Azure Automation Shutdown VM’s

Tip: Change Tag to define Schedule like 10pm -> 6am, Saturday and Sunday

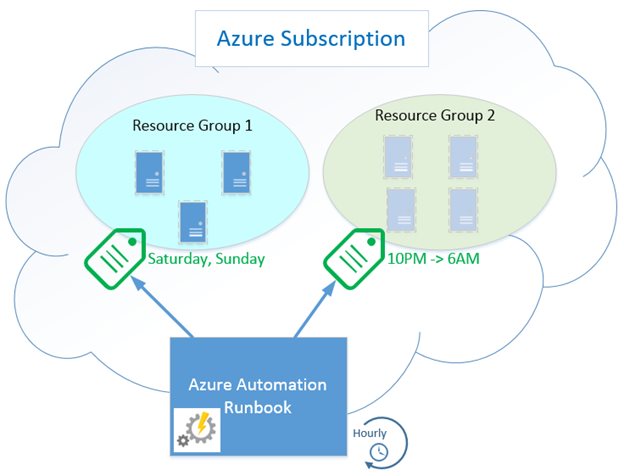


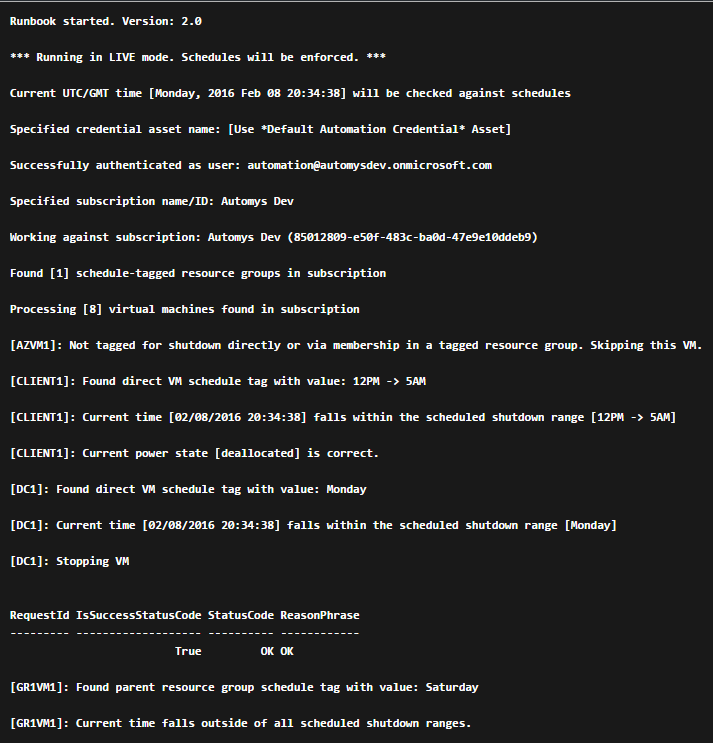
https://automys.com/library/asset/scheduled-virtual-machine-shutdown-startup-microsoft-azure

What It Does

This runbook automates scheduled startup and shutdown of [Azure virtual machines](http://azure.microsoft.com/en-us/services/virtual-machines/). You can implement multiple granular power schedules for your virtual machines using simple tag metadata in the Azure portal or through PowerShell. For example, you could tag a single VM or group of VMs to be shut down between the hours of 10:00 PM and 6:00 AM, all day on Saturdays and Sundays, and during specific days of the year, like December 25.

The runbook is intended to run on a schedule in an [Azure Automation](http://azure.microsoft.com/en-us/services/automation/) account, with a configured subscription and associated access credentials. For example, it can run once every hour, checking all the schedule [tags](http://azure.microsoft.com/en-us/documentation/articles/resource-group-using-tags/) it finds on your virtual machines or [resource groups](http://azure.microsoft.com/en-us/documentation/articles/resource-group-portal/). If the current time falls within a shutdown period you’ve defined, the runbook will stop the VM if it is running, preventing any compute charges. If the current time falls outside of any tagged shutdown period, this means the VM should be running, so the runbook starts any such VM that is stopped.

