# EITS Runbooks

# A. Create a Storage Account & Upload Arm Template

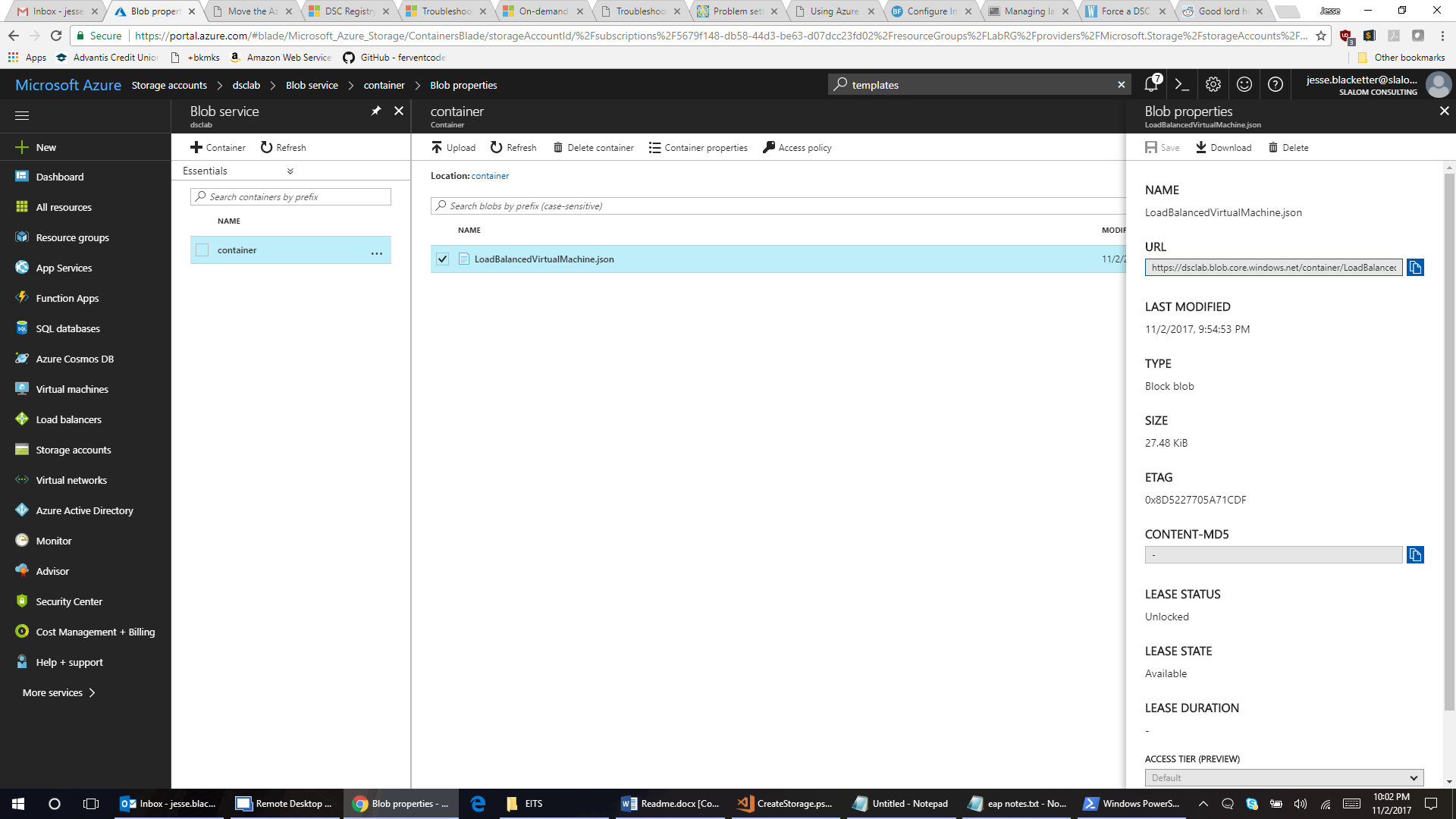
1.) Run the powershell Create-storage.ps1. This requires the following parameters (adjust the values in quotes accordingly each time this script is run):

$ResourceGroupName = "rg-automation"  
$Location = "eastus2"  
$ProjectName = "hbtest"

The script will create a container within the blob storage of the resultant storage account and upload the ARM template there called LoadBalancedVirtualMachine.json.

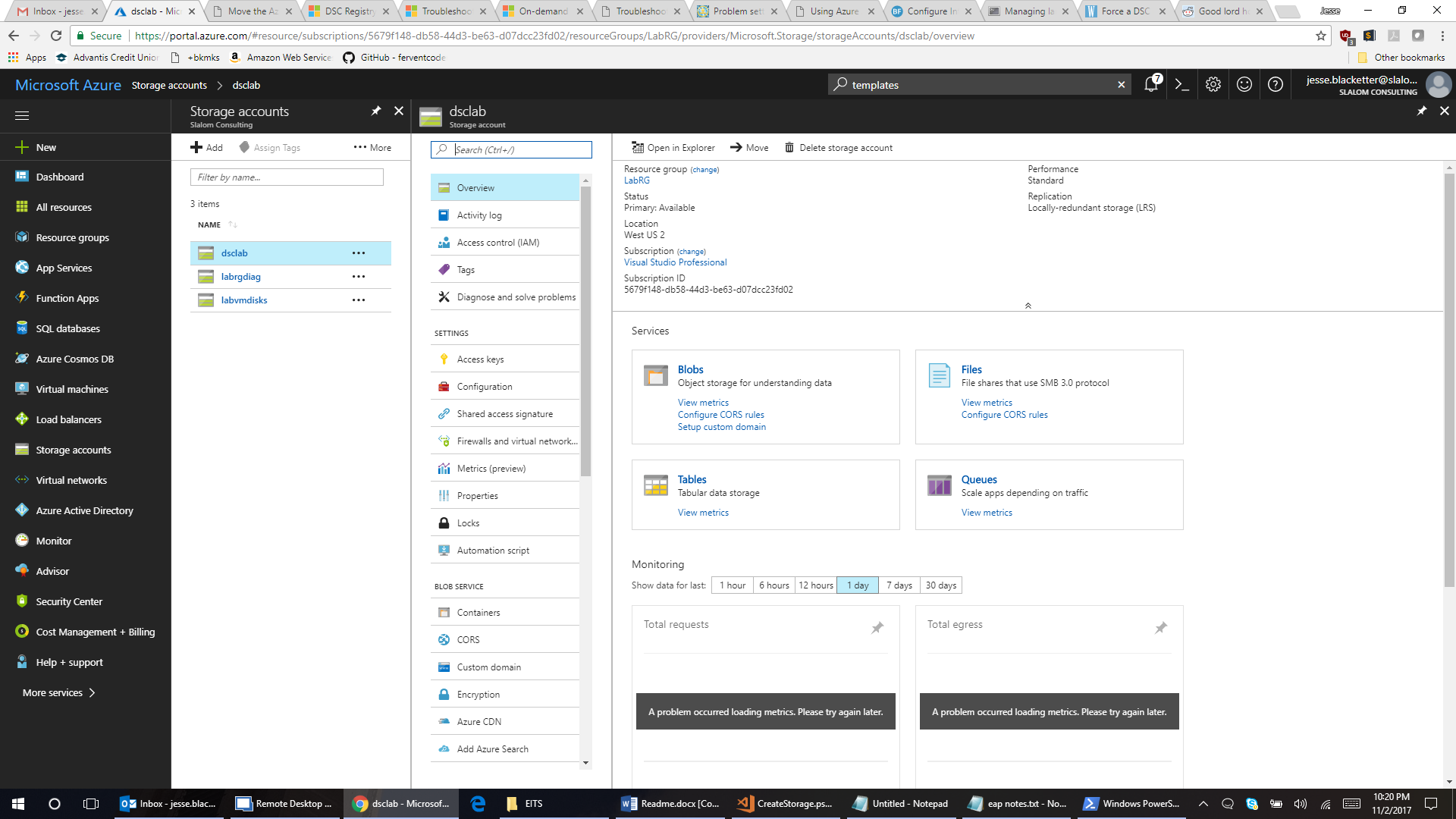
Look at Storage accounts in the Azure Portal and within the storage account you just created. Within that storage account, click on Blobs and then the name of the container you created. Now click on the LoadBalancedVirtualMachine.json file and make note of the URL listed that pops up on the right side of the page. It will be something like:

