MC2=My Computer on the Cloud – Phase 1



Admin manual

Solution Details

Prepared for

Italian Universities

14-Jun-17

Version 1.0

Prepared by

**???**

Consultant

?

Revision

Change Record

| Date | Author | Version | Change Reference |
| --- | --- | --- | --- |

Reviewers

| Name | Version Approved | Position | Date |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 1](#_Toc485220668)

[2 Phase 1 Solution overview 2](#_Toc485220669)

[2.1 Summary for lab configuration and script settings 2](#_Toc485220670)

[2.1.1 Self-service scenario 3](#_Toc485220671)

[2.1.2 Classroom scenario 3](#_Toc485220672)

[3 Lab creation 4](#_Toc485220673)

[4 Base image creation 7](#_Toc485220674)

[4.1 Procedure 7](#_Toc485220675)

[4.2 Create the VM for the first base image 8](#_Toc485220676)

[4.3 Automatic shutdown on idle 10](#_Toc485220677)

[4.3.1 Final version 11](#_Toc485220678)

[4.3.2 Alternative version 15](#_Toc485220679)

[5 Active directory groups and roles 17](#_Toc485220680)

[5.1 Assign access to resources to AD groups 17](#_Toc485220681)

[6 VM scheduling automation 20](#_Toc485220682)

[6.1 Create an Automation Account 20](#_Toc485220683)

[6.2 Import a runbook 23](#_Toc485220684)

[6.3 Authentication method 24](#_Toc485220685)

[6.4 Execute a runbook on demand 26](#_Toc485220686)

[6.5 Schedule a runbook execution 27](#_Toc485220687)

[7 Access to the lab 31](#_Toc485220688)

[7.1 List of scripts 32](#_Toc485220689)

[7.1.1 Add-AzureDtlVmAutoVar: creation of VMs 32](#_Toc485220690)

[7.1.2 DeallocateStoppedVM 37](#_Toc485220691)

[7.1.3 Manage-AzureDtlFixedPool 37](#_Toc485220692)

[7.1.4 Test-AzureDtlVM 37](#_Toc485220693)

[Glossary 41](#_Toc485220694)

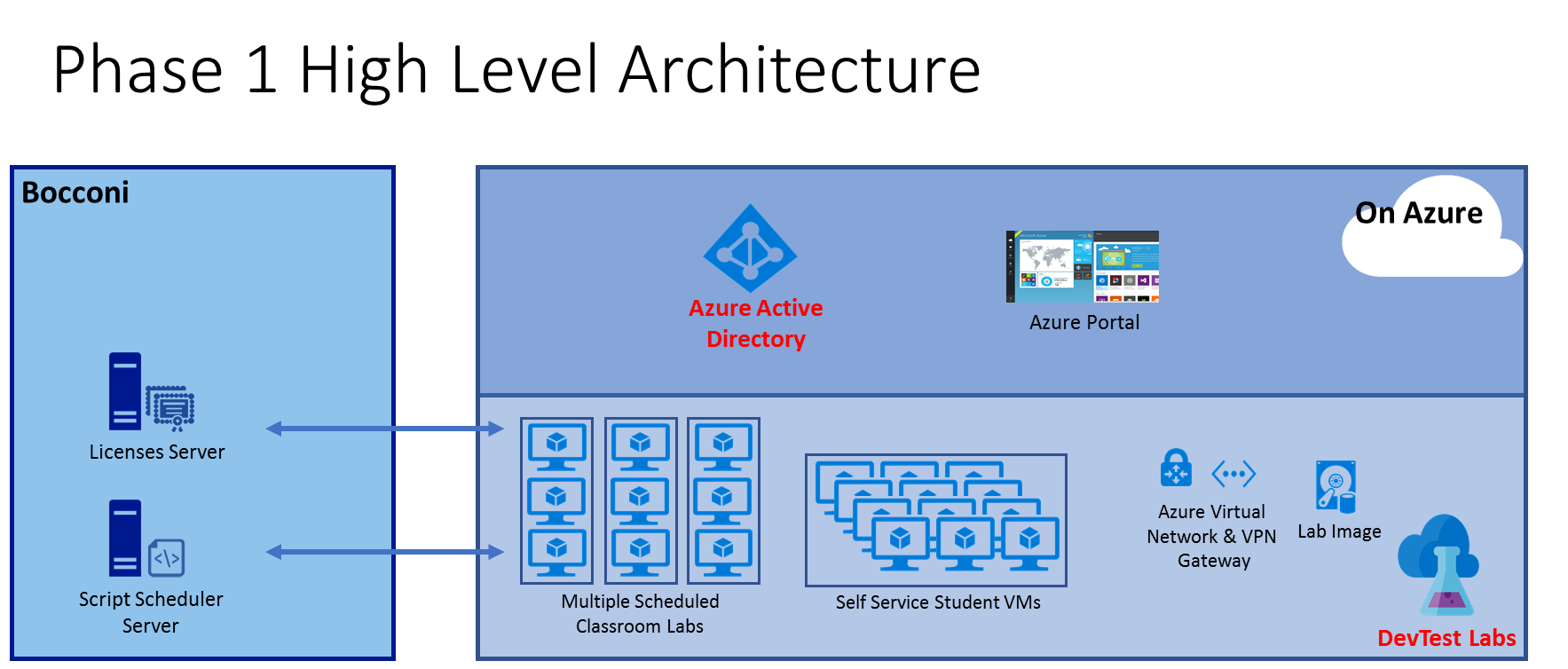
1. Introduction

The document describes the solution implemented for the Italian universities regarding the adoption of **Azure DevTest Labs** (DTL) in the university environment for the Phase 1 of the MC2 project.

The solution gives IT administrators the ability to create and manage the virtual machines within the labs and schedule the creation of the VMs through automated scripts. The document represents a guide for IT administrators to manage the solution autonomously, in order to fit the purpose of both the classroom teaching lessons and the self-service scenario for the students.

1. Phase 1 Solution overview

Following is the high-level diagram of the phase 1 architecture implemented in the first phase of the project, as previously agreed.



As we will describe in the following chapters, the solution leverages the Azure DevTest Labs (DTL) to provide fresh virtual machines, ready to use for the students and teachers. The management of the VMs, consisting in the creation, deallocation and destruction of the machines at specific time, is provided through some scripts. Then, to give IT admins the flexibility to schedule the provisioning according to the courses’ needs, we decided to use the **Runbooks** inside the **Azure Automation** account instead of using a script scheduler server. This service allows the admin to manage the script scheduling and track the activity logs within the Azure Portal in an easier way, without accessing a scheduler VM every time it is needed.

* 1. Summary for lab configuration and script settings

This is a quick guide of the steps needed for the lab configuration for both the self-service scenario and the classroom scenario. All the details are described in the next chapters.

* + 1. Self-service scenario

Here is the list of the steps for the self-service lab:

* Create the DevTest lab (Chapter 3)
* Create the base image with the shutdown on idle task configured on the machine (Chapter 4.1 to 4.4)
* Create the Azure Active Directory group and add permissions to the self-service lab to that group (Chapter 5)
* Add users to the group manually
* Create a set of schedules for the VM creation (Chapter 7.1.1)
* Create a schedule for the deallocate of the stopped VM (Chapter 7.1.2)
  + 1. Classroom scenario

Here is the list of the steps for the class labs:

* Create the DevTest lab for each class lab (Chapter 3)
* Create the base image without the shutdown on idle task configured on the machine (Chapter 4.1 to 4.3). As described, copy the VHD in each lab
* Create the Azure Active Directory group and add permissions to the class lab to that group (Chapter 5). Do this step for each lab
* Add users to the group manually
* Create a set of schedules for the VM creation (Chapter 7) for each class lab, according to the lessons calendar

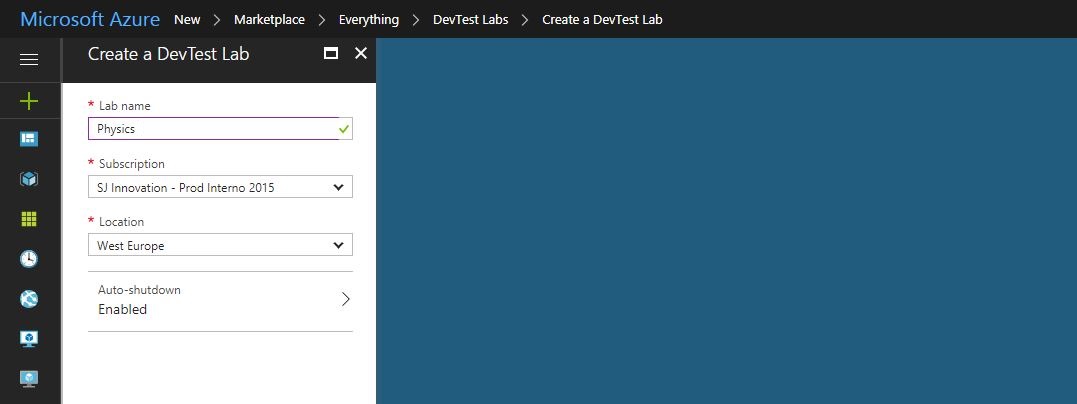
1. Lab creation

The first step for the implementation of the solution is the creation of the DevTest Lab. This step is performed once for each lab, so it can be done manually from the Azure portal.

To create the lab, you need to click on the “+” button, then choose the DevTest Labs resource and Create.

Microsoft Azure 
New 
Marketplace 
Everything 
Compute 
Networking 
Storage 
Web + Mobile 
Databases 
Data + Analytics 
Marketplace 
> Everything > DevTest Labs 
X Everything 
Y Filter 
p DevTest Labs 
Results 
NAME 
DevTest Labs 
DevTest Labs for 310ckchain as a Service 
Related to your search v 
Template deployment 
M nft 
PUBLISHER 
Microsoft 
Microsoft 
Search resources 
CATEGORY 
Developer tools 
Developer tools 
O 
c.cauterucci@s 
DevTest Labs 
DevTest Labs helps developers and testers to quickly create virtual machines in Azure to deploy and 
test their applications. You can easily provision Windows and Linux machines using reusable 
templates while minimizing waste and controlling cost. 
Quickly provision development and test virtual machines 
Minimize waste with quotas and policies 
Set automated shutdowns to minimize costs 
Create a VM in a few clicks with reusable templates 
Get going quickly using VMS from pre-created pools 
Build Windows and Linux virtual machines 
PUBLISHER 
USEFUL LINKS 
Create 
Microsoft 
DevTest Labs 
Al + Cognitive Services 
Internet of Things 
Enterprise Integration 
Security + Identity 
Developer tools 
Monitoring + Management 
Add-ons 
0 
Containers 
Blockchain 
ops Monitor Enterprise 
ops 
ops AB 

At this step, you have to choose the Lab name (note that it cannot be changed later) as well as other standard Azure parameters (subscription, location). Feel free to ignore the auto-shutdown property at this point, because it will be set by the script during the creation of the VMs:

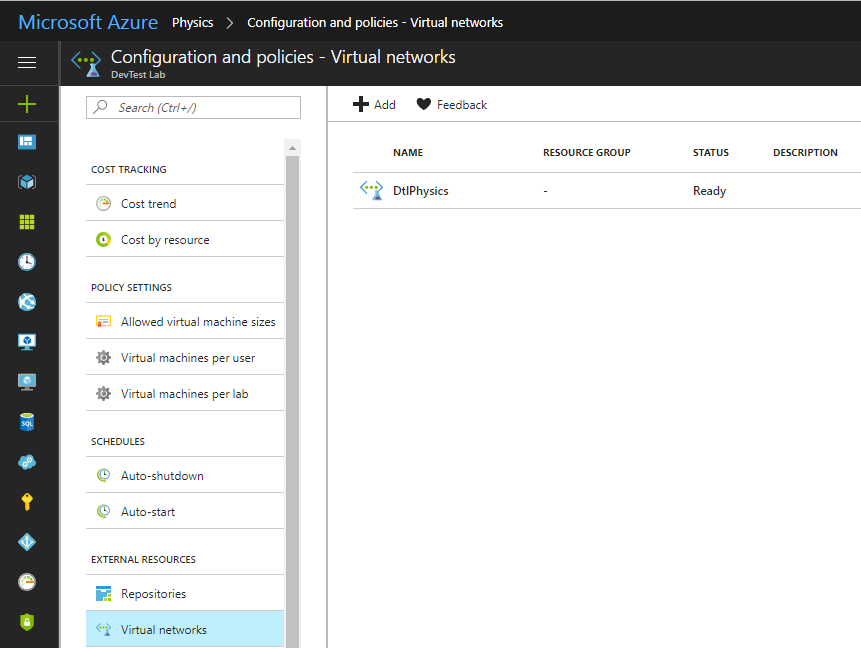


<https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-create-lab>

After the creation of the lab, we must give the VMs the access to the University VNET, if it already exists.