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150531191605165.jpg)Using tags to control power schedules

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-160208204925517.png)Scheduled startup/shutdown runbook output example

Once the runbook is in place and scheduled, the only configuration required can be done through simple tagging of resources, and the runbook will implement whatever power schedules it finds during its next scheduled run. Think of this as a quick and basic power management scheduling solution for your Azure virtual machines.

Sound enticing? Read on for more details.

Why Use This?

Money! The largest share of Azure subscription costs when using Virtual Machines (IaaS) is the compute time: how many hours the VMs are running per month. If you have VMs that can be stopped during certain time periods, you can reduce the bill by turning them off (and “[deallocating](http://blogs.technet.com/b/keithmayer/archive/2013/06/19/windows-azure-virtual-machines-there-s-more-than-1-way-to-shutdown-a-vm.aspx)” them).

Unfortunately, Microsoft doesn’t include any tools to directly manage a schedule like this. That’s what this runbook helps achieve without 3rd party management tools or chaining a junior admin to the keyboard for 6AM wakeup call.

How it Works

Let’s get into more detail about how the runbook makes the magic happen.

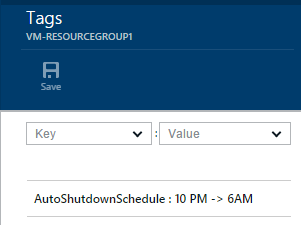
Tag-based Power Schedules

If our goal is to manage the times that our virtual machines are shut down and powered on, we need a way to define this schedule. For example, perhaps we want to shut down our VMs after close of business and have them start up before people arrive in the office in the morning. But we also might want them shut down all weekend, not just at night. And what about holidays? Clearly, we also need an approach that allows some flexibility to get granular with scheduling.

The first thing we might think to use is a runbook schedule, which Azure already provides out of the box. In essence, we can configure a runbook to run hourly or daily and do a task like shutting down VMs. But as just discussed, what if you have multiple schedules for different VMs? And that’s for shutting down – what about starting them again? Do you use multiple runbooks following multiple schedules? This starts to get confusing and awkward to manage. Unfortunately most of the existing examples I came across followed this kind of approach.

When you think about it, the power schedule applies to the resource, not to the runbook. The alternative approach used by the runbook solution here described is to tag a VM with a schedule, so that the mechanism used to stop and start VMs is transparent – it just happens when you declare that it should. If you’re especially nerdy when it comes to programming, you might recognize this as a [declarative rather than imperative](http://latentflip.com/imperative-vs-declarative/) approach. It doesn’t use PowerShell [Desired State Configuration](https://technet.microsoft.com/en-us/library/dn249912.aspx) (yet?), but is in the same spirit.

So what does it look like? We simply apply a tag to a virtual machine or an Azure resource group that contains VMs. This tag is a simple string that describes the times the VM should be shut down.

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024650991.png)Use tags to define shutdown schedules

Tag Content

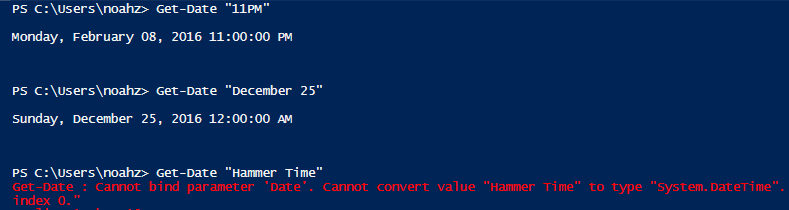
The runbook looks for a tag named “AutoShutdownSchedule” assigned to a virtual machine or resource group containing VMs. The value of this tag is one or more schedule entries, or time ranges, defining when VMs should be shut down. By implication, any times not defined in the shutdown schedule are times the VM should be online. So, each time the runbook checks the current time against the schedule, it makes sure the VM is powered on or off accordingly.