<https://hbtestautomationstorage.blob.core.windows.net/hbtestautomationstoragecontainer/LoadBalancedVirtualMachine.json>

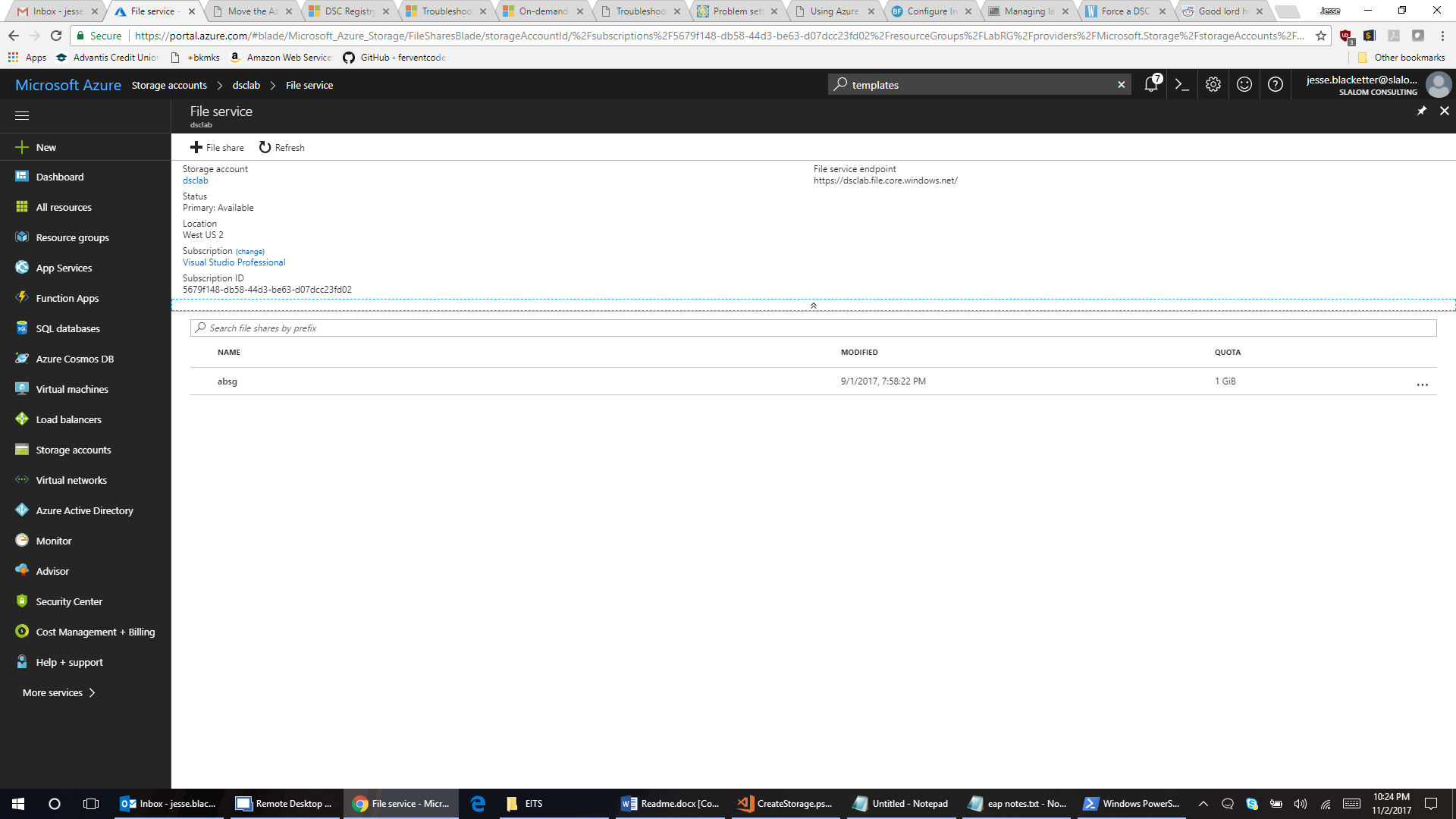


# B. Create File Storage For Installer Files

1.) In the storage account created in Step A, choose Files in order to make use of the Azure File Service. See the screenshot below for reference:



3.) Create a file share (e.g. in this case we named the file share “absg” but this can be named whatever is preferred).