By default, when you create a new lab, the system creates a new VNET with the lab name. However, in our scenario we need to connect the VMs to the previous University VNET.

To do this, inside the lab view you need to click on the “Configuration and policies” blade, then under the Virtual Networks tab you need to click on Add:



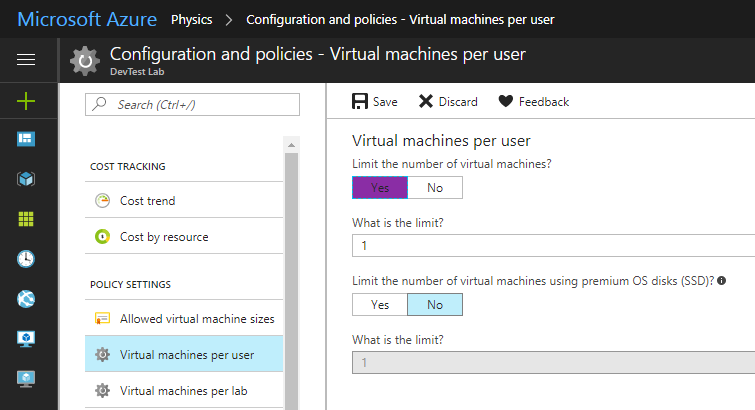
Then, choose the existing DtlUniversityVLab. After this, it is important to set the following configuration to the subnet “DTLUniversityVlabVMs01”, which will be used for the labs:

In details, you need to explicitly indicate the DTLUniversityVlabVMs01 as the subnet used in the VMs creation, you need to enable the use of shared public IPs and allow the public IP creation.

Then, you can save and close the configuration. At the end, it is important to remove the previous VNET, to allow VMs use only the University VNET:

If needed, you can access other configuration to the labs. One of these that are provided for the University scenario is the limit on the VMs per user and on the type of VM.

You can configure this by accessing the section “Configuration and policies” by modifying the policy settings “Allowed virtual machine sizes” and “Virtual machines per user”, respectively:



In this way, if you try to create, either manually or with a script, a VM with a size different from specified you will get an error. Similarly, if a user tries to claim a VM when he already has claimed a machine, it will get an error.

Since each lab represent a separate scenario (e.g. self-service lab, Stats classroom lab, SPSS classroom lab, etc…), you need to follow this process for each lab. As we said, the steps for the creation labs are done only once. As we will describe later, the script solution provides the creation and the deletion of the VMs inside a lab, but the lab itself is not destroyed every day.

1. Base image creation

Each VM inside the lab is created using a base image. It can be taken from the Marketplace, or it can be a custom image created from a previous VHD.

For both the classroom and the self-service scenario, the base image is created from a custom image, which contains all the scientific software already installed as well as other environment configuration. We suppose that the base image is identical for all the VMs in the same lab.

For different labs, the IT admin can provide a different base image, to fit the purpose of the scenarios. For example, one of the difference between the base image for the self-service lab and the classroom labs is the possibility to shut down the VM when it is in idle: as we will show, this can be managed with a Windows task in the self-service lab, while it is not available in the classroom labs’ base image.

* 1. Procedure

In order to create the very first base image, the best way would be using a clean Windows Server image from the Marketplace, install all the software on it and, finally, create the custom base image from the VM. Then, the final base image can be copied as VHD inside each lab using the Azure tools.

We did this step using the “**Physics**” DevTest Lab, where a VM was created from the Marketplace, customized with the university software, then deallocated from the Portal and, finally, used as one of the base image for the lab. Starting from this image, its VHD was copied inside the final labs.

Note that it can be useful to maintain one VM used to create the base image, even in deallocated status, in a separate lab and retain it. This is because, as we will show, all the VMs are deleted in each night. So, every modification done at the VMs of either the classroom labs or the self-service lab will be lost after one day.

Let’s see how to create the base image and move it through the labs.

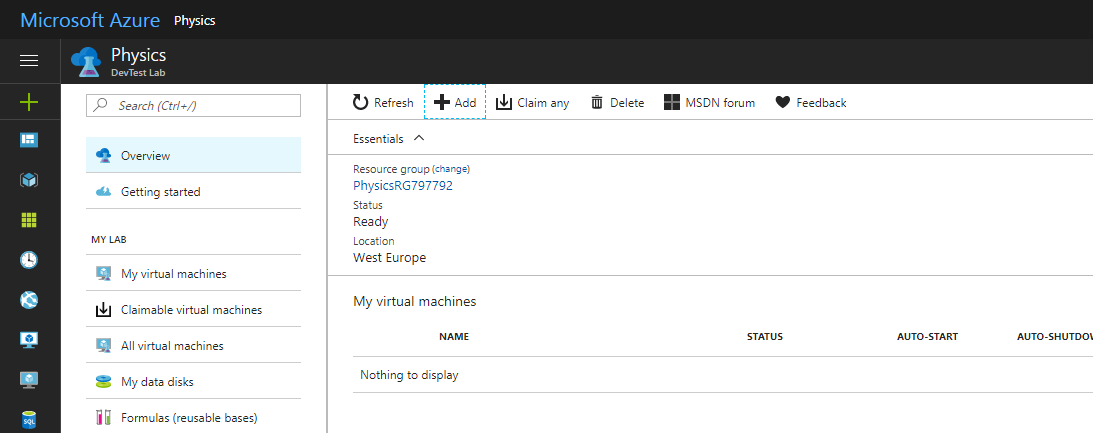
* 1. Create the VM for the first base image

First, we can use a clean lab for the very first image by creating a VM manually from the Marketplace. For the activities, you can use a Windows Server 2016 – Nano Server image.

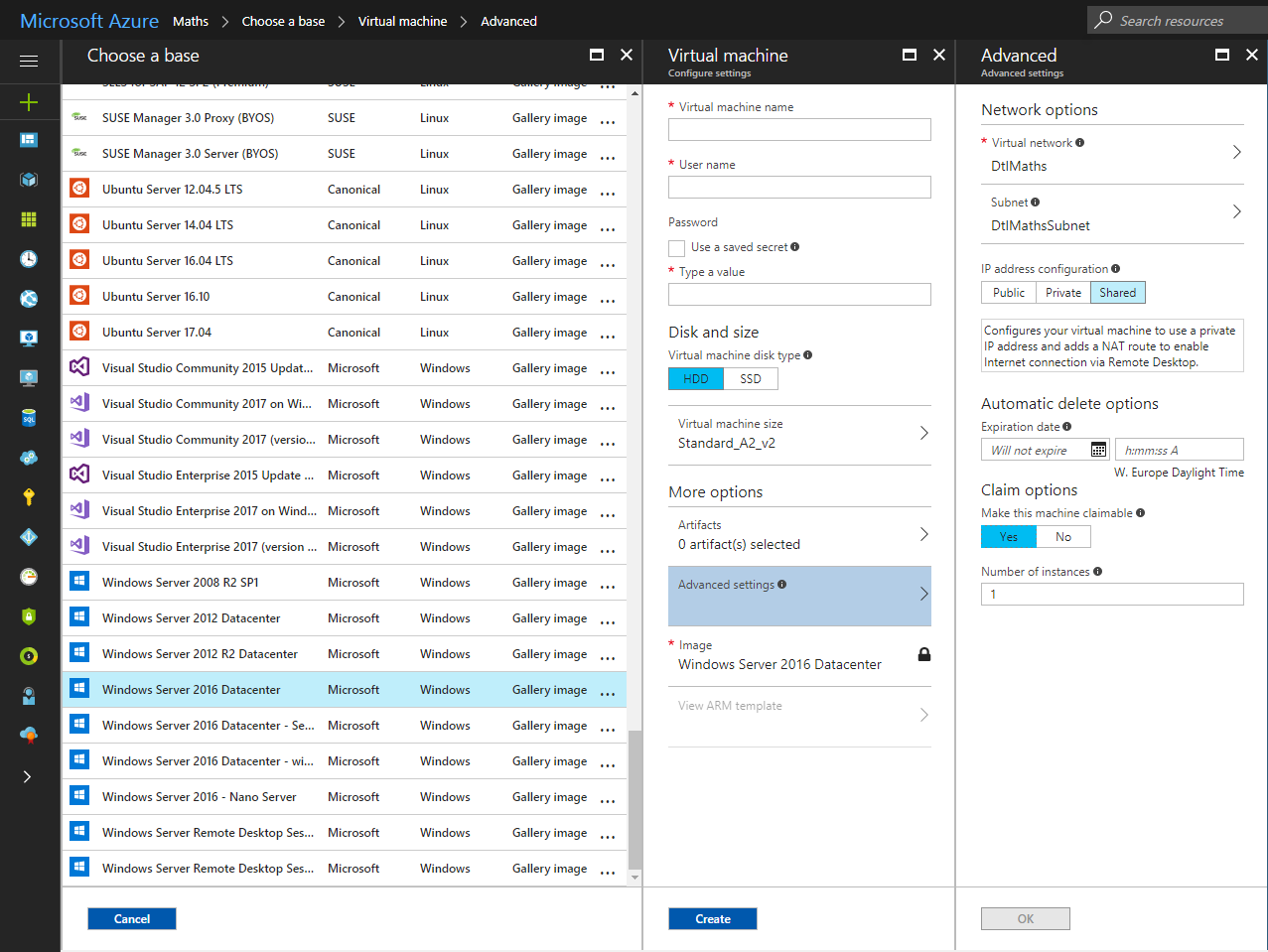
The steps for the manual creation of the VM follow the standard process:

<https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-add-vm>

Inside the lab, you need to click on Add:

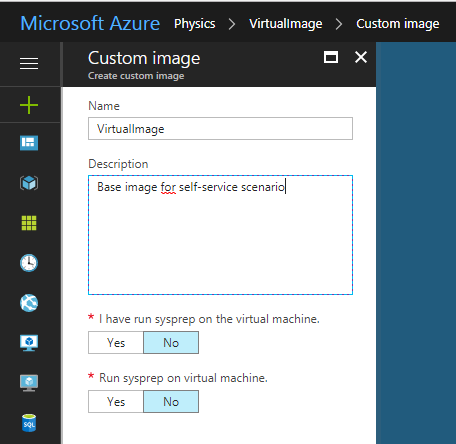


Then choose the image you need from the Marketplace. At this step, you are free to choose the settings you want for the VM. Regarding the Virtual network, you can choose the default “DtlPhysics” network and the subnet “DtlPhysicsSubnet”. Make sure you select “yes” for the claim options under the advanced settings ribbon and the virtual machine size is the same as the ones you are going to create.



Then, after having customized the VM, you create a custom image from it. First, stop the VM from the portal. NOTE: Remember to stop it from the portal instead of shut down the OS.