There are two kinds of entries:

* Time range: two times of day or absolute dates separated by ‘->’ (dash greater-than). Use this to define a period of time when VMs should be shut down.
* Day of week / Date: Interpreted as a full day that VMs should be shut down.

**Get to Know DateTime**

All times must be strings that can be successfully parsed as “[DateTime](https://technet.microsoft.com/en-us/library/ff730960.aspx)” values. In other words, PowerShell must be able to look at the text value you provide and say “OK, I know how to interpret that as a specific date/time”. There is a surprising amount of flexibility allowed, and the easiest way to verify your choice is to open a PowerShell prompt and try the command *Get-Date “<time text>”*, and see what happens. If PowerShell spits out a formatted timestamp, that’s good. If it complains that it doesn’t know what you mean, try writing the time differently.

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-160208204925689.png)Testing schedule text. Somehow PowerShell doesn't know about Hammer Time.

**UTC Timezone**

To simplify the code, **the runbook expects all times to be in**[**UTC**](http://en.wikipedia.org/wiki/Coordinated_Universal_Time)**, not local time**. This means that before you define a schedule, first [convert](http://www.timeanddate.com/worldclock/converter.html) the times to UTC / GMT.

Schedule Tag Examples

The easiest way to write the schedule is to say it first in words as a list of times the VM should be shut down, then translate that to the string equivalent. Remember, any time period not defined as a shutdown time is online time, so the runbook will start the VMs accordingly. Let’s look at some examples:

|  |  |
| --- | --- |
| **Description** | **Tag value** |
| Shut down from 10PM to 6 AM UTC every day | 10pm -> 6am |
| Shut down from 10PM to 6 AM UTC every day (different format, same result as above) | 22:00 -> 06:00 |
| Shut down from 8PM to 12AM and from 2AM to 7AM UTC every day (bringing online from 12-2AM for maintenance in between) | 8PM -> 12AM, 2AM -> 7AM |
| Shut down all day Saturday and Sunday (midnight to midnight) | Saturday, Sunday |
| Shut down from 2AM to 7AM UTC every day and all day on weekends | 2:00 -> 7:00, Saturday, Sunday |
| Shut down on Christmas Day and New Year’s Day | December 25, January 1 |
| Shut down from 2AM to 7AM UTC every day, and all day on weekends, and on Christmas Day | 2:00 -> 7:00, Saturday, Sunday, December 25 |
| Shut down always – I don’t want this VM online, ever | 0:00 -> 23:59:59 |

What the Runbook Does

The runbook **Assert-AutoShutdownSchedule** is an [Azure Automation runbook](http://azure.microsoft.com/en-us/documentation/articles/automation-runbook-concepts/). It can be run once at a time manually, but is intended to be configured to run on a schedule, e.g. once per hour.

The runbook expects two parameters: the name of the Azure subscription that contains the VMs, and the name of an Azure Automation credential asset with stored username and password for the account to use when connecting to that subscription. If not specifically configured, the runbook will try by default to find a credential asset named “Default Automation Credential” and a variable asset named “Default Azure Subscription”. Setting these up is discussed in more detail below. There is a third parameter called "Simulate" which, if True, tells the runbook to only evaluate schedules but not enforce them. This is discussed further below.

Once successfully authenticated to the target subscription, the runbook looks for any VM or resource group that has a tag named “AutoShutdownSchedule”. Any resource groups without this specific tag are ignored. For each tagged resource found, we next look at the tag values to see what the schedule entries are. Each is inspected and compared with the current time. Then, one of several decisions is made:

* If the current time is outside of the defined schedules, the runbook concludes that this is “online time” and starts any directly or indirectly tagged VM that is currently powered off.
* If the current time matches any of the schedules, the runbook concludes that this is “shutdown time” and stops any directly or indirectly tagged VM that is currently powered on.
* If any of the defined schedules can’t be parsed (PowerShell doesn’t understand “beer thirty”), it will ignore that and treat whatever was intended as online time. Therefore**, the default failsafe behavior is to keep VMs online** or start them, not shut them down.

