2314863Z DistributedTests: Update test agent: False
2016-10-31T08:45:55.2314863Z DistributedTests: Run test agent as DataCollection only : False
2016-10-31T08:45:55.2314863Z DistributedTests: Path to script that installs test agent on remote machine: C:\Temp\agent\tasks\DeployVisualStudioTestAgent\1.0.21\TestAgentInstall.ps1
2016-10-31T08:45:55.2314863Z DistributedTests: Path to script that configures test agent on remote machine: C:\Temp\agent\tasks\DeployVisualStudioTestAgent\1.0.21\TestAgentConfiguration.ps1
2016-10-31T08:45:55.2314863Z DistributedTests: Path to script that downloads test agent on azure remote machines : C:\Temp\agent\tasks\DeployVisualStudioTestAgent\1.0.21\DownloadTestAgent.ps1
2016-10-31T08:45:55.2314863Z DistributedTests: Path to script that checks if test agent is installed on remote machine: C:\Temp\agent\tasks\DeployVisualStudioTestAgent\1.0.21\CheckTestAgentInstallation.ps1
2016-10-31T08:45:56.0275557Z DistributedTests: Task 'CheckTestAgentInstallation' on machine '165.88.57.223:5985' is being run
2016-10-31T08:45:56.0275557Z DistributedTests: Task 'CheckTestAgentInstallation' on machine '165.88.57.211:5985' is being run
2016-10-31T08:46:23.0123387Z DistributedTests: Task 'CheckTestAgentInstallation' on machine '165.88.57.223:5985' completed.
2016-10-31T08:46:56.0434207Z DistributedTests: Task 'CheckTestAgentInstallation' on machine '165.88.57.211:5985' is taking time. Please Wait
2016-10-31T08:47:55.9962799Z DistributedTests: Task 'CheckTestAgentInstallation' on machine '165.88.57.211:5985' completed.
2016-10-31T08:47:55.9962799Z DistributedTests: Update test agent is disabled. Skipping test agent updates.
2016-10-31T08:47:56.0272958Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.223:5985' is being run
2016-10-31T08:47:56.0272958Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.211:5985' is being run
2016-10-31T08:48:56.0426533Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.223:5985' is taking time. Please Wait
2016-10-31T08:48:56.0426533Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.211:5985' is taking time. Please Wait
2016-10-31T08:49:12.4021040Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.223:5985' completed.
2016-10-31T08:49:56.0432643Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.211:5985' is taking time. Please Wait
2016-10-31T08:50:14.1984515Z DistributedTests: Task 'ConfigureTestAgent' on machine '165.88.57.211:5985' completed.
2016-10-31T08:50:14.1984515Z DistributedTests: Testagents are configured and ready to run tests, make sure you use the 'Visual Studio Test using Test Agent' task in the Build Definition after this task.

Testiajon valmistuminen