Select it and then, from the VM blade click on “Create custom image (VHD)”. Choose a name and do not set properties for sysprep:



At this point, you have created a custom image for your clean lab.

* 1. Automatic shutdown on idle

One of the features provided for the **self-service lab scenario** is to automatically shut down the VM OS when it is not used from the user. The machine idle is measured against both the user interaction and the use of the CPU.

NOTE: this task must be disabled or deleted in the base image used for the **classroom labs**

In order to give flexibility in changing the shutdown period or, in general, task parameters, we provided a mechanism to create, update and delete Windows tasks using scripts. This way, the IT admin can modify a parameter of the task directly in the file on the storage account; then, the modification is applied to every VM at system startup using a second task.

The following section shows two different implementation we provided for Bocconi to check the idle and manage the shutdown. Note that you must use only one of them.

IMPORTANT NOTE: all the materials is under the Azure storage account “vlabresources” under the container “content”:

<https://vlabresources.blob.core.windows.net/content>

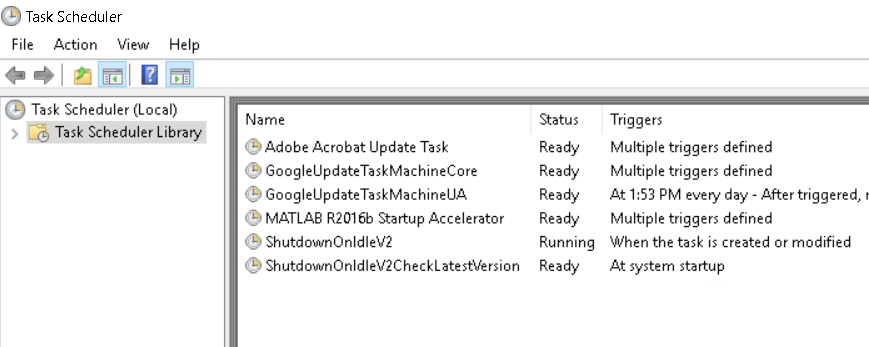
* + 1. Final version

In order to provide a method for checking the machine idle in an accurate way, we created a set of scripts scheduled with Windows Task Scheduler. These script leverages Windows API to detect the user interaction with the system and the Windows counters to measure CPU and disk idle time in a target period. We created this solution to give an accurate and, at the same time, flexible way for admin to configure the script depending on the scenario needs.

Let’s see how it is configured.

First of all, the tasks are run as administrator, so the students, who access with non-privileged credentials, cannot change the tasks by themselves.

To see the tasks, run the Windows Task Scheduler as administrator:



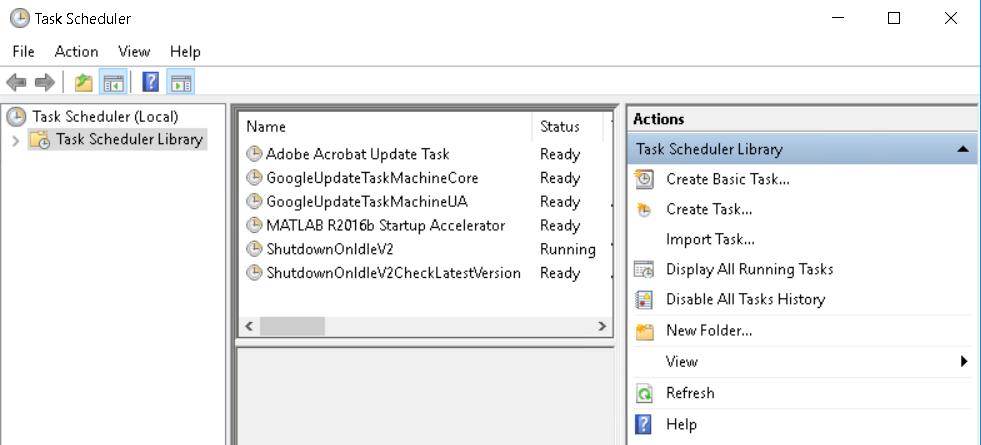
In this version, we use two Windows tasks and two scripts. The following is the workflow:

1. At system startup, **ShutdownOnIdleV2CheckLatestVersion** task runs the script “LoadIdleScript.ps1”
2. **LoadIdleScript.ps1** script downloads the latest version of the script “ShutdownOnIdleV2.ps1” and the task “ShutdownOnIdleV2.xml” from an Azure Storage account indicated as parameter (default is “vlabresources” in the container “content”).  
   Then, it updates (or create, if not exists) the task “ShutdownOnIdleV2”
3. At the task creation, **ShutdownOnIdleV2** task launches the script “ShutdownOnIdleV2.ps1”
4. **ShutdownOnIdleV2.ps1** script continuously check if the machine is in idle based on the user interaction and the CPU and disk usage.

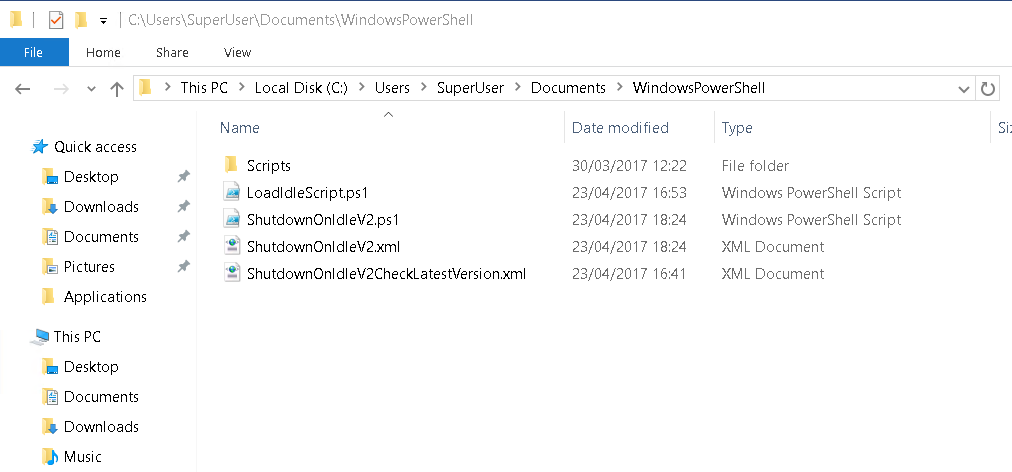
**IMPORTANT NOTE**: to configure the machine to use this script system, you must manually do these two actions:

1. set the **ShutdownOnIdleV2CheckLatestVersion** task by importing the XML definition we provided
2. copy the **LoadIdleScript.ps1** in the folder indicated in the previous task (by default, we put all the scripts in the folder “C:\Users\SuperUser\Documents\WindowsPowerShell”)

To import a task you can use the Windows Task Scheduler UI by clicking on “Import Task”:



Then, copy the **LoadIdleScript.ps1** in the specified folder (if you want, you can change the location by changing the parameter inside the XML task definition). By default, the script as well as the task XML resoruces are located in the same folder:



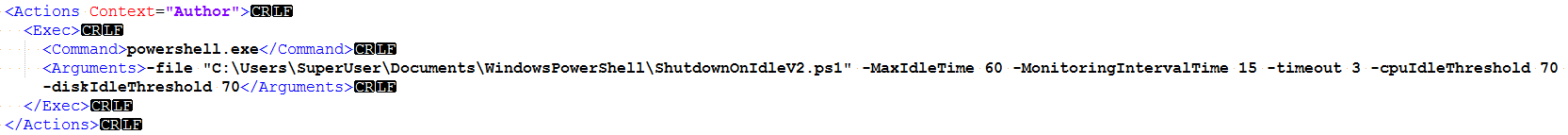
* + - 1. Configure the parameters for shutdown

The following are the parameters of the ShutdownOnIdleV2.ps1 script that you can change and configure as you want:

* ***MaxIdleTime***: maximum allowed time for machine idle (in minutes) (default = 60)
* ***MonitoringIntervalTime***: interval for CPU and disk monitoring (in minutes). It cannot be more than 15 minutes. (default = 15)
* ***Timeout***: Timeout before shutdown after waited for MaxIdleTime (in minutes). No more than 10 minutes (default = 3)
* ***cpuIdleThreshold***: average min %CPU on idle during MonitoringInterval to be considered in idle (default = 70)
* ***diskIdleThreshold***: average min %disk on idle during MonitoringInterval to be considered in idle (default = 70)

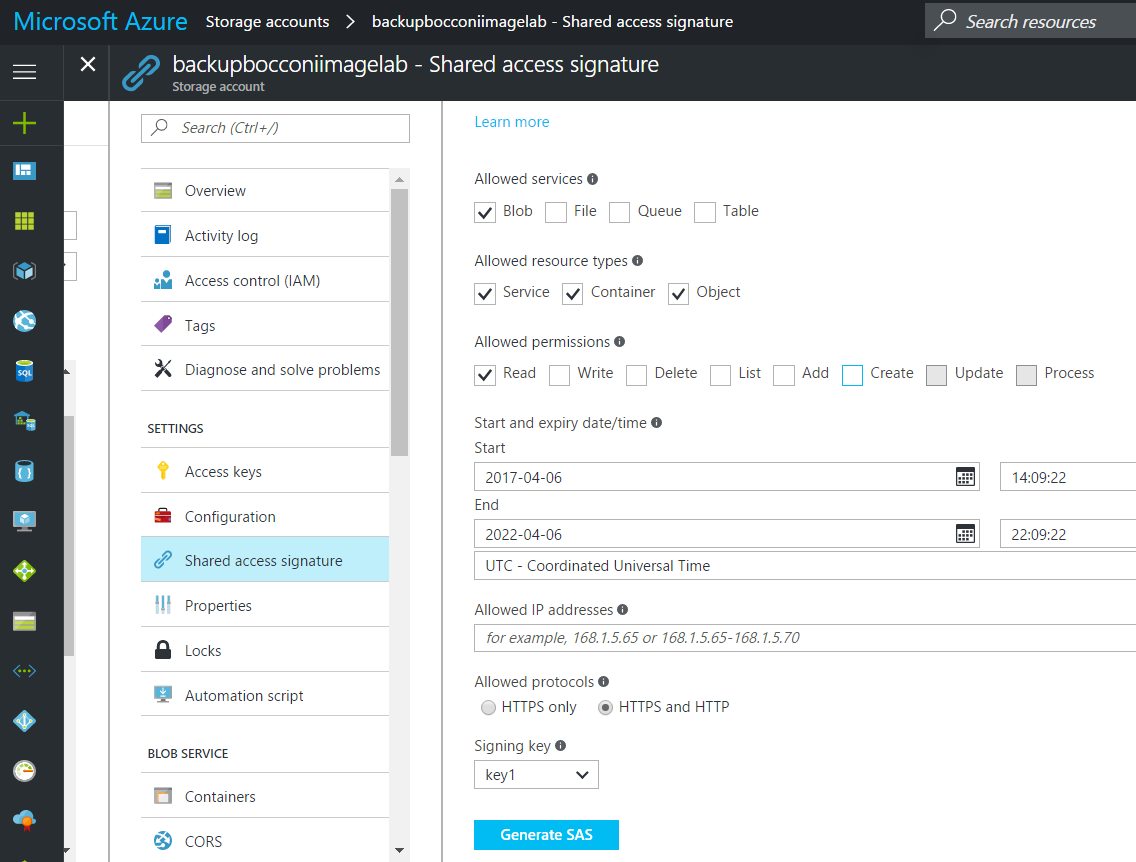
These parameters need to be passed in the **Actions** section of the XML definition of the ShutdownOnIdleV2 task. If any parameters are passed, then the default values defined in the script are used.

This is an example of a configuration that uses the same values indicated before:



As you can see, you can set these parameters directly in the “ShutdownOnIdleV2.xml” file. When you want to change the configuration for a machine, the only thing you need to do is to change the parameters and upload the file in the specified Azure storage account by overwriting the previous version. Then, at the system startup (either when rebooting or starting the machine), if the “ShutdownOnIdleV2CheckLatestVersion” task is configured and the “LoadIdleScript.ps1” is available in the local folder, the configuration will be updated.