4.) Within the newly created absg file share, you must create the appropriate directory structure for the SCCM client installation exactly as it exists coming from the SCCM team. The directory structure must be setup as shown below in the text or diagram example:

absg

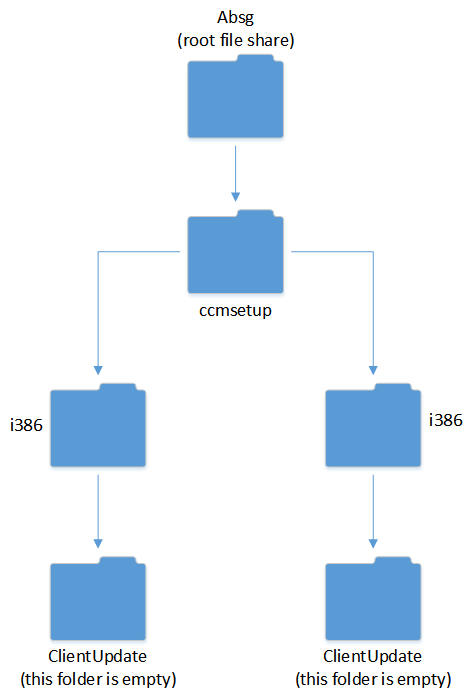
ccmsetup

i386

ClientUpdate

x64

ClientUpdate

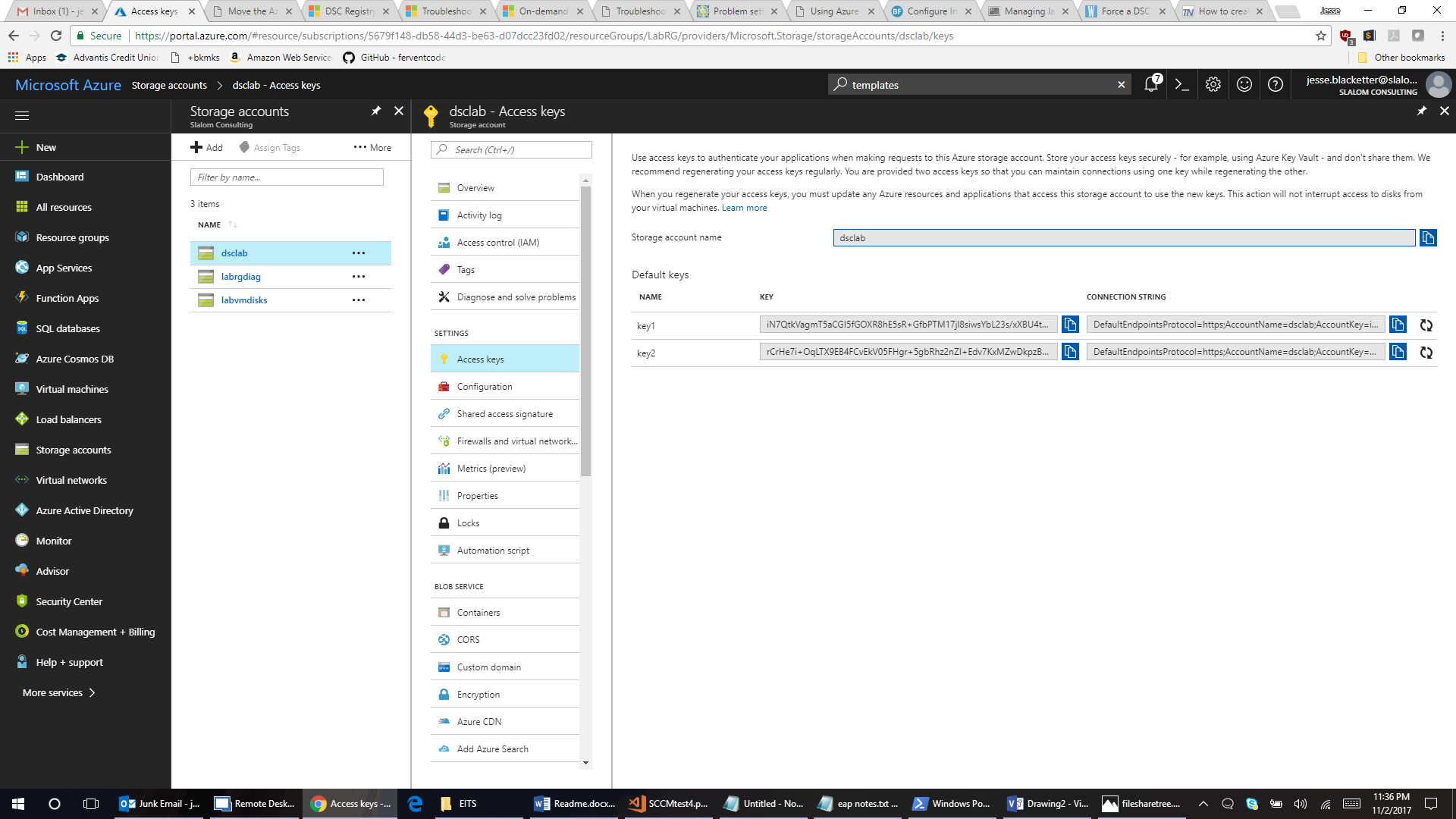


5.) Upload ccmsetup files to their appropriate directories after creating the directory structure.

Go back to the storage account view as shown here:



6.) Click on the storage account you created your file share in where ccmsetup files are stored. In this view, you will see a center column with a series of choices. Under the settings section, click on Access keys shown below:

