AutomatedLab Architecture

Contents

[AutomatedLab Tutorial Part 1 – AutomatedLab Architecture 2](#_Toc401522689)

[Introduction 2](#_Toc401522690)

[Why AutomatedLab 2](#_Toc401522691)

[What AutomatedLab covers 2](#_Toc401522692)

[How it works 3](#_Toc401522693)

[The installer 3](#_Toc401522694)

[The PowerShell modules 4](#_Toc401522695)

[VHDs 5](#_Toc401522696)

[PowerShell Remoting 5](#_Toc401522697)

[AutomatedLab / PowerShell Remoting Requirements 5](#_Toc401522698)

[Checkpoints 6](#_Toc401522699)

[Post Installation Activities and remote command execution 6](#_Toc401522700)

[Software Packages 6](#_Toc401522701)

# AutomatedLab Tutorial Part 1 – AutomatedLab Architecture

## Introduction

### Why AutomatedLab

We all know the situation: Your Company wants to upgrade some software product or wants to introduce something new, and you need to test this in a lab environment that looks somehow similar like to your production environment. Let’s say you need to test the integration of a software product with Active Directory. No big deal. You install a hand full of servers and as you have something more to do than looking at the installation process, you do a lot of other tasks while setting up the lab. Because you do these (often undocumented) other tasks, you might miss one or two important settings. Settings that you cannot change, like the forest functional level or domain name. Now, you will need to start from scratch, and invest even more time. Or even worse, you do not realize that something went wrong with the setup, and you end up with unreliable test results.

It is a best practice to automate as much as possible. It is faster, more efficient and – what might be even more important – more reliable. If you run a script two times you will get the same result two times. However, if you setup a lab with 5 machines more than once, it is quite possible that these labs will have some differences.

The people working on AutomatedLab are consultants and field engineers who work with very different customers. On several occasions, it happens that we need to write some code based on specific infrastructure requirements. Or we have to troubleshoot a certain issue. Obviously having just one test environment does not work, as there are too many different designs, software version and customer specific configurations. Hence, setting up labs to include all of these parameters, can be quite time consuming.

### What AutomatedLab covers

AutomatedLab is a solution able to install a lab scenario in a very short time, AutomatedLab is a solution that can install a lab scenario in minutes, with the complexity of your choice. For example, setting up a domain with a single server or client takes about 15 minutes. Setting up a big lab with more than 10 machines takes about 1.5 to 2 hours, largely depending on the speed of your disk. You can specify all the important settings like names, IP addresses and network configuration, OS version, forest functional level, etc. You can also assign roles to machines like IIS, Exchange Server or Domain Controller. Installation of any machine with any role in AutomatedLab, is of course, unattended.

To enable you to get started quickly, AutomatedLab comes with a bunch of sample installation scripts covering very different scenarios. There are sample installation scripts for simple things like single clients or domains with just one member machine. Also, there are sample installation scripts which install Exchange 2013, SQL Server 2012, Orchestrator 2012, a client with Visual Studio 2013 and a web server. Imagine how long it would take to install all of this manually. AutomatedLab does it in about 2 hours on an SSD drive.

NOTE: The purpose of AutomatedLab is installing lab and test environments. It is not meant for production scenarios.

## How it works

### The installer

AutomatedLab is provided as an MSI file and can be downloaded from <http://automatedlab.coplex.com>. The installer performs four actions:

* PowerShell Modules  
  90% of AL is just pure PowerShell code, however some stuff had to be done in C#. PowerShell modules provides a very nice and easy way to creating script packages. If you need more information about how Powershell modules are working, take a look at [about\_Modules](http://technet.microsoft.com/en-us/library/hh847804.aspx). AutomatedLab comes with 7 PowerShell modules that will be covered in more detail later in this article. These modules need to be in the ModulePath to get auto-loaded. The default location of PowerShell modules being installed, is in the WindowsPowerShell folder of the documents folder of the Windows user installing AutomatedLab.
* Sample Scripts  
  There are a number of sample scripts that demonstrate the capabilities of AL. You may have to change a sample scripts prior running it to make it work on your machine. Each sample script looks for a folder called ‘LabSources’ on drive E and tries to install the VMs on drive D. Please change these drives to match your drives in your computer.  
  Of course, you can also remove or add machines. This requires minimal knowledge of PowerShell, but is merely just a copy and paste of the calls to ‘Add-LabMachineDefinition’ and change the parameters to the values of your choice.
* Documentation

Yes, that’s the documentation. By default, this will be copied to the documents folder of the Windows user installing AutomatedLab.