Note that the access to the storage account is done through a read-only Shared Access Signature (SAS) token. This avoid the need to authenticate to Azure through the machine; it also gives more fine-grained access control by giving read-only access to the blob. Of course, if the storage account access key that signs the SAS token is regenerated for some reasons, you need to regenerate the SAS token as well. The following image shows where to find and configure the SAS token:



* + 1. Alternative version

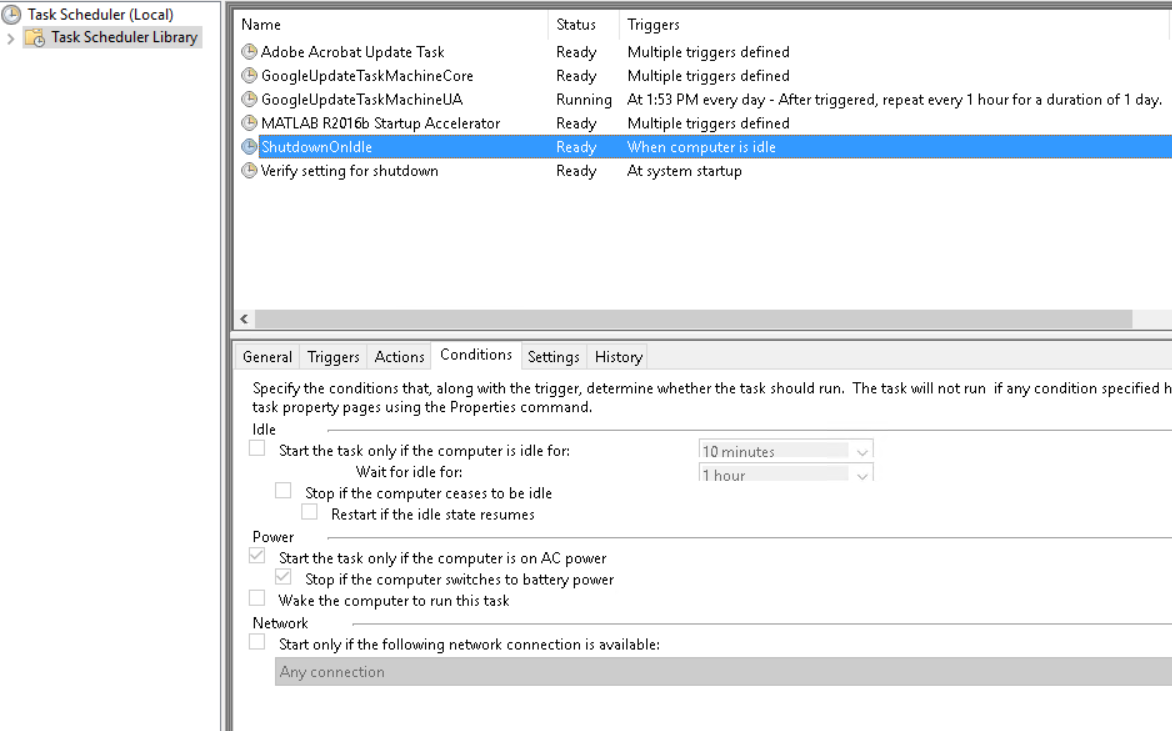
Another version for checking the machine idle is based on the standard idle trigger inside the Windows Task scheduler. From the documentation, “The Task Scheduler service will check if the computer is in an idle state every 15 minutes. A computer is considered to be in an idle state when a screen saver is running. If a screen saver is not running, then the computer is considered to be in an idle state if there is 0% CPU usage and 0% disk input or output for 90% of the past fifteen minutes and if there is no keyboard or mouse input during this period of time. Once the Task Scheduler service detects that the computer is in an idle state, the service only waits for user input to mark the end of the idle state.”

<https://technet.microsoft.com/en-us/library/cc721902(v=ws.11).aspx>

<https://msdn.microsoft.com/en-us/library/windows/desktop/aa383561(v=vs.85).aspx>

As before, the tasks are run as administrator, so the students, who access with non-privileged credentials, cannot change the tasks by themselves.

To see the tasks, run the Windows Task Scheduler as administrator:



In this version, we use two tasks, “**Verify setting for shutdown**” and the “**ShutdownOnIdle**” task. The first task is triggered at system startup: it executes a script named “changeTaskIdle\_remote.ps1”, which gets the latest version of the ShutdownOnIdle task from a XML document on the “vlabresources” storage account. Then, the script updates the existing ShutdownOnIdle task (if not existing, it will be created) from the XML file downloaded in the folder. Note that the script uses parameters, so you are flexible to change it in the Action section of the “verify setting” task.

1. Active directory groups and roles

In order to give students and teachers the right permissions for the labs, you need to create the Azure AD groups and associate the correct resources access and roles.

Students and teachers have the same needs, so we can associate both the type of users to the same role. For the scenario, we created a role named “**University DevTest Lab user**”. The role is derived from the standard DevTest Lab user role: a user can claim a VM, it can start and stop its own VM but cannot neither manage the VMs claimed by other users nor perform administrative tasks. In addition, the University DevTest Lab user cannot create VMs inside the lab.

Regarding the AD groups, you can organize the groups per labs: for example, you can create a group for the self-service lab, one for the Stats class, etc... In addition, to separate the management of the user, you can create two groups for each lab, one for students and one for teachers. Anyway, as we said, the role that will be assigned to both is the same.

* 1. Assign access to resources to AD groups

Currently, you cannot assign access to Azure resources from the UI. So, we created a script that allows IT admins to give programmatically the permissions to access lab resources to a specific group using the University DevTest lab role.

The script is the “**Add-GroupPermissionsDevTestLab.ps1**”. You need to pass as parameters the AD group and the labName you want to give access to.

You can find this script, as well all the other scripts, under the storage account “vlabresources”. To access the storage, use Microsoft Azure Storage Explorer (see section 5.3)

IMPORTANT NOTE: in order to execute the script for the shutdown on idle task that we will describe in the next chapters, you have to install the Azure Powershell modules on the machine.

With a Powershell run as administrator, run the following script in this order:

Install-Module azurerm

Install-Module azure

At the end you could get an error about a cmdlet already existing. You can ignore it.  
Then execute the following script:

Set-ExecutionPolicy -ExecutionPolicy Unrestricted

If you still get errors while executing Azure-specific cmdlet, such as Login-AzureRmAccount or Save-AzureRMProfile, try to install the latest Azure modules from Web Platform Installer:

<https://www.microsoft.com/web/downloads/platform.aspx>

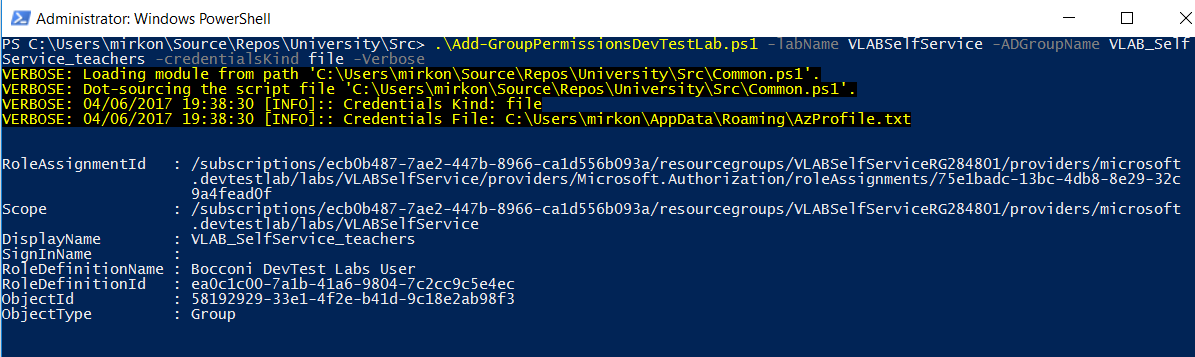
Before to use the script, you need to have the Azure context saved in a profile file on your local machine. To clarify, these are the instruction to create the profile file:

Login-AzureRmAccount

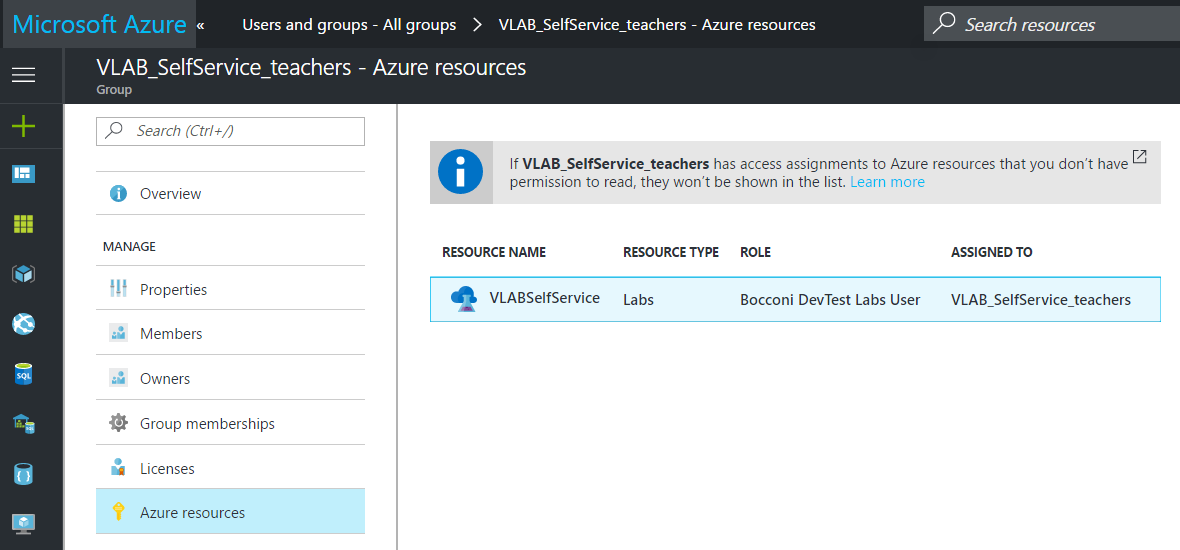
Set-AzureRmContext -SubscriptionId "XXXXX-XXXX-XXXX"

Save-AzureRMProfile -Path "$env:APPDATA\AzProfile.txt"

Then, you can launch the script. Here is an example of an execution:



This is the result from the Azure Portal from the AD group blade:



If you need to remove that permission to a group, you can run the “**Remove-GroupPermissionsDevTestLab.ps1**”, which performs the opposite action of the previous script.

1. VM scheduling automation

Along with the DevTest labs, an important Azure service used for automating the VM management script is Azure Automation. This service provides the ability to run Powershell scripts in the cloud in a secure and scalable environment. It also gives the admin the ability to run Powershell jobs either on demand or on specific schedules. This last feature is used to schedule the scripts for the VM management we provided to the universities, from the creation to the deallocation and destruction of the lab virtual machines.