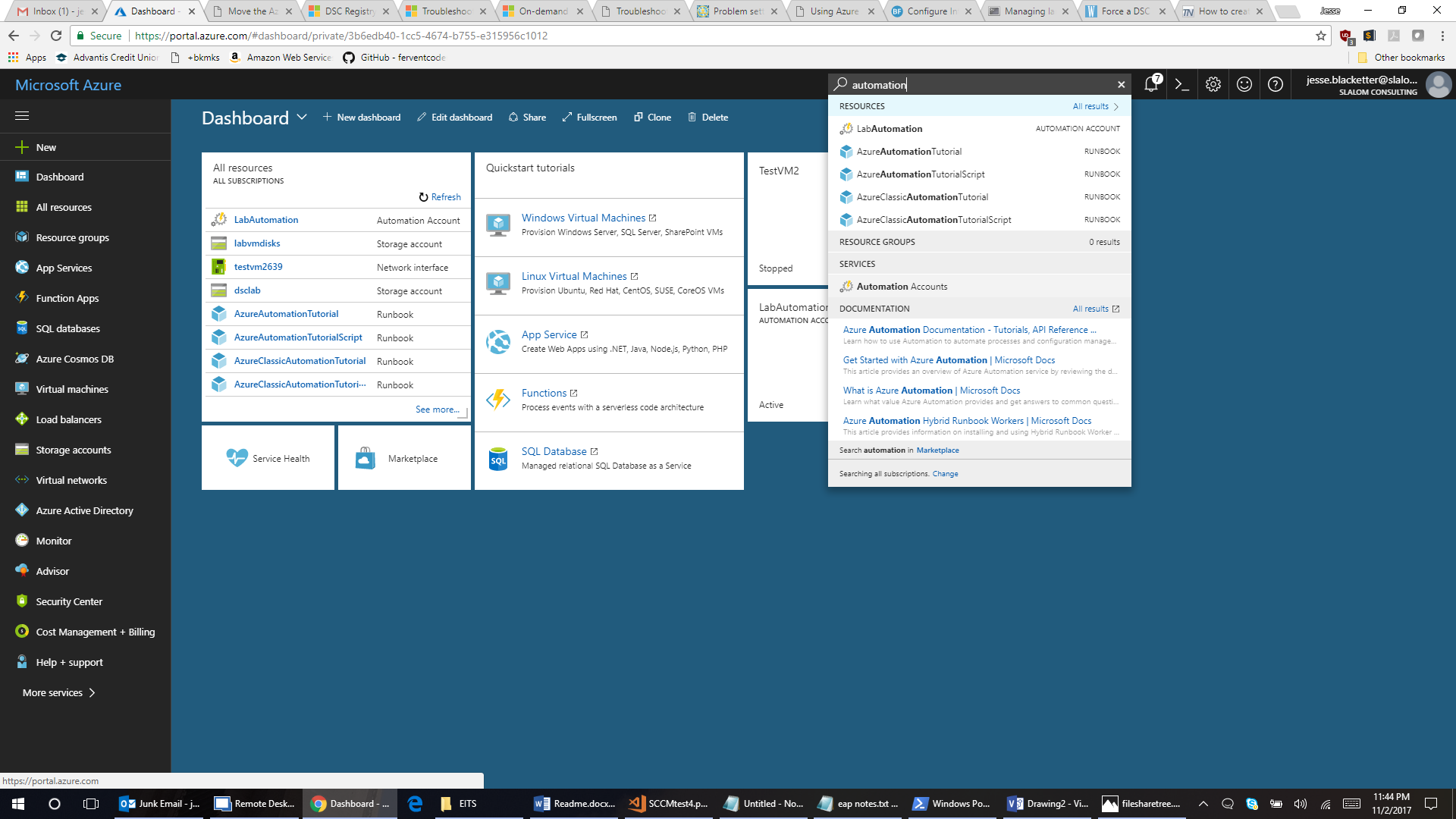


**Make note of the key & Storage account name**

Note down the Key & Storage Name (you can use either key1 or key2 but in our examples we are making use of key1) You will need it when creating the Credential Asset in the Automation Account.

# C. Create an Automation account

1.) To create an Automation account, it is easiest to use the search function at the top of the Azure Portal page as illustrated below. Type automation in the search field and notice under services “Automation Accounts” is available. Select to continue.



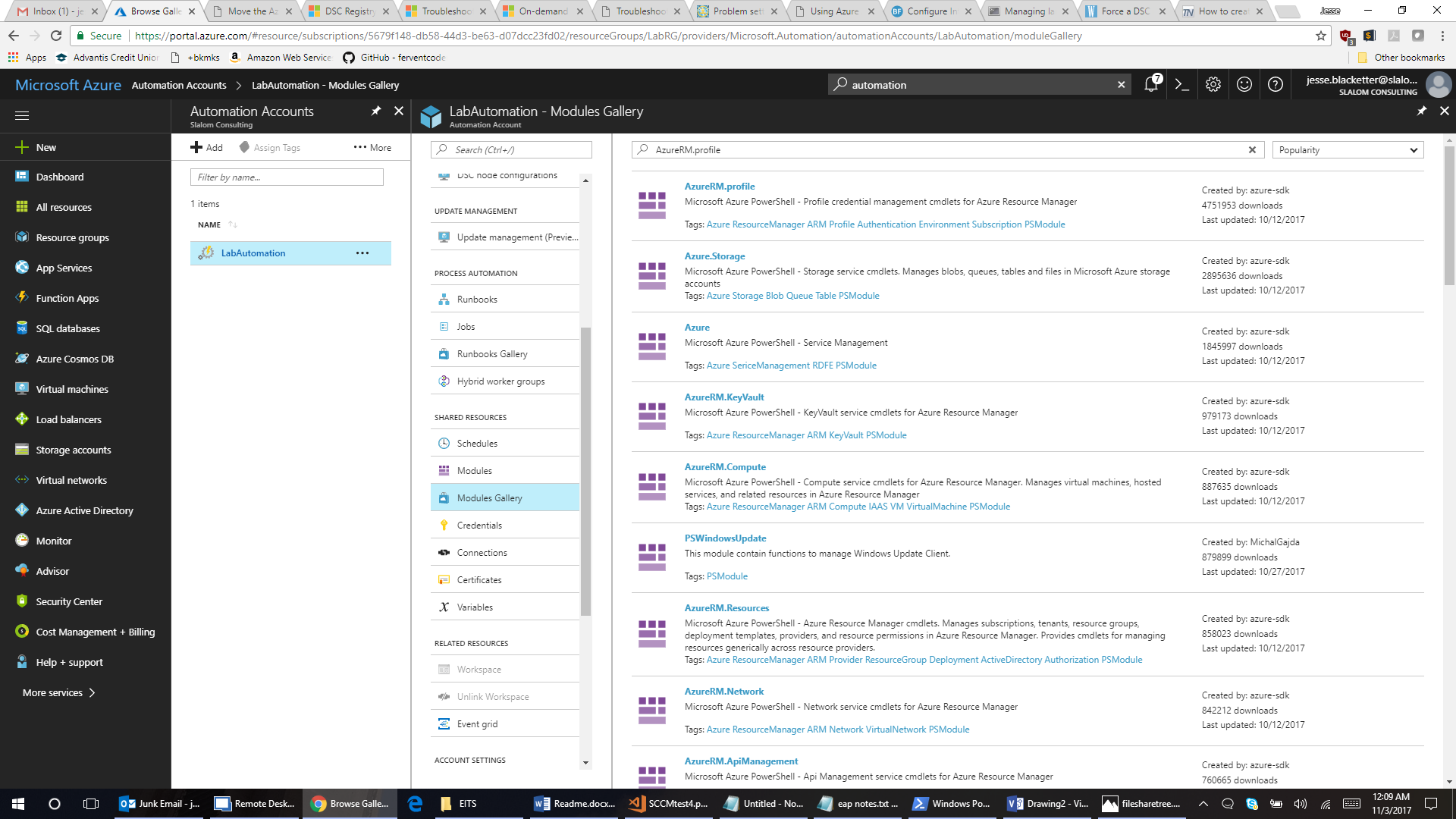
2.) Add an Automation account with appropriate naming, subscription placement, Resource Group, location, and it may be best to leave the check box to pin this account to the dashboard for easy access later.