* Lab Sources

This is a folder hierarchy created by the installer to provide all the sources for installing a lab environment:

* + ISOs  
    Place all ISO images referred to in the installation script in this folder. If you download images from MSDN or other sources, the names may vary. So please make sure you change the installation script accordingly.
  + PostInstallationActivities

AutomatedLab comes with some built-in role dependent Post Installation Activity like creating Active Directory users, install the well-known sample databases on SQL or creating trust relationships between all installed Active Directory forests. How to write your own Post Installation Activity, will be explained later.

* + SoftwarePackages

AutomatedLab provides you with an easy way to install a piece of software on some or all lab machines. All you need to know is how to install it silently. For example, to install Notepad++ on all lab machines, you just need two lines of code and a copy of the installer executable for Notepad++ in this folder.

* + Tools

AutomatedLab can copy a folder to the VM while creating it. By default, AutomatedLab will install the SysInternals tools on each VM using this Tool folder. Everything you put into here is going to be copied to each VM (to the folder C:\Tools of the VM). The next articles will tell more about this.

### The PowerShell modules

AutomatedLab comes with 7 PowerShell modules. Why do we have 7 modules and not just one like most products? We have tried to separate the solution into its main building blocks. This makes the coding and troubleshooting easier.

The next article will explain how to use the cmdlets in each PowerShell module. This article gives you an overview, but is not explaining how to create your own lab.

Let’s discover the PowerShell modules one by one.

* AutomatedLabDefinition
* This module is for gathering your requirements. It contains cmdlets for defining domains, machines, roles, virtual network adapters, ISO images and Post Installation Activities. AutomatedLab is based on XML files. However, the cmdlets provided with AutomatedLabDefinition, enables you to define the desired configuration using PowerShell cmdlets. Hence, no XML knowledge is required. If required, once the configuration is complete, you can export it (to XML file), and make it persistent.
* AutomatedLab

This is the main module. It starts all the actions based on the configuration you have created (and exported to a lab XML file) with AutomatedLabDefintion. After this export, AutomatedLab needs to import the lab XML file. This task is performing a number of validations to make sure the specifications is valid (like checking for duplicate names or IP addresses and making sure that all paths specified, are valid). If all validations are successful, the actual lab installation can be started. All the details will be covered in the next article.

* AutomatedLabUnattended

The actual installation of the operating systems is performed using classical unattended setup. This is based on an XML file containing the machine details. AutomatedLabUnattended modifies the standard XML file and is used only internally. There is no meaningful use case outside of AutomatedLab.

* AutomatedLabWorker

We have tried to separate the main parts of the solution into two modules. AutomatedLab is based on XML files, and has connections to most of the other PowerShell modules whereas AutomatedLabWorker is more independent and handles workloads. These workloads are triggered by functions of AutomatedLab. The AutomatedLabWorker do not have any relationships with the XML files, and the intelligence build around the lab.

* HostsFile  
  AutomatedLab is heavily based on PowerShell Remoting (explained later). And PowerShell Remoting needs proper name resolution. As we are creating labs we cannot use the corporate DNS and furthermore, we do not want the lab machines in DNS. That’s why AutomatedLab uses the good old hosts file to make sure it can contact the machines. The names of the lab machines, and thereby the entries in the hosts file, are short names (not fully qualified names). This is to avoid Kerberos being attempted used when authenticating to the lab machines.
* PSFileTransfer

Any time AutomatedLab copies something to a lab machine, it uses PowerShell Remoting, never SMB. This module is based on some functions by Lee Holmes.

* PSLog

This module does verbose logging and also writes all messages into a central location

### VHDs

AutomatedLab uses differential disks to save disk space and speeding up the overall installation process. This requires AutomatedLab to create the parent disks first – one disk per operating system. So if you have a lab with a 10 machines where some machines are running Windows Server 2012R2, some machines are running 2008R2 and one client is running Windows 8.1 you will have three parent disks. All VMs refer to one of these parent disks. When starting the lab deployment you will see, that we interact with the VHDs, e.g. mount them.

### PowerShell Remoting

AutomatedLab requires an internal or external virtual network adapter for the VMs to use. A private virtual network adapter is not supported as AutomatedLab (on the host machine) needs to be able to speak to the machines.