* 1. Create an Automation Account

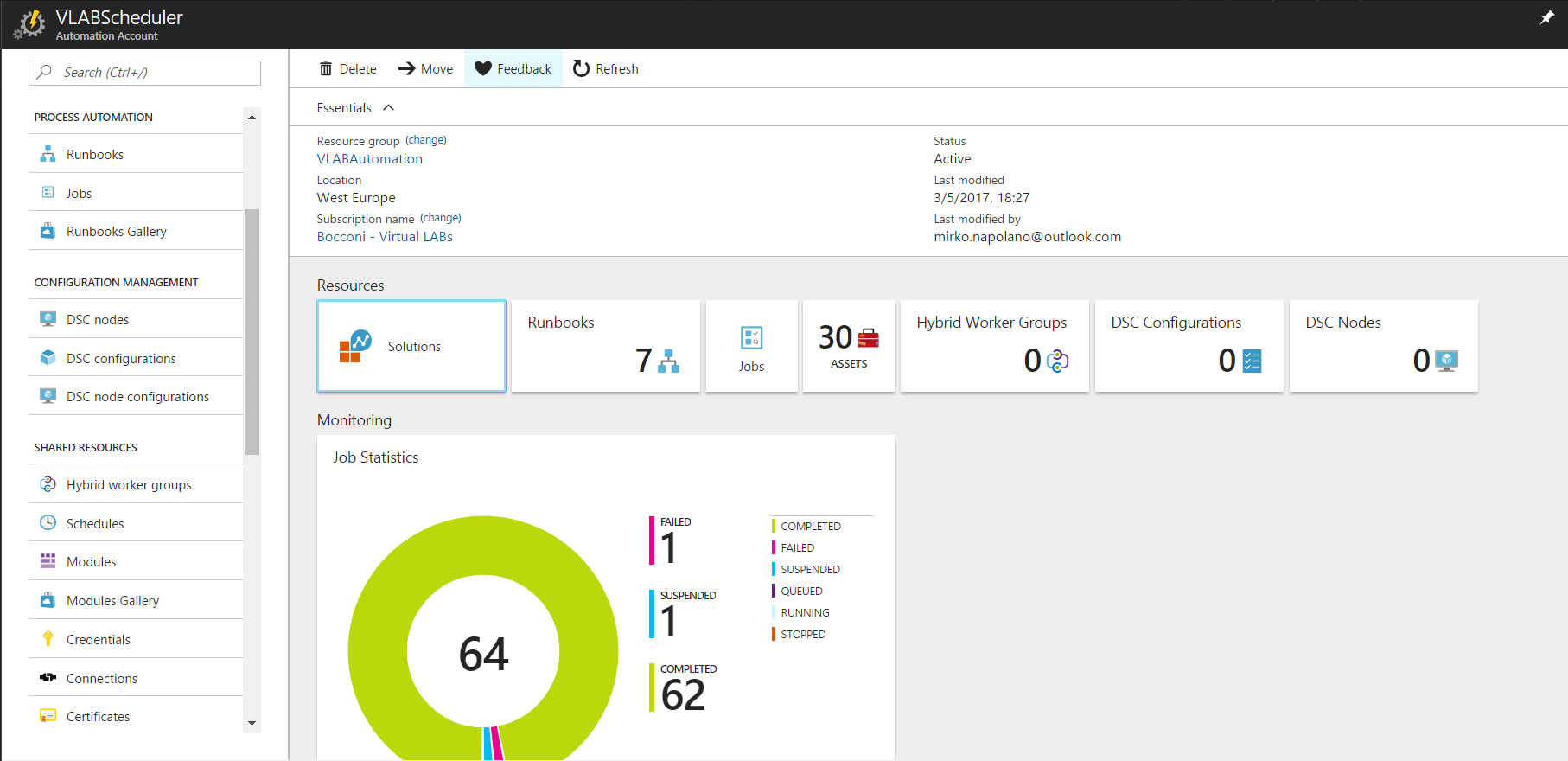
To create the Automation Account, you need to click on the “+” button, then choose the Automation resource and Create.

Microsoft Azure 
Everything 
Everything 
Y Filter 
Automation Account 
Results 
NAME 
Auto mat ion 
Automation 
Automation Hybrid Worker 
Runbook Studio for Azure Automation 
Logic App 
Start/Stop VMS during off-hours [Preview) 
Barracuda Web Application Firewall (WAF) 
Azure AD Identity Protection 
Barracuda Web Application Firewall (WAF) 
_ PAYG 
- 3YOL 
PUBLISHER 
Microsoft 
Microsoft 
Kelverion 
Microsoft 
Microsoft 
Barracuda Networks, Inc. 
Microsoft 
Barracuda Networks, Inc. 
Barracuda Networks, Inc. 
ODC Business Solutions 
Barracuda Networks, Inc. 
Aqua Security 
Microsoft 
Stop Azure Classic VMS 
M nft 
CATEGORY 
Developer tools 
Developer tools 
Compute 
Web Mobile 
Data * Analytics 
Compute 
Security + Identity 
Compute 
Compute 
Compute 
Compute 
Compute 
Monitoring Manage... 
Search resources 
Automation 
M o nft 
Create an Automation Account 
An Automation Account is a container for your Azure Automation resources. It provides a way to 
separate your environments or further organize your Automation workflows and resources. 
Process automation that simplifies cloud management 
Azure Automation allows you to automate the creation, deploynnent, monitoring, and maintenance 
of resources in your Azure environment and across external systems. Azure uses a highly scalable 
and reliable workflow execution engine to simplify cloud management. Orchestrate time-consuming 
and frequently repeated tasks across Azure and third-party systems. 
Integrate into the systems you depend on 
With Automation, you can connect into any system that exposes an API cn.'er typical Internet 
protocols. Azure Automation includes integration into many Azure services, including: 
Web Sites (management) 
Cloud Services (management) 
Virtual Machines (management and WinRM support) 
Storage (management) 
SQL Server (management and SQL support) 
Need your workflows to integrate into another service? Extend Azure Automation to third-party 
solutions simply by importing an existing PowerSheII module or writing your crun in or Windows 
PowerSheII. 
0 
Barracuda Web Application Firewall Solution (PAYG) 
SmartMessage Autoflow 
Barracuda Web Application Firewall Solution (BYOL) 
Aqua Container Security Platform 
Change Tracking 
Related to your search 
Hello World 
M nft 
PUBLISHER 
USEFUL LINKS 
Create 
Microsoft 
re Automation Documentation 
Find runbooks on Script Center 
Pricing Details 

Fill the fields with a name, choose a subscription, a new or existing Resource group and the location; leave the last field as default.

Microsoft Azure 
Everything > Automation 
Add Automation Account 
* Name O 
UniversityHogwarts 
* Subscription 
SJ Innovation - Prod Intemo 2015 
* Resource group O 
@ Create new 
C) use existing 
UniversityHogwarts 
* Location 
West Europe 
* Create Azure Run As account 
The Run As account feature will 
create a Run As account and a 
Classic Run As account.CIick here to 
learn more about Run As accounts. 
0 
Pin to dashboard 
Create 
Add Automation Account 

Here is the dashboard of an Azure automation account example:



In Azure Automation, each Powershell script is encapsulated in a process called Runbook. From a functional point of view, a runbook is a Powershell script that uses the Azure Automation account as environment to store external Powershell modules as well as certificates and generic variables (settings).

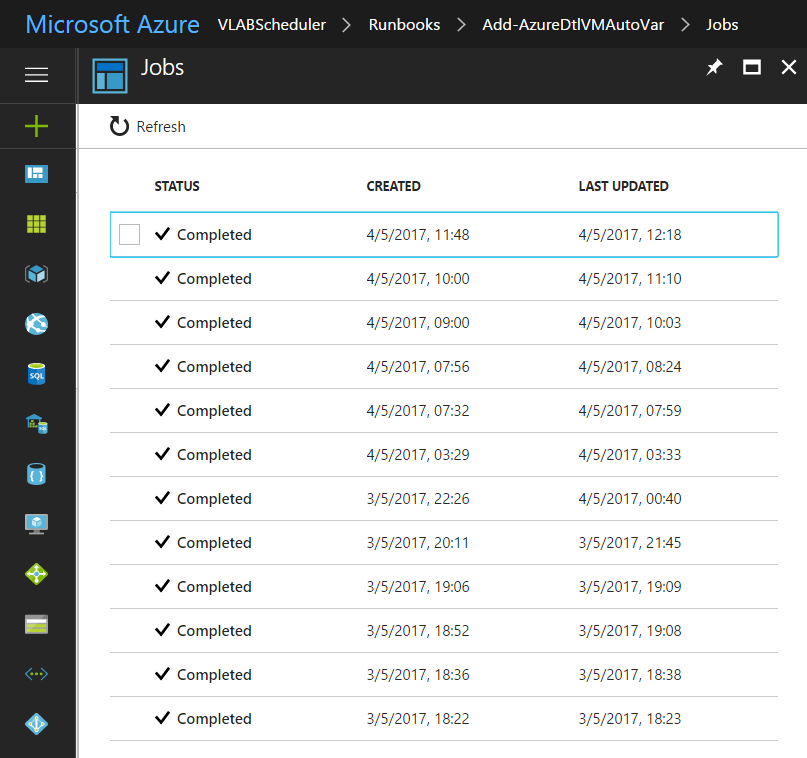
Automatically it creates some default scripts, which are useless for our purposes, therefore you can delete them all. You can find them inside the Runbooks tab.

Microsoft Azure 
U niversityHogwarts 
Runbooks 
Add a runbook 
Browse gallery 
bearch runbooks.„ 
NAME 
AzureAutomationTutoriaI 
AzureAutomationTutoriaIScript 
AzureCIassicAutomationTutoriaI 
AzureCIassicAutomationTutoria IS. 
0 
Runbooks 
Refresh 
AUTHORING STATUS 
Published 
Published 
Published 
Published 
LAST MODIFIED 
6/1/2017, 10:17 AM 
6/1/2017, 10:17 AM 
6/1/2017, 10:17 AM 
6/1/2017, 10:17 AM 
TAGS 

You must open them one by one and select Delete.

Microsoft Azure 
U niversityHogwarts > 
Schedule 
Runbooks 
Start 
AzureAutomationTutorial 
View 
Search resources 
c.cauterucci@softjam.it 
0 
AzureAutomationTutorial 
Search (Ctrl +1) 
Overview 
Activity og 
Tags 
Diagnose and solve problems 
RESOURCES 
CJ Jobs 
Q) Schedules 
Webhooks 
RIJN800K SETTINGS 
Properties 
Description 
Logging and tracing 
SETTINGS 
Locks 
Automation script 
suppoRT + TROUBLESHOOTING 
New support request 
Webhook 
Delete 
Export 
Refresh 
Delete Runbook 
Are you sure that you want to delete the runbook 'AzureAutomationTutorial'? 
scri Ion name 
S] Innovation - Prod Interno 2015 
Details 
Schedules 
Jobs 
(9 
Web hooks 

Finally, from the Jobs view you can see the results and logs of each run:



* 1. Import a runbook

There are two ways to import a runbook, either sync with a GitHub repo (through Source Control), or, as we will describe, pointing the file.

To use a GitHub repository you must be a contributor; instead, to import them, inside the Runbooks tab, click Add a runbook, choose the runbook type and then choose the file.