### Add the following modules from the module gallery

1. AzureRm.Profile (wait for a while after importing this)
2. AzureAD
3. AzureRm.Network
4. AzureRM.Tags
5. CredentialManager
6. AzureADPreview
7. AzureRM.KeyVault
8. MSOnline

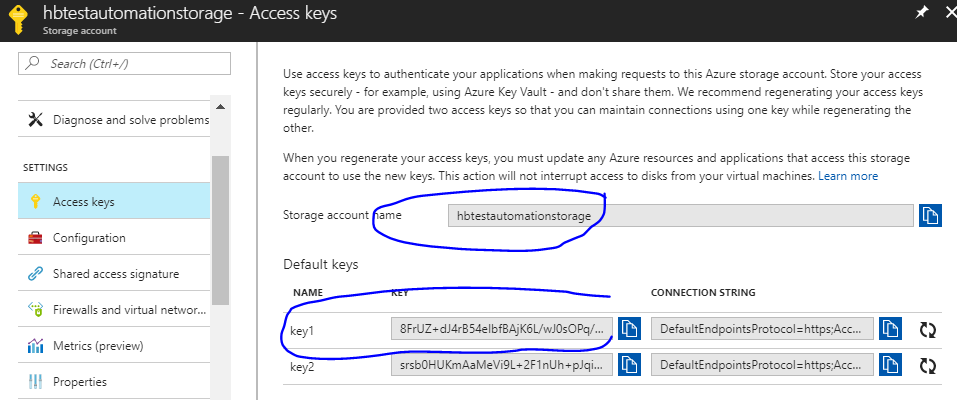
To add modules to an Automation account, you must navigate to the account you just created and look for the Modules Gallery in the center column under the Shared Resources section as shown below. Use the search function in the right column to find and install each of the above listed modules.



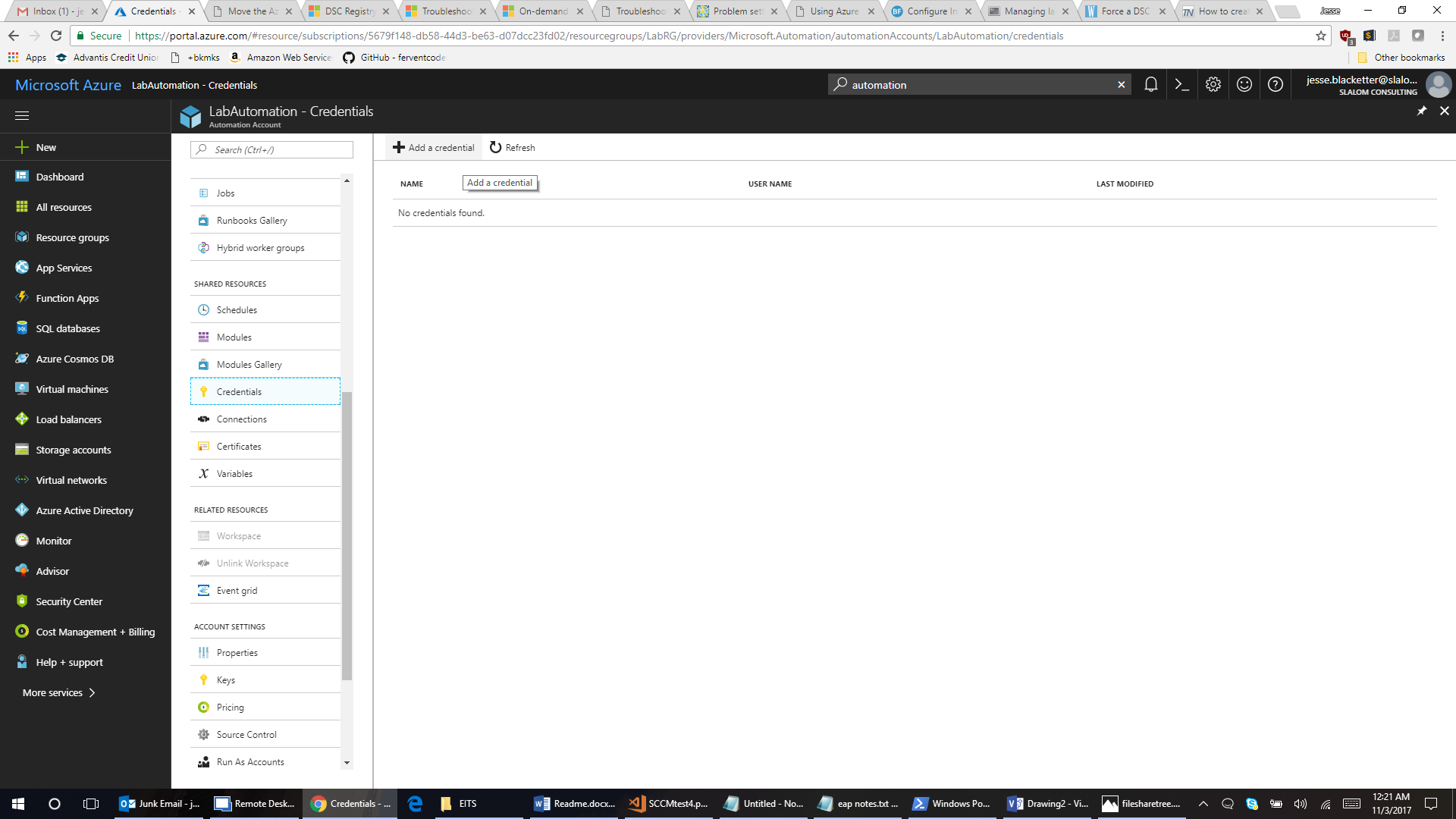
### Create a Credential variable

In Section B, you created a file storage to hold the ccmsetup installer files. In the Automation account, you create a credential asset called **DSCPackageStorage** that will be used as a variable within the DSC configuration to allow access to the file storage.