Runbook Logs

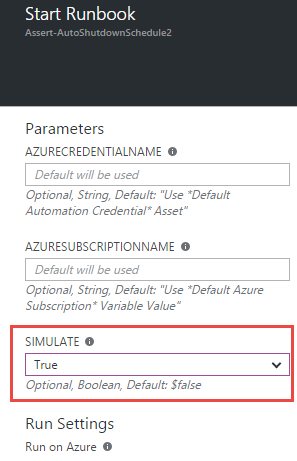
Various output messages are recorded by the runbook every time it runs, indicating what actions were taken and whether any errors occurred in processing tags or accessing the subscription. These logs can be found in the output of each job.

Performance

Startup and shutdown are done sequentially, one VM at a time. This helps to avoid problems with VMs that belong to the same cloud service, which allow only one member VM at a time to power on or off. Starting and stopping can take a minute or two per VM, so if you have a large environment, it could be worthwhile to customize the runbook to perform parallel actions where possible to minimize job duration.

Testing

To test the runbook without actually starting or stopping your VMs, you can use the "Simulate" option. If true, the schedules will be evaluated but no power actions will be taken. You can then see whether everything would have worked as you expect before setting up the runbook to run live (runbook runs live by default).

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-160208205152698.png)Using "Simulate" option to safely test

Azure Modules

As of version 2.0 of the runbook, there are no other requirements for modules other than what is included in your Azure Automation account by default. This includes the "Azure" and "AzureRM.Resources" modules. If you don't know what modules are, don't worry. If you have removed either of these or they aren't present, make sure to address that.

Setting it up in Azure

Now we’ll go through the steps to get this working in your subscription. It will be beer thirty before you know it.

Prerequisites

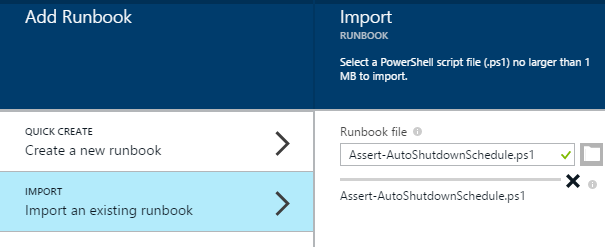
This is an Azure Automation runbook, and as such you’ll need the following to use it:

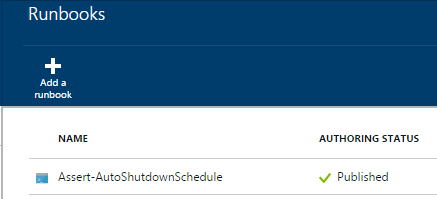
* Microsoft Azure [subscription](http://azure.microsoft.com/) (including trial subscriptions)
* Azure Automation account created in subscription ([instructions](https://azure.microsoft.com/en-us/documentation/articles/automation-configuring/))
* Runbook file downloaded from this page or imported from runbook gallery

Import Runbook

The runbook is contained in the file “Assert-AutoShutdownSchedule.ps1” within the download. You can import this into your Automation Account like so:

1. Open subscription in [https://portal.azure.com](https://portal.azure.com/)
2. Open the Automation Account which will contain the runbook
3. Open the **Runbooks** view from the Resources section
4. Click **Add a runbook** from the top menu
5. Select **Import an existing runbook**
6. Click **Create** to upload
7. Confirm “Assert-AutoShutdownSchedule” now appears in the runbooks list
8. Open the runbook from the list
9. Click **Edit** from the top menu
10. Click **Publish** from the top menu and confirm
11. Confirm the runbook now shows a status of **Published**

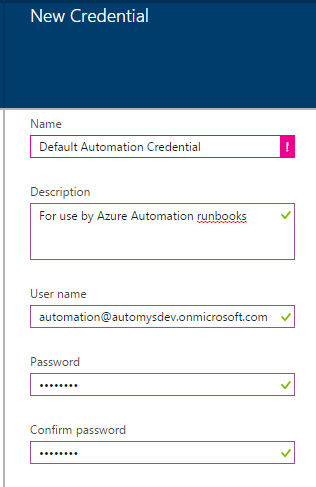
[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024651648.png)Import runbook from downloaded ps1 file