Microsoft Azure 
U niversityHogwarts 
OSDisk (C) 
Runbooks 
Refresh 
Add Runbook 
Import 
LAST MODIFIED 
Search resources 
Runbooks 
Add a runbook 
Search runbooks. 
NAME 
No runbooks found. 
X 
Browse gallery 
AUTHORING STATUS 
TAGS 
Open 
Organize • 
Add Runbook 
Quick Create 
Create a new runbook 
Import 
Import an existing runbook 
Search Src 
ThisPC 
New folder 
Name 
Documents 
University 
S rc 
0 
Date modified 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 1026 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
01/06/2017 10:26 
Type 
File folder 
File folder 
File folder 
File folder 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Windows PowerS... 
Size 
* Quick access 
Desktop 
Downloads 
Documents 
[e] Pictures 
iCIoud Drive 
OSDisk (C) 
Dropbox 
OneDrive - Persor 
OneDrive 
- SoftJar 
This pc 
ObsoleteScripts 
Roles 
Shutdown scripts 
SimplifiedJSPortaI 
Add-AzureDtIVM.psI 
Add-AzureDtIVMAutoVar.psI 
Add GroupPermissionsDevTestLab.psI 
Common.psl 
DeallocateStoppedVM.psI 
Manage-AzureDtIFixedPooI.psI 
Remove-AzureDtILabVMs.psI 
Remove-AzureDtIVM.psI 
Remove- GroupPermissionsDevTestLab.psI 
Test-AzureDtIVMs.psI 
File name: Common.psl 
10 G 
10 KB 
File personalizzati 
Import 
Select a file smaller than I Ma to import. 
* Runbook file O 
Select a file 
* Runbooktype O 
PowerSheII 
* Name 
Description 
Create 

In the end, you should find all the scripts:

M i crosoft Azure UniversityHogwarts 
Runbooks 
Add a runbook 
Browse gallery 
Search runbooks. 
NAME 
Add-AzureDtIVM 
Add-AzureDtIVMAutoVar 
Common 
DeallocateStoppedVM 
Remove-AzureDtILabVMs 
Remove-AzureDtIVM 
Test-AzureDtIVMs 
0 
Runbooks 
Refresh 
AUTHORING STATUS 
Search resources 
LAST MODIFIED 
6/1/2017, 10:28 AM 
6/1/2017, 10:29 AM 
6/1/2017, 10:28 AM 
6/1/2017, 10:29 AM 
6/1/2017, 10:30 AM 
6/1/2017, 10:29 AM 
6/1/2017, 10:30 AM 
TAGS 

* 1. Authentication method

Usually, if you have a script running from your local machine that accesses Azure resources, you don’t need to deeply modify the script to let it work as runbook.

For sure, one of the part that you need to change is the authentication method. Since you cannot perform login interactively with Runbooks, you must authenticate to Azure using a service principal. When you create an Automation account, you automatically get two “Run as Account”, one for the Azure Resource group model and one for the classic Azure Service resource model. Both allow you to connect to Azure Automation by using the service principal.

In order to log on to Azure from the scripts, we used the following code for managing the authentication to the target Azure subscription:

$connectionName = "AzureRunAsConnection"

$servicePrincipalConnection = Get-AutomationConnection -Name $connectionName

Add-AzureRmAccount `

-ServicePrincipal `

-TenantId $servicePrincipalConnection.TenantId `

-ApplicationId $servicePrincipalConnection.ApplicationId `

-CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint

#Set-AzureRmContext -SubscriptionId $servicePrincipalConnection.SubscriptionID

Select-AzureRmSubscription -SubscriptionId $servicePrincipalConnection.SubscriptionID | Write-Verbose

# Save profile so it can be used later and set credentialsKind to "File"

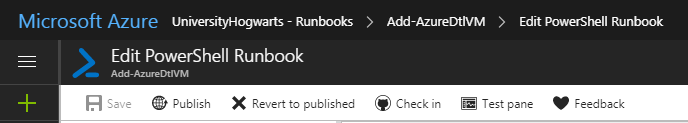
$global:profilePath = (Join-Path $env:TEMP (New-guid).Guid)

Save-AzureRmProfile -Path $global:profilePath | Write-Verbose

The following link from documentation gives more details about the authentication:

<https://docs.microsoft.com/en-us/azure/automation/automation-sec-configure-azure-runas-account>

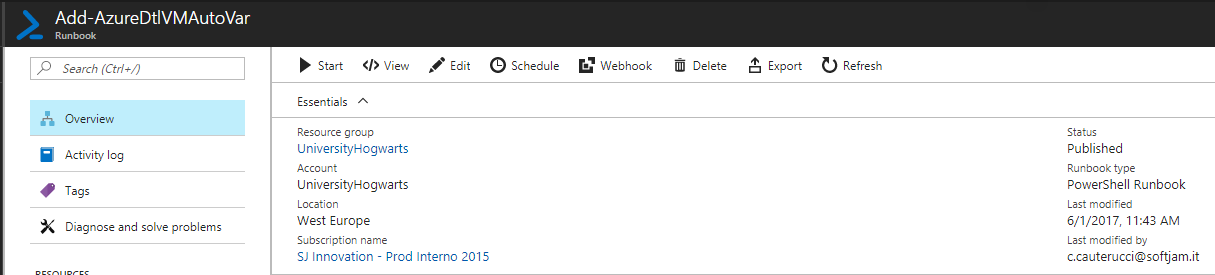
You can view the code of a runbook by clicking on the button “View” inside the runbook blade. As a note, if for some reasons, you want to modify a script in the runbook, you have to click on “Edit”. However, all the changes are effective only when you publish the runbook by clicking on the related button:



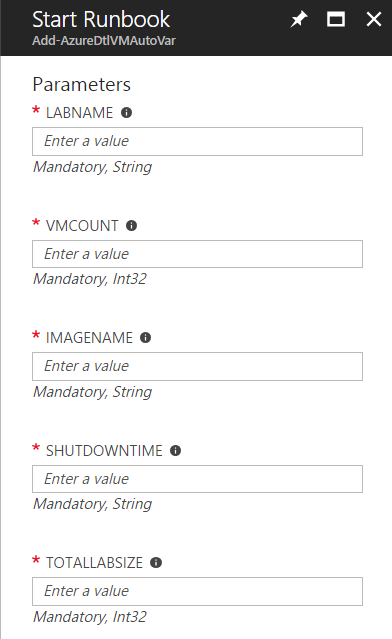
Once you’ve uploaded every script, you must publish them all in order to work.

* 1. Execute a runbook on demand

Supposing that you have already created or imported a runbook as described in the previous section, let’s say that you want to run your script on-demand. The easiest way to run a script on-demand in Azure Automation is from the Azure Portal. Just open the runbook you want to run and click on Start:



Then, if the script has parameters, you will be prompted to insert the mandatory parameters and, if not needed, you can ignore the optional parameters, which will take the defaults. Here is an example:

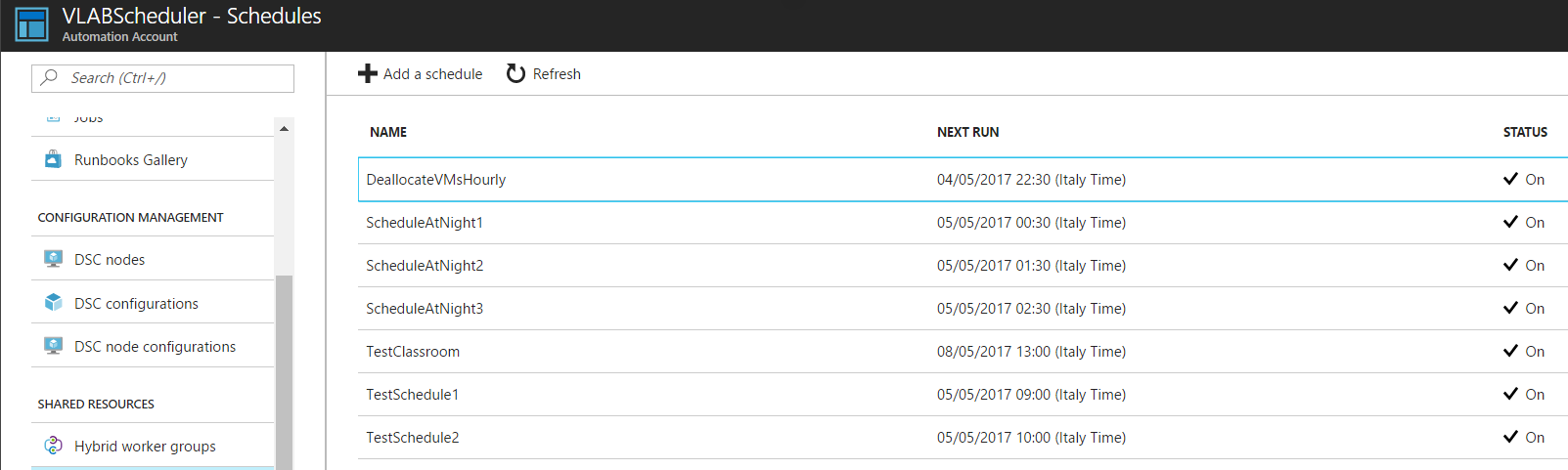


* 1. Schedule a runbook execution

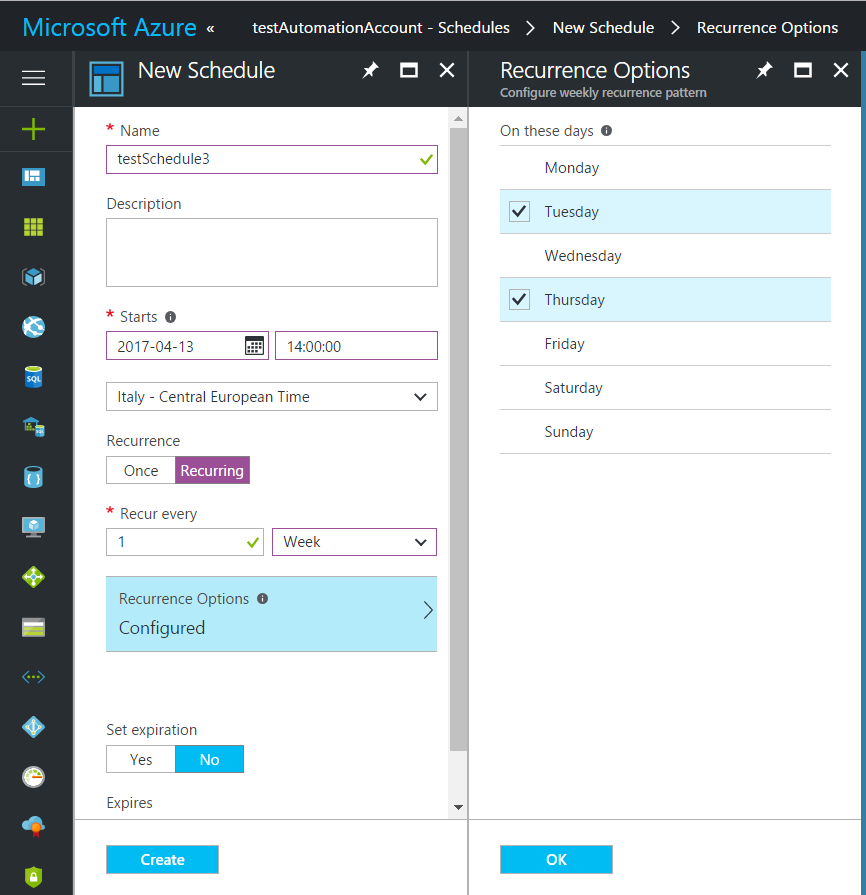
To schedule the execution of a Runbook in Azure Automation, you need to create a schedule and associate the runbook to it. The schedule is an asset of the Automation account, so that you can reuse the schedule definition for multiple runbooks.

NOTE: each script needs to be published before being associated to a schedule.