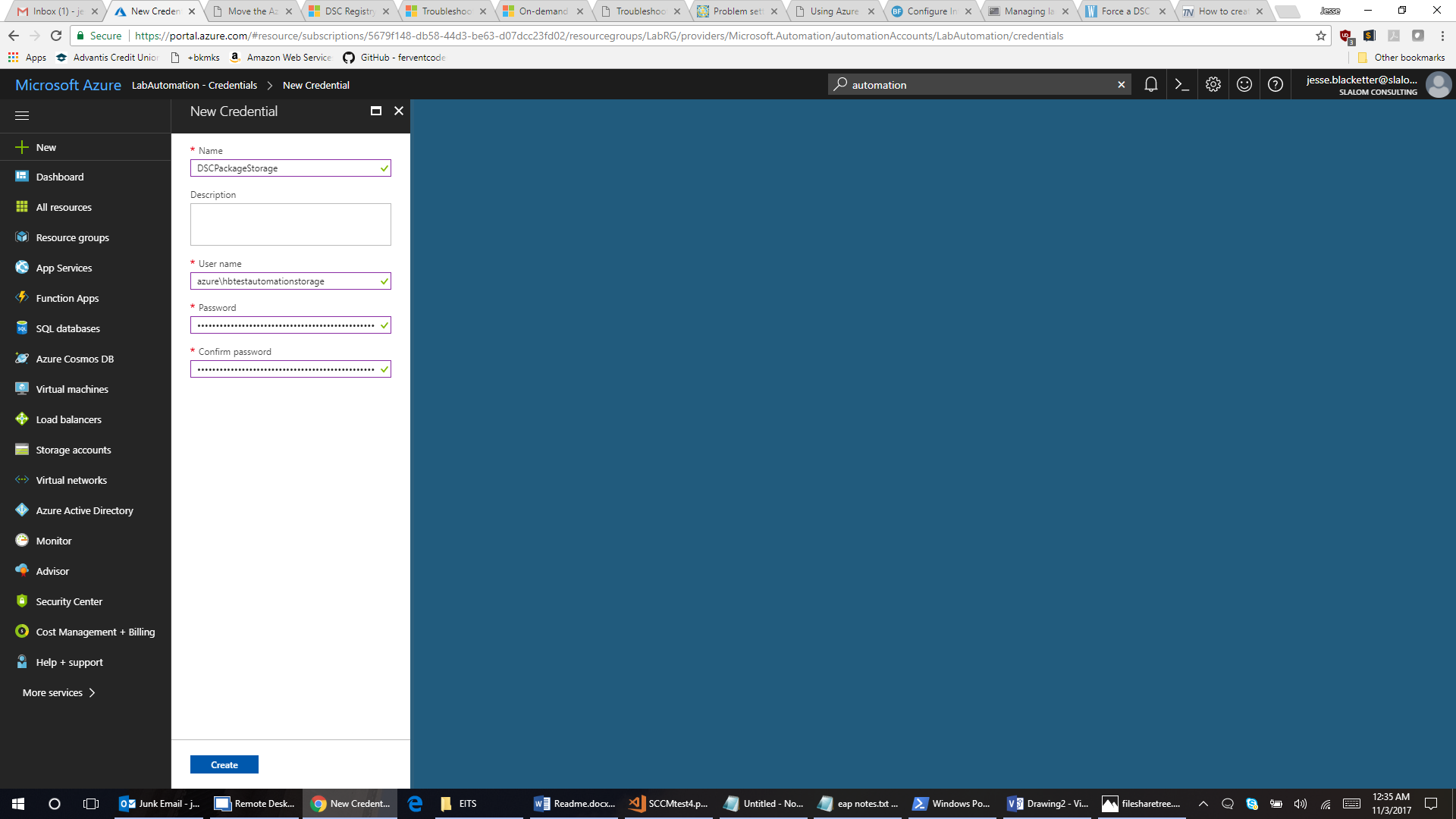
1. Gather the username and password to access the file share by going back to your storage account and Access keys under the settings section.



1. Navigate back to your Automation account and look in the center column under Shared Resources for Credentials. Click on Add a credential to proceed as shown below:



1. Give the credential asset the name **DSCPackageStorage** and use the storage account name and key you made note of back in section B of this guide after setting up the file share for ccmsetup. You will use the storage account name and key in the username and password field of this credential asset.



**Name**: DSCPackageStorage

Username: AZURE\file-storage-name (this is the name of the storage account, prefixed by "AZURE\" )

Password: Key of the file storage

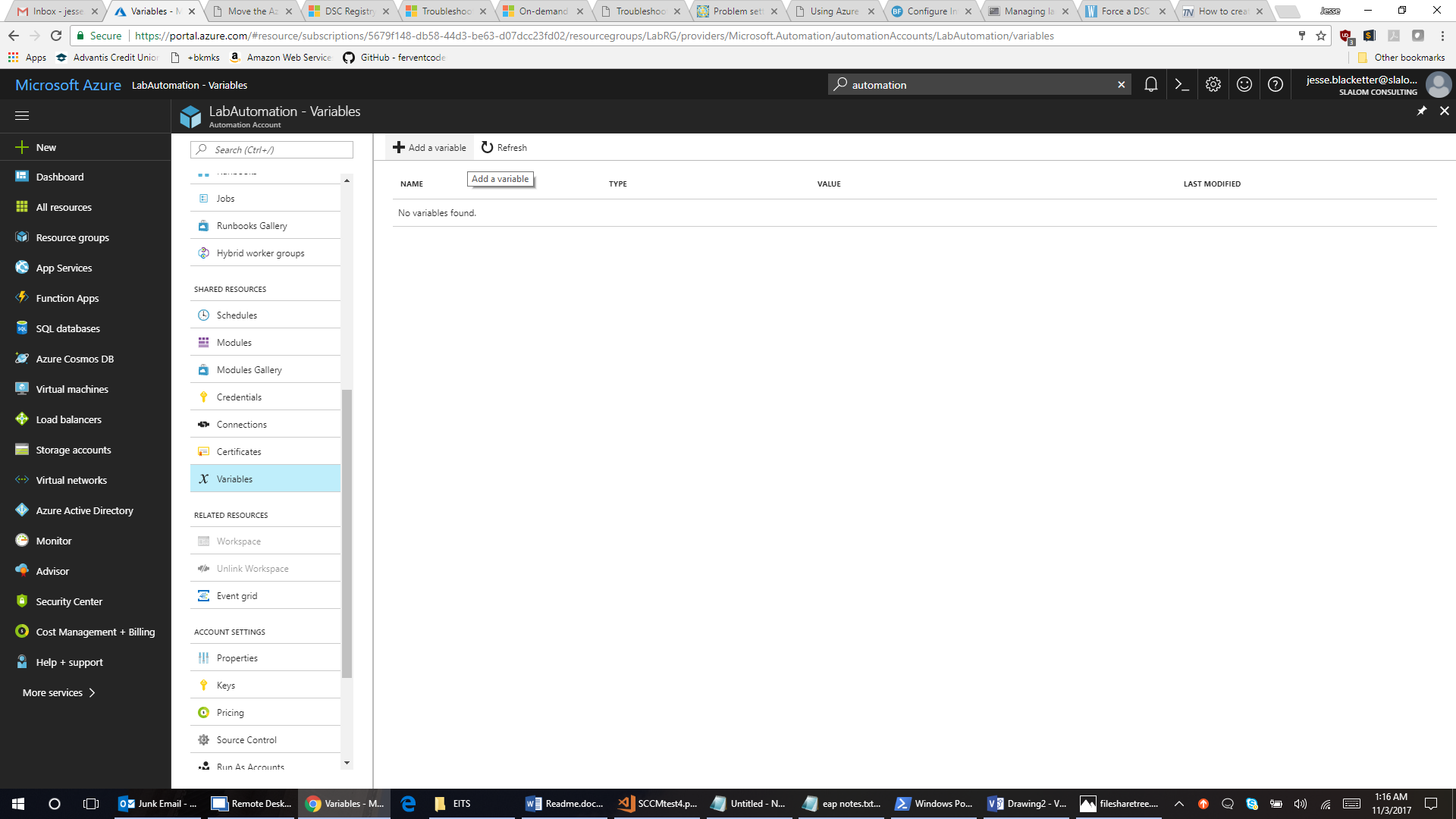
Example (used in the above screenshot):