Creating the VHDs and the VMs is performed by the host. After starting the lab machines they become available. And thanks to the modified hosts file, AutomatedLab can contact the machines by name.

Leveraging PowerShell Remoting, AutomatedLab can now invoke any command or script block on the lab machines. This is similar to PSExec, but way more comfortable and powerful. There are lots of resources on the internet and a first start could be “[about\_Remote](http://technet.microsoft.com/en-us/library/hh847900.aspx)” and the “[PowerShell 2.0 remoting guide](http://www.ravichaganti.com/blog/?p=1025)”.

Example of installing a domain controller:

* AutomatedLab first creates a parent disk (if this doesn’t exist already)
* AutomatedLab creates the differential disk and the VM
* AutomatedLab starts the VM and waits until the machine is reachable (ICMP and WSMan)
* When the machine is reachable, AutomatedLab running on the host machine sends the script block for domain controller promotion to the lab machine, and invokes it
* AutomatedLab waits until the machine has restarted
* AutomatedLab waits for the machine to be reachable again
* That’s it then. The domain controller installed.

Some actions requires a lab computer to contact another lab computer. This requires a so called double-hop authentication which is not enabled by default for security reasons. AutomatedLab uses CredSSP to make this work. “[PowerShell 2.0 remoting guide: Part 12 – Using CredSSP for multi-hop” authentication](http://www.ravichaganti.com/blog/?p=1230) contains more information about CredSSP.

### AutomatedLab / PowerShell Remoting Requirements

There are a couple of things that needs to be configured on the host machine before remoting works in a lab scenario. By default, PowerShell Remoting uses Kerberos for authentication. Kerberos will not work between the host and the lab machines due to a number of requirements not being fulfilled. That’s why AutomatedLab uses NTLM.

However, by default, PowerShell Remoting connect only to trusted machines. “Trusted” relates to the domain membership. However the lab machines are either standalone or member of an Active Directory domain that is not trusted by the domain of the host machine. Hence, the TrustedHosts settings has to be modified, to allow the host machine to talk to the lab machines.

Before the lab installation can be started, the cmdlet ‘Set-LabHostRemoting’ must be called (this call is part of any AutomatedLab sample script). This function enables PowerShell Remoting on the host computer and sets TrustedHosts to “\*” which means we can connect (authenticate) to any computer using PowerShell Remoting. Additionally unencrypted traffic will be allowed and CredSSP authentication enabled, so that the host computer is able to send the necessary credentials to the lab machines, which can pass it further to another lab machine.

### Checkpoints

If you want to test something dangerous in a lab, you may want to create a checkpoint first. Creating a checkpoint manually of one or two machines is not a big deal. But what if your lab contains 10 or 20 machines? Then you might want to use the functions made for this; ‘Checkpoint-LabVM’ and ‘Restore-LabVMSnapshot’. These functions can create and restore checkpoints for all lab machines when you specify the ‘All’ switch. If a certain checkpoint is no longer needed, you can use the function ‘Remove-LabVMSnapshot’. Hence, this is a rapid way of freezing the state of a full test lab.

Note: When you plan to snapshot and restore domain controllers, make sure you have an understanding about USN rollback and Virtualization safeguards introduced in Server 2012. By default the GenerationID feature is disabled on all Windows Server 2012 Domain Controllers setup by AutomatedLab.

### Post Installation Activities and remote command execution

To make it easy to perform actions (configurations) on lab machines after all (or some) of the lab machines has been installed, Post Installation Activities is used for this. A PostInstallationActivity is a task that can be executed after a machine is ready and any (optional) role is installed. A Post Installation Activity calls the specified script on the machine it is mapped to. You do not need to care about authentication as this is handled by AutomatedLab. If your script is based on files or needs to access an ISO file, this is automatically copied or mounted for you. The next articles will explain this in more detail.

### Software Packages

Another cumbersome task when setting up labs, is installing software on all the machines. AutomatedLab takes over this task. All you need to provide, is the name of the executable or MSI to install. If using an executable (exe), you need the command line parameter to switch into a silent installation. Now, it is a one-liner to install the software package on all lab machines. The sample scripts are covering this topic.

The next part of this series goes through the installation of a simple lab environment. You will learn about the many functions AutomatedLab provides, and have an installed lab at the end.