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024651726.png)

Create Credential Asset

When the runbook executes, it accesses your subscription with credentials you configure. By default, it looks for a credential named “Default Automation Credential”. This is for a user you create in your subscription’s Azure Active Directory which is granted permissions to manage subscription resources, e.g. as a co-administrator. The steps:

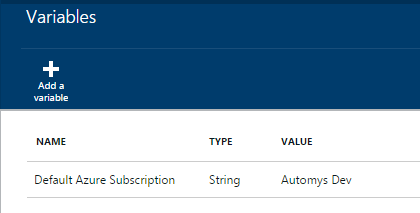
1. [Create an Azure Active Directory user](http://azure.microsoft.com/blog/2014/08/27/azure-automation-authenticating-to-azure-using-azure-active-directory/) for runbook use if you haven’t already. This account will be the "service account" for the runbook and**must be a co-administrator** in the target subscription.
2. Open subscription in [https://portal.azure.com](https://portal.azure.com/)
3. Open the Automation Account which will contain the runbook
4. Open the **Assets** view from the resources section
5. Open the **Credentials** view
6. Click **Add a credential** from the top menu
7. Enter details for the new credential. Recommended to use name “**Default Automation Credential**”.
8. Click **Create**

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024651835.png)

Create Variable for Subscription Name

The runbook also needs to know which subscription to connect to when it runs. In theory, a runbook can connect to any subscription, so we must specify one in particular. This is easily done by setting up a variable in our automation account.

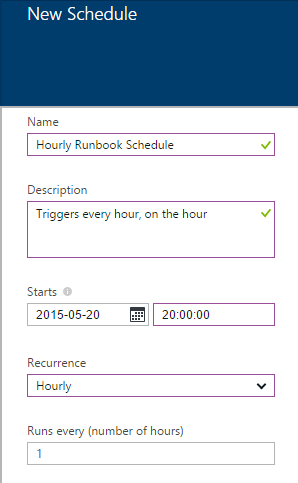
1. Open subscription in [https://portal.azure.com](https://portal.azure.com/)
2. Note your target subscription name as shown in Browse > Subscriptions
3. Open the Automation Account which will contain the runbook
4. Open the **Assets** view from the resources section
5. Open the **Variables** view
6. Click **Add a variable** from the top menu
7. Give the variable a name (“**Default Azure Subscription**” expected by default), and enter the subscription name as the variable’s value. Click Create.

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024651913.png)

Schedule the Runbook

The runbook should be scheduled to run periodically. As previously discussed, this does not determine the power on/power off schedule. It only determines how often the power schedules on resources are checked. Azure allows up to an hourly frequency, so we’ll take advantage of that:

1. Back in the runbooks list, open the new runbook “Asset-AutoShutdownSchedule”
2. Open the **Schedules** view under details
3. Click **Add a schedule** in top menu
4. Click **Link a schedule to your runbook**
5. Click **Create a new schedule**
6. Provide a name like “Hourly Runbook Schedule”
7. Set the start time to time you want to first run, e.g. the next upcoming hour mark
8. Set **Recurrence** to Hourly
9. Click **Create**
10. (Optional) If you want to provide a credential or subscription name directly and didn’t use the default names, click **Configure your runbook parameters**
11. (Optional) Enter the name of the credential asset the runbook should use
12. (Optional) Enter the name of the subscription the runbook should use
13. Click **OK** to close the open dialogs
14. Confirm the schedule now appears in the list with status **Enabled**