Username: Azure\hbtestautomationstorage

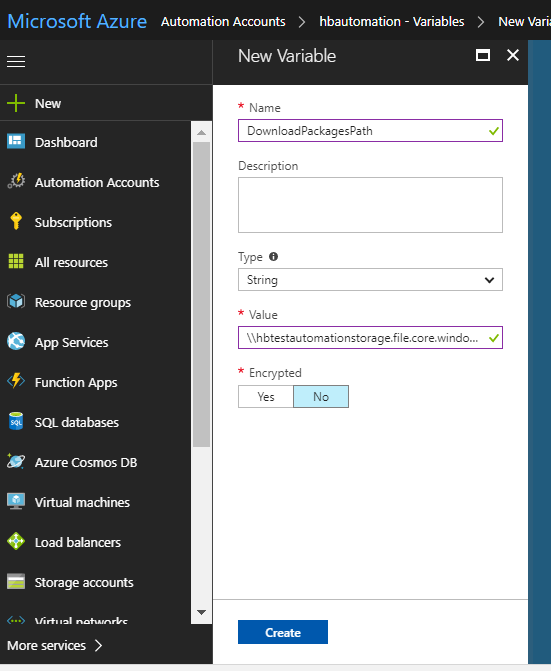
Password: 8FrUZ+dJ4rB54eIbfBAjK6L/wJ0sOPq/XYQ9j4F7gig7kEiQWSY9mXlucHefGTCluq4kLPXvf7TgPZ+hPYNJ7A==

### Create an Automation Variable Asset

1. Return to the root of your Automation account and this time look for the Variables function under Shared Resources and choose Add a variable as shown here:



1. Give the Variable a name, type is string in this case, and UNC path to the ccmsetup folder under your file share. Example paths are shown in the screenshot and following text below:



Name: DownloadPackagesPath

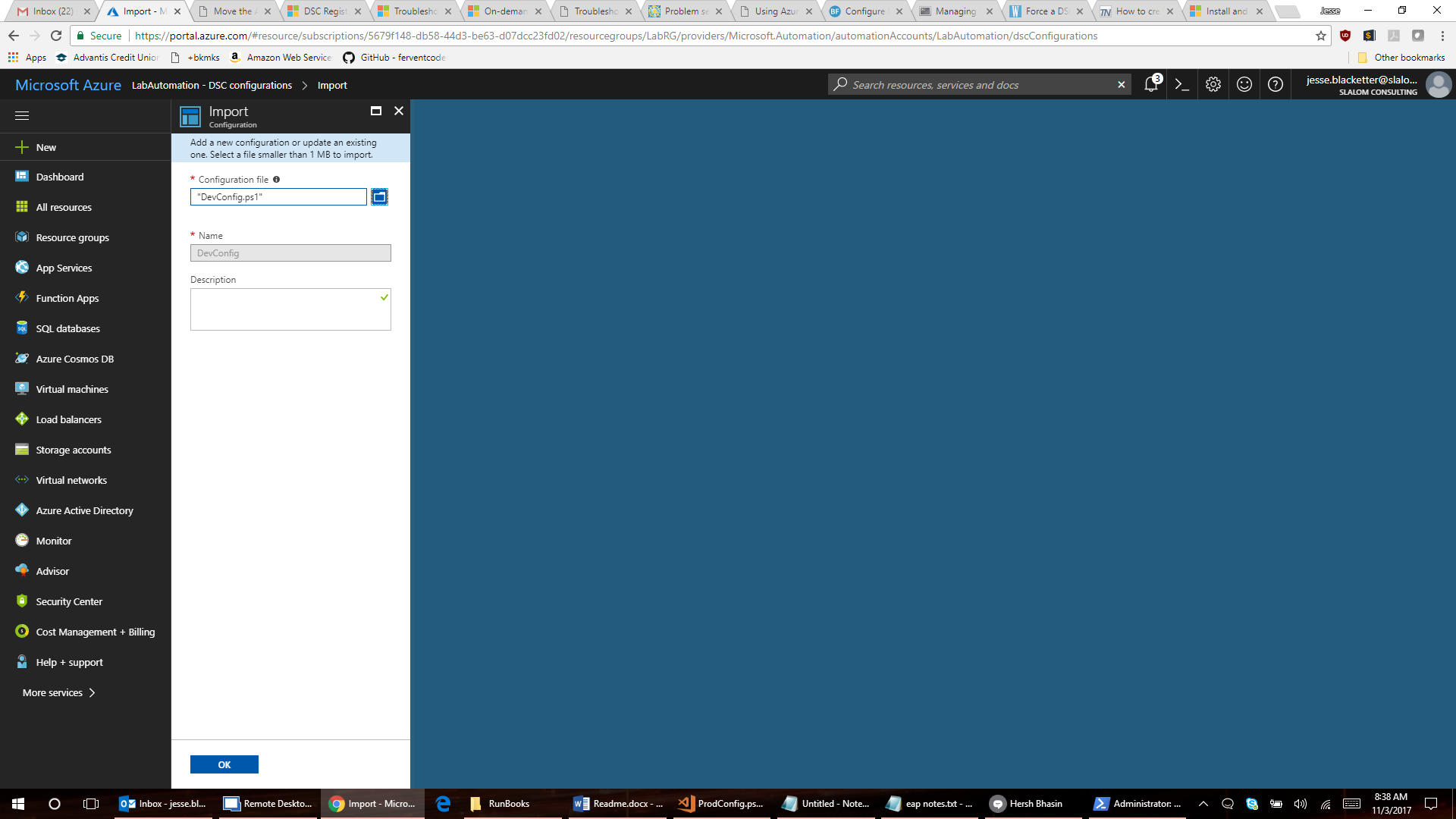
Value: \\[storageaccountname].file.core.windows.net\absg\ccmsetup