The first step is to create the schedule definition. You can do this from the Automation account dashboard by clicking on the tab “Schedule” in the left blade:

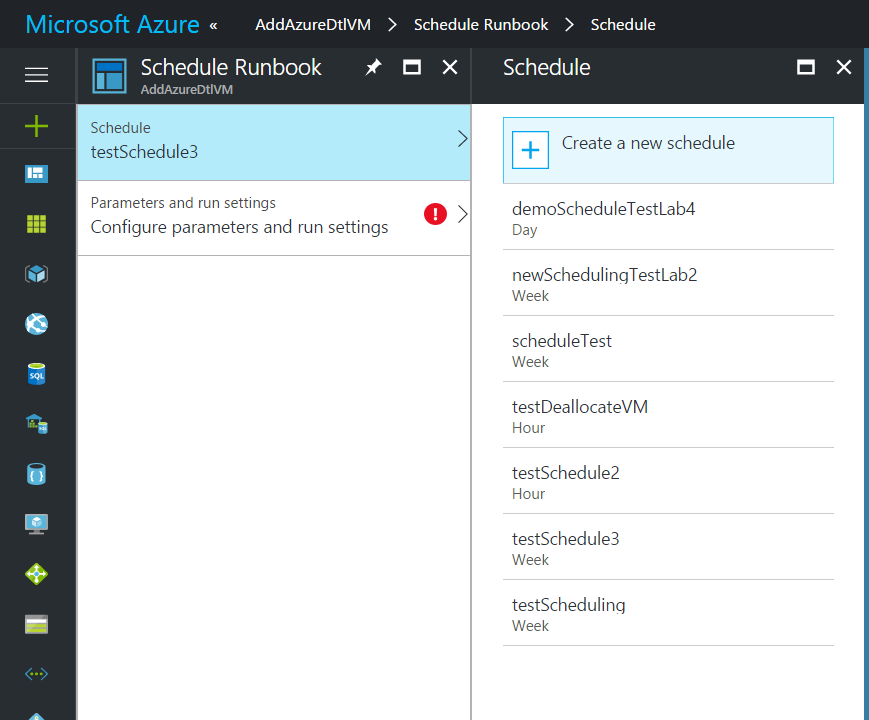


Then, you can configure your schedule. The most important part is the recurrence setting, which allows you to apply the same schedule many times. The following is an example:



The second step for scheduling a runbook is to link the schedule definition to the runbook, so that it can run with the set of input parameters you want scheduled as you defined before.

To do this, you need to open your runbook, then select “Schedule” and choose one of the schedule definition. Note that you can also create a new schedule at this point:



Then, as for the on-demand scenario, you must choose the script parameters.

NOTE: each schedule definition can be attached to one instance of the same script. So, if you need to change the parameters of a script scheduled, you need to unlink the script from the schedule, then reattach it with the new parameters.

If you want to check the status of the script, inside the logging and tracing tab you must set the logging to verbose.

Microsoft AZUre UniversityHogwarts - Runbooks > Add-AzureDtlVMAutoVar - Logging and tracing 
Add-AzureDtlVMAutoVar - Logging and tracing 
0 
Search (Ctrl 
Overview 
Activity og 
Tags 
Diagnose and solve problems 
RESOURCES 
CJ Jobs 
Q) Schedules 
Webhooks 
RIJN800K SETTINGS 
Properties 
Description 
Logging and tracing 
SETTINGS 
Locks 
Automation script 
suppoRT + TROUBLESHOOTING 
New support request 
R 
x 
Discard 
Logging 
Log verbose records 
Log progress records 
Activity-level tracing 
This configuration is available only for graphical runbooks. 
Trace level 
None 
Basic 
Detailed 

1. Access to the lab

In this phase, students will access to the labs by using a deep link. This URL refers to the dashboard of the specified lab inside the Azure Portal. So, in order to provide students the access for each lab, you need to have a link for each lab.

The pattern of the lab link is the following:

<https://portal.azure.com/#resource/subscriptions/{subscriptionId}/resourceGroups/{LabResourceGroup}/providers/Microsoft.DevTestLab/labs/{LabName}/overview>

where

* **{subscriptionId}** is the guid of the Azure Subscription (if you use only one subscription, you don’t need to change it)
* **{LabResourceGroup}** is the Resource Group name where the lab resides
* **{LabName}** is the name of your target lab

The lab resource group is automatically created together with the lab at the creation, and it does not change until the lab is destroyed. You can find it in the Azure Portal by looking at the blade of your lab:

Microsoft Azure Physics 
Physics 
Du Test Lab 
Search (Ctrl 
Overview 
Getting started 
My virtual machines 
Claimable virtual machines 
All virtual machines 
My data disks 
Formulas (reusable bases) 
My secrets 
SETTINGS 
Configuration and policies 
0 
Refresh 
Resource group 
PhysicsRG797792 
Status 
Ready 
Location 
West Europe 
My virtual machines 
NAME 
Nothing to display 
Claimable virtual machines 
Claim any 
Delete 
MSDN forum 
Feed back 
Search resources 
Subscription name 
SJ Innovation - Prod Interno 2015 
Subscription ID 
7e287d18-306f-4454-ge5d-2187098c3c1c 
NAME 
Virtual Image 
vm44705529001 
vm44705529000 
STATUS 
STATUS 
O Stopped (deallocated) 
O Stopped (deallocated) 
Stopping 
AUTO-START 
AUTO-START 
No 
No 
No 
AUTO-SHUTEH)WN 
AUTO-SHUTEH)WN 
Yes 
Yes 
Yes 
Windows Server 2016 - 
Univlmage 
Univlmage 
Nano Server 

* 1. List of scripts

The solution for the University DevTest lab leverages the use of some custom scripts scheduled as Runbook in Azure Automation. Specifically, the following are the scripts that need to be used for the solution with the list of parameters for the execution.

* + 1. Add-AzureDtlVmAutoVar: creation of VMs

This is the main script. It allows the creation of the VMs inside a specific lab. This script needs to be scheduled for the creation of the VMs of each lab. The following is the list of the parameters:

1. **Labname**: the name of the lab
2. **VMCount**: the number of VM instances (batch) to create for the target lab with this execution
3. **ImageName**: the name of the base image
4. **ShutDownTime**: the automatic shutdown time for the VM, in form of “HH:mm”
5. **TotalLabSize**: the desired total number of VMs in the lab

Note that the VMs are created when the script is launched/scheduled. As for the request of keeping the cost of VMs execution low, the machines are created and suddenly stopped.

IMPORTANT NOTE:

The VMCount indicates the number of VM to create with a single run of the script, while the TotalLabSize indicates the limit of VMs in the lab. We had to split these two parameters in order to manage a high demand of VM with a single runbook instance. As we will show later, this means that you have to run/schedule the script for adding the VM multiple times (e.g. if you need to have 300 VMs in your lab, you can schedule the script 3 times with VMCount=100 every hour)

Moreover, in order to have a workaround to possible issues in the runbook that can cause the automatic relaunch of the same script (and, so, the creation of more VMs), you have to set a higher limit when launching the script with VMs already in the lab. This is usually the case of the self-service lab, where you have old VMs that will be destroyed at 3am and you schedule the first script before 3 am (e.g. 00:30am). In that case, you have to set as TotalLabSize a value higher than the actual target, otherwise no VMs will be created (e.g. if you want to have 300 VMs, set 100 as VMCount and 600 as TotalLabSize of the first schedule, the same for the second schedule and 100 VMCount and 300 TotalLabSize as third schedule)

This runbook calls another script behind the scenes, which is **Add-AzureDtlVM**. This script use additional parameters, such as virtual network configuration and VM size, because it was created more generic. However, since some of them are configuration settings and don’t change for each scenario, you can avoid setting them. Such parameters are set as “**variables**” in the Azure Automation account and they are loaded into the script when it is run as runbook.

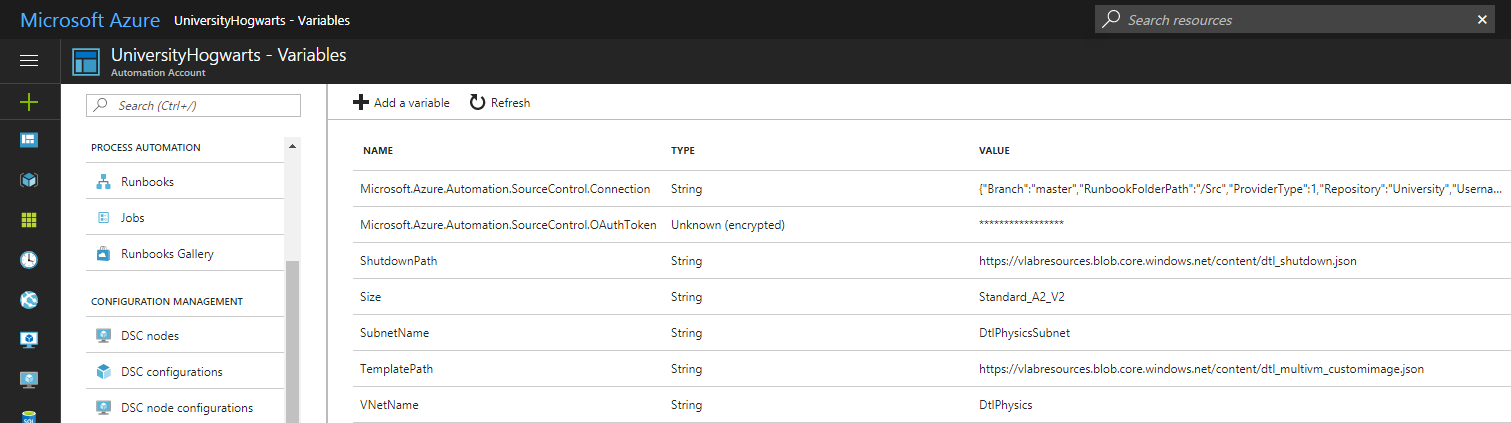
NOTE: Any Lab you will create it will share the same. If you want to customize every lab, i.e. every lesson you may not want to use this script, instead you can use the Add-AzureDtlVM.

The following is the list of parameters used by the creation script that are bound into Azure Automation variables:

1. **TemplatePath:** <https://vlabresources.blob.core.windows.net/content/dtl_multivm_customimage.json>
2. **ShutdownPath:** <https://vlabresources.blob.core.windows.net/content/dtl_shutdown.json>
3. **Size**: Standard\_A2\_v2
4. **VNetName**: DtlPhysics
5. **SubnetName:** DtlPhysicsSubnet

The json files can be anywhere (better inside a storage account).

You can access the variables from the Variables blade in the main menu of the Automation account:



You can access the VNet name here:

Microsoft Azure 
Physics > Configuration and policies - Virtual networks 
Configuration and policies - Virtual networks 
Search resources 
DuTest Lab 
Search (Ctrl 
COST TRACKI NG 
Cost trend 
@ Cost by resource 
POLICY SETTINGS 
Allowed virtual machine sizes 
Virtual machines per user 
Virtual machines per lab 
SCHEDULES 
to-shutdown 
Auto-start 
EXTERNAL RESOURCES 
Repositories 
Virtual netwn•rks 
0 
VIRTUAL MACHINE BASES 
Marketplace images 
Custom images 
Formulas (reusable bases) 
MANAGE 
Access control (IAM) 
Diagnose and solve problems 
Feedback 
NAME 
DtIPhysics 
RESOURCE GROUP 
STATUS 
Ready 
DESCRIPTION 

The Subnet is under the net: if you click the net you will find the subnet name.

If you don’t want to use the default net and subnet, you can use the name of your own existing net and subnet.

NOTE: if for some reasons you want to change one of these parameters, you must modify the related variables in the Automation account.

EXAMPLE: SELF-SERVICE SCENARIO

The following is an example that can be used to create 300 VM in the self-service lab with the automatic shutdown set at 03:00.

You must create 3 different schedules (for example one each hour at 00:30, 01:30 and 04:30). As we said before, you must set a “oversized” TotalLabSize parameter if you schedule the runbook when there are still VMs in the lab. In this case, if you schedule the script before 3am, you will still have the VMs of the previous day expiring at 3am. So, you must oversize the limit for the first two schedules and set the correct limit in the schedule after the expiration.

So, this can be a configuration:

* first schedule at 00:30 AM

1. **LabName**: VLABSelfService
2. **VMCount**: 100
3. **ImageName**: UBLabT-baseImageV2
4. **ShutDownTime**: 03:00
5. **TotalLabSize**: 600

* second schedule at 01:30 AM

1. **LabName**: VLABSelfService
2. **VMCount**: 100
3. **ImageName**: UBLabT-baseImageV2
4. **ShutDownTime**: 03:00
5. **TotalLabSize**: 600

* third schedule at 04:30 AM

1. **LabName**: VLABSelfService
2. **VMCount**: 100
3. **ImageName**: UBLabT-baseImageV2
4. **ShutDownTime**: 03:00
5. **TotalLabSize**: 300

In this way, at the end of the last script, no more than 300 VMs will be available in the lab.