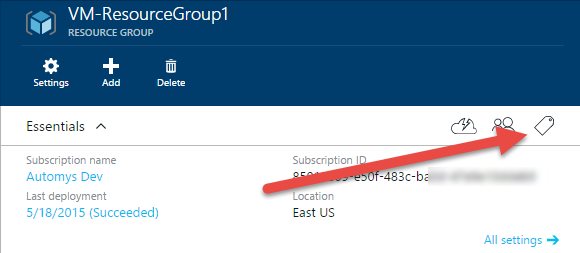
[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024652007.png)

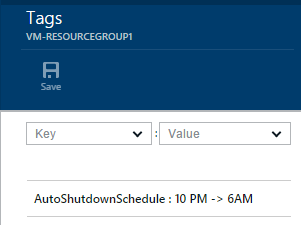
The runbook will now run every hour and perform power actions as indicated by the tags on resource groups in your subscription.

Configure Shutdown Schedule Tags

Finally, we need to tag our VM resource groups. The tag format was discussed above. To create schedule tags:

1. Open subscription in [https://portal.azure.com](https://portal.azure.com/)
2. Navigate to **Browse > Resource Groups**, and open a resource group that contains VMs to schedule
3. Click the tag icon in the upper right
4. In the **Key** field, enter “**AutoShutdownSchedule**”
5. In the **Value** field, enter a schedule as discussed above, such as “10PM -> 6AM”
6. Click **Save** in the top menu

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024652110.jpg)Azure resource group tag configuration

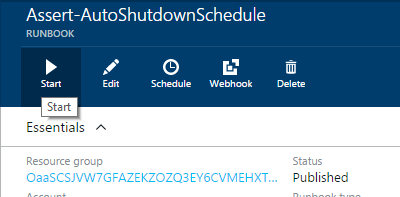
[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024652188.png)Example shutdown schedule tag

After repeating this process for each VM resource group in your subscription, everything is set to automatically shut down and start up your virtual machines. Going forward, you can simply update the tag as needed to adjust the schedule, and add a tag to new resource groups that require a shutdown schedule. Remember, VMs in untagged resource groups will not be managed by the runbook.

Initial Testing

To validate that the runbook works, we can run an initial test manually and inspect the results. This is easy:

1. Assign a shutdown schedule tag to the VM or resource group you want to use for testing. Give it a schedule the covers the current time. The easiest way is to just use today’s day of the week, e.g. “Wednesday”.
2. Start the test VM(s)
3. In the runbook view under your automation account, click the **Start** button from the top menu.
4. Verify the parameters are correct if you opted not to use the defaults. Set **Simulation** to **True** in order to test without making changes. Verify**Run on Azure** is selected and click **OK**
5. Open the Output view, and wait for the runbook to execute. It takes a minute or two to queue and run.

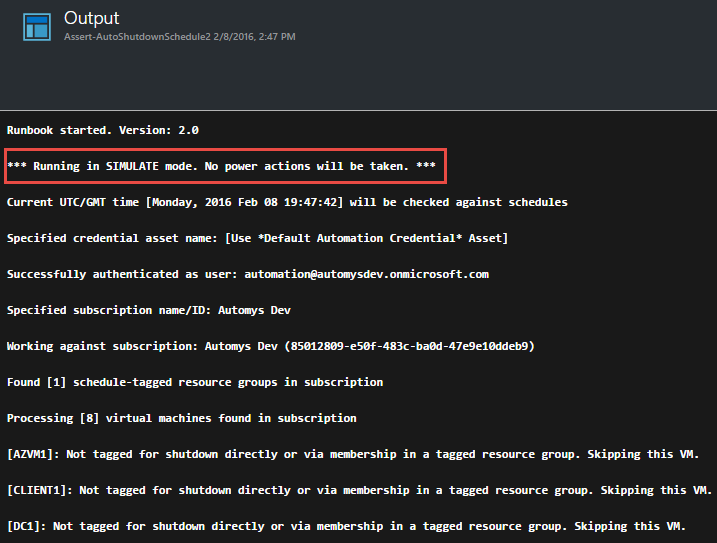
[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024652251.png)Start runbook for a manual test

At this point, we hope to see messages in the output telling us a tagged VM or resource group was found, that the current time is within a shutdown schedule, and that the intended VMs would have been stopped in a normal execution. Any errors that occur should also be recorded in the output.