(example: \\hbtestautomationstorage.file.core.windows.net\absg\ccmsetup)

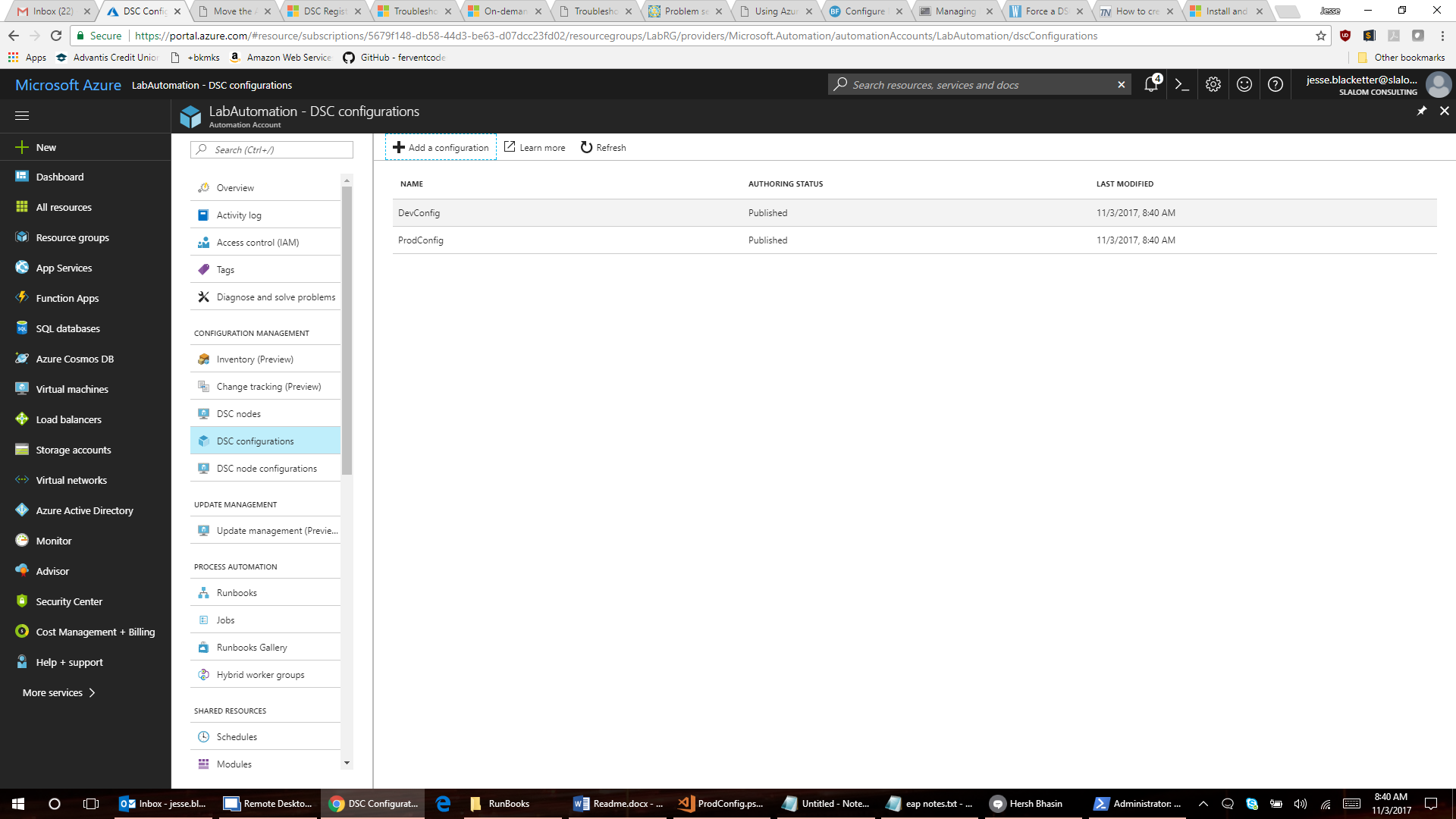
# D. DSC Configuration File

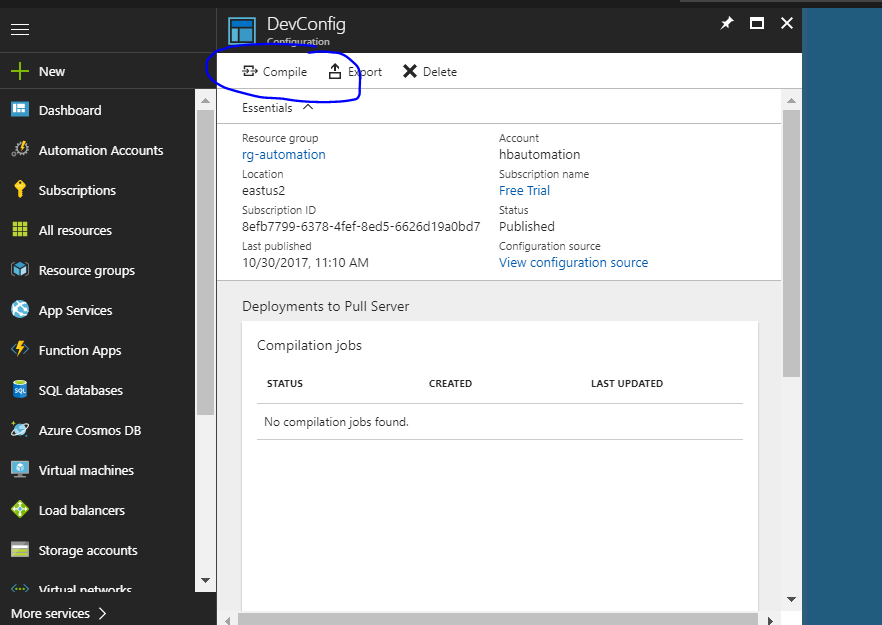
1.) From this git project, import two files called DevConfig.ps1 and ProdConfig.ps1 to the **DSC Configurations** section in your-automation-account