NOTE: the script must run every day.

EXAMPLE: CLASSROOM SCENARIO

For the classroom scenario, the script for create the machine is the same. You need to schedule this script for each classroom lab.

Supposing to populate a lab for 150 VMs, you can follow the previous pattern to “split” the creation in batches.

In this case, the **ShutdownTime** parameter needs to be set after the end of the lessons (e.g. 1 hour after the end of the lessons, which must be known, of course).

Differently from the self-service lab, you can schedule the script later on during the day, in order not to overlap with the other scripts for self-service lab. For example, for a class starting at 2pm, you can schedule the creation during the morning.

This can be a possible configuration scheduled at 9am and 10am for a class starting at 2pm and finishing at 4pm:

* first schedule at 11:00 AM

1. **LabName**: Physics
2. **VMCount**: 75
3. **ImageName**: UBLabT-class
4. **ShutDownTime**: 17:00
5. **TotalLabSize**: 150
6. **ExpirationTime**:17:00
7. **DaysToExpiry**:1

* second schedule at 12:00 AM

1. **LabName**: Physics
2. **VMCount**: 75
3. **ImageName**: UBLabT-class
4. **ShutDownTime**: 17:00
5. **TotalLabSize**: 150
6. **ExpirationTime**:17:00
7. **DaysToExpiry**:1

* second schedule at 01:00 PM

1. **LabName**: Physics
2. **VMCount**: 150
3. **ImageName**: UBLabT-class
4. **ShutDownTime**: 17:00
5. **TotalLabSize**: 150
6. **ExpirationTime**:17:00
7. **DaysToExpiry**:1

NOTE: since you can only attach one schedule definition per each script instance, you need to create as many schedule definition as the number of the labs (e.g. one definition for the self-service lab, one definition for the VLABStata, one definition for AdvancedExcel and so on)

* + 1. DeallocateStoppedVM

This script checks the status of the VMs inside the target lab. It verifies if there are VMs in “Stopped” status, which means a VM that has been shut down by the user within the OS. If so, it changes the status in “Stopped (deallocated)” by deallocating the resources and stopping the VM.

It uses only one parameter, which is the “LabName”.

This script is scheduled every hour for the **self-service lab**, so that, if a student shuts down the VM voluntarily or if the VM goes in shutdown automatically for the “ShutdownOnIdle” task, the script will deallocate the VM.

* + 1. Manage-AzureDtlFixedPool

This script checks how many machines are inside a lab, tipically in the self-service lab and guaratees the presence of an appropriate number of Virtual machines inside the Lab depending on the PoolSize you have specified inside the lab tag.

* + 1. Test-AzureDtlVM

This script checks the number of VMs in a lab and verifies whether the number is expected or not. You can start it on demand or schedule after the creation of VMs in a lab. In this way, you can have a quick view about the status of VMs in the lab and, accordingly, add more VMs with the creation script.

Otherwise, you can add a schedule to run the script at the time the lesson should begin, to check how many VMs are actually available:

* Self-service scenario, schedule at 05:30 AM

1. **LabName**: VLABSelfService
2. **LabSize**: 300

* Class scenario, schedule at 02:00 PM

1. **LabName**: Physics
2. **LabSize**: 150

Microsoft Azure 
U niversityHogwarts 
Schedule Runbook 
TEt-An Dt IV M s 
Schedule 
t4 
Parameters and run settings 
Runbooks > 
Test-AzureDtlVMs > 
Parameters 
TEt- Dt IV M s 
Parameters 
* LABNAM 
Physics 
Mandatory, String 
PROFILEPATH O 
Default wit/ be used 
Schedule Runbook > 
Parameters 
o 
Configure parameters and run settings 
0 
Optional String, Default: 
"Senv•ßPPDAAAzProfi1e.txt" 
* LABSIZE O 
Mandatory, String 
VMDELTA O 
Default wit/ be used 
Optional, Double, Default: O. 
Run Settings 
Run on Azure O 

In order to let this script send you an email when something goes wrong, you have to creat a Log Anlytics workspace inside Operation Management Suite and then connect the automation account to Log Analytics inside Operation Management Suite. At this link, you can find how to set up a Log Analytics environment:

<https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-get-started>

Once you’ve done this, you have to connect the Automation Account to your Log Analytics environment and you can do it only via PowerShell.

You need to run the following lines of code:

Login-AzureRmAccount, which allows you to connect to Azure with your credentials.

If you need change the current subscription you’re working on run the following commands to get the list of subscriptions and then select the one you need.

Get-AzureRmSubscription,

Select-AzureRmSubscription -SubscriptionId {SubscriptionID}

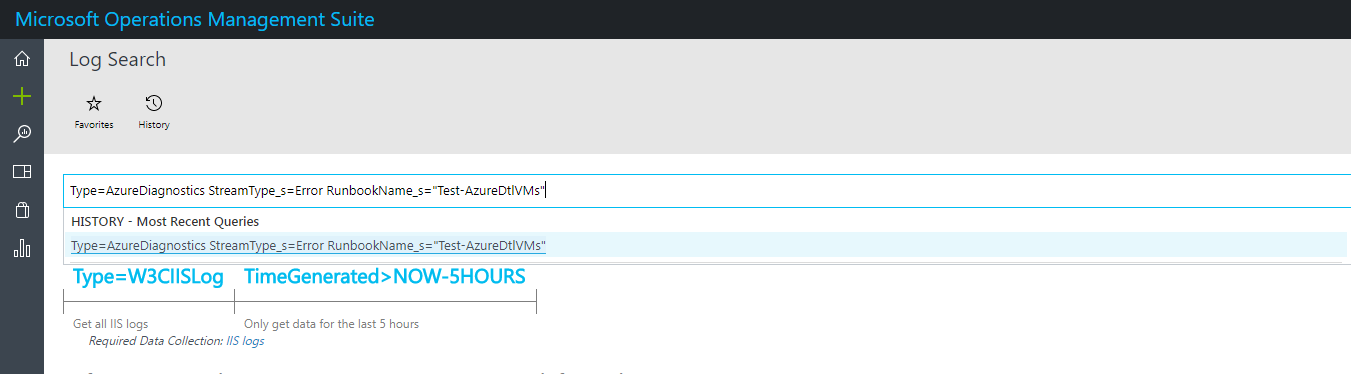
Finally, run the last line of code to link your automation account to Log Analytics and tracking the logs.

Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/{SubscriptionID}/resourceGroups/{AutomationAccountName}/providers/Microsoft.Automation/automationAccounts/{AutomationAccountName}  -WorkspaceId /subscriptions/{ SubscriptionID}/resourcegroups/{ResourceGroupName}/providers/microsoft.operationalinsights/workspaces/{LogAnalyticsName}  -Enabled $true

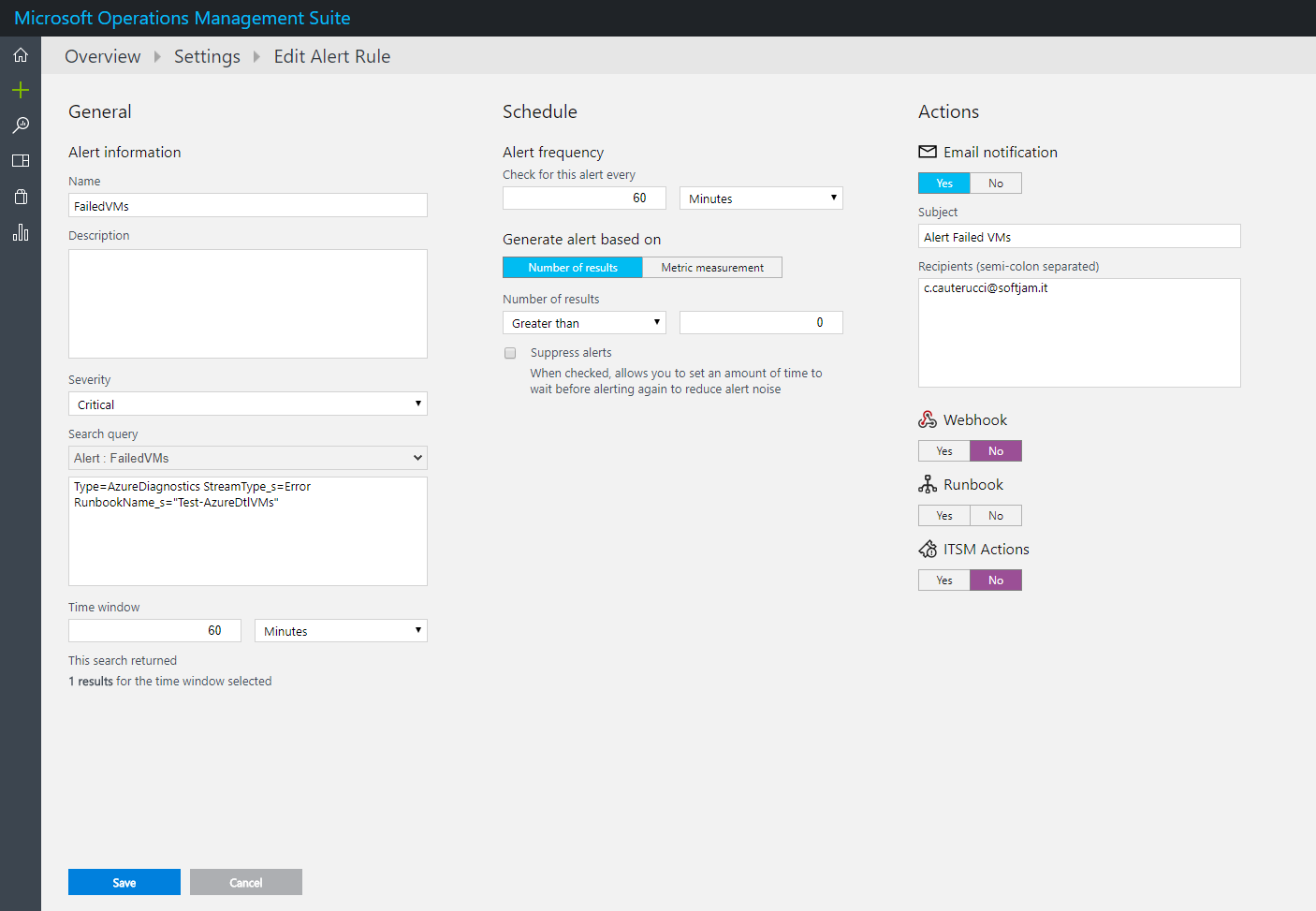
Once you’ve done this, go to the OMS portal by login in at this site: <https://login.mms.microsoft.com/signin.aspx?ref=ms_mms> or look for your Log analytics inside Azure and click OMS Portal button.

Inside Operation Management Suite select “Log Search” and inside the Query Tab type the following command:

Type=AzureDiagnostics StreamType\_s=Error RunbookName\_s="Test-AzureDtlVMs"



Once the query has run, click on alert and set the parameters of the email as prompted here:



Glossary

Shutdown: with the term “shutdown” we refer to the cessation of the operating system running on the VM. This will not free the allocated resources assigned by Azure to the Virtual Machine. In the Azure Portal, you see the status as “Stopped”.

Deallocation: By stopping (deallocating) a VM, you not only stop the VM’s Operating System, you also free up the hardware and network resources Azure previously provisioned for it. In the Azure Portal, you see the status as “Stopped (deallocated)”.

Claim a VM: claiming a VM is an action that a user performs to become the owner of a lab VM that was already created and not assigned yet. As soon as a user become the owner of a VM he can start/stop it and connect to it via RDP