Now, test the opposite case: starting VMs that should be running according to the schedule (if they aren’t in an explicitly-defined shutdown period, they should be started). So, we can update our schedule tag and test again as follows:

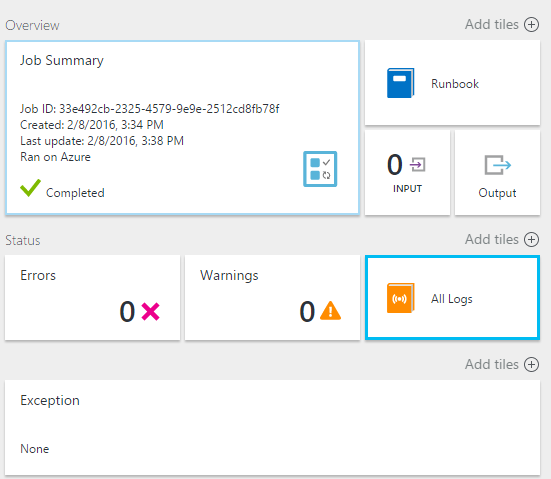
1. Go back to the test VM or resource group and set the AutoShutdownSchedule such that it doesn’t cover the current time. For example, if today is Wednesday, set the tag value to “Tuesday”. Setting the tag again and saving overwrites any existing tag with the same name. (Hint: you can use the dropdown to select previous tag keys and values).
2. Now start the runbook again using the same steps as before and watch the output

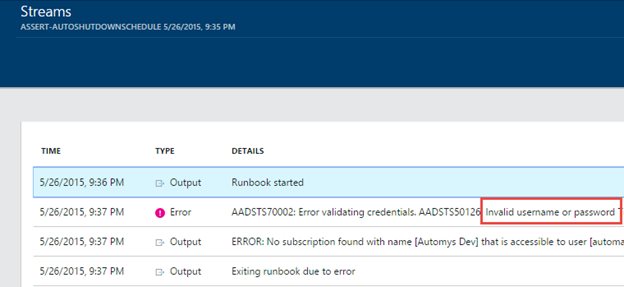
This time, we should see that the current time doesn’t match any shutdown schedules for the VM or group, and see the runbook report that it would have started the intended VMs.

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-160208205153042.png)Testing runbook in Simulate mode

Troubleshooting

To check for problems, you can inspect the runbook job history to look at the output and streams / history for each individual job. In the new portal, the output view doesn't necessarily show error details, so make sure to check the Streams view as well.

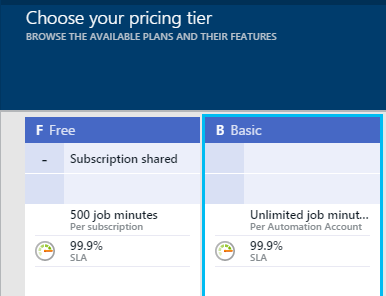
[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-160208204551287.png)Check runbook job logs for errors or warnings

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150527141445200.jpg)All Logs view contains exception details not shown in output view

Automation Account Configuration

Before putting this runbook into production where you count on it to reliably manage your VM power state, I recommend configuring your automation account as a “Basic” rather than a free account. This ensures that the 500 minute monthly [run time limitation](http://azure.microsoft.com/en-us/pricing/details/automation/) will not be hit and prevent the runbooks from working. The cost is extremely low for additional minutes, so the few extra dollars, if any, will easily be offset by the compute time savings.

This can be changed in the “Pricing Tier” view under the automation account in the portal at [https://portal.azure.com](https://portal.azure.com/).

[](https://automys.blob.core.windows.net/images/scheduled-virtual-machine-shutdown-startup-microsoft-azure-150521024652995.png)

Taking it Further

Please share your experiences with this runbook in the comments below, and if you have ideas for making it better, don’t be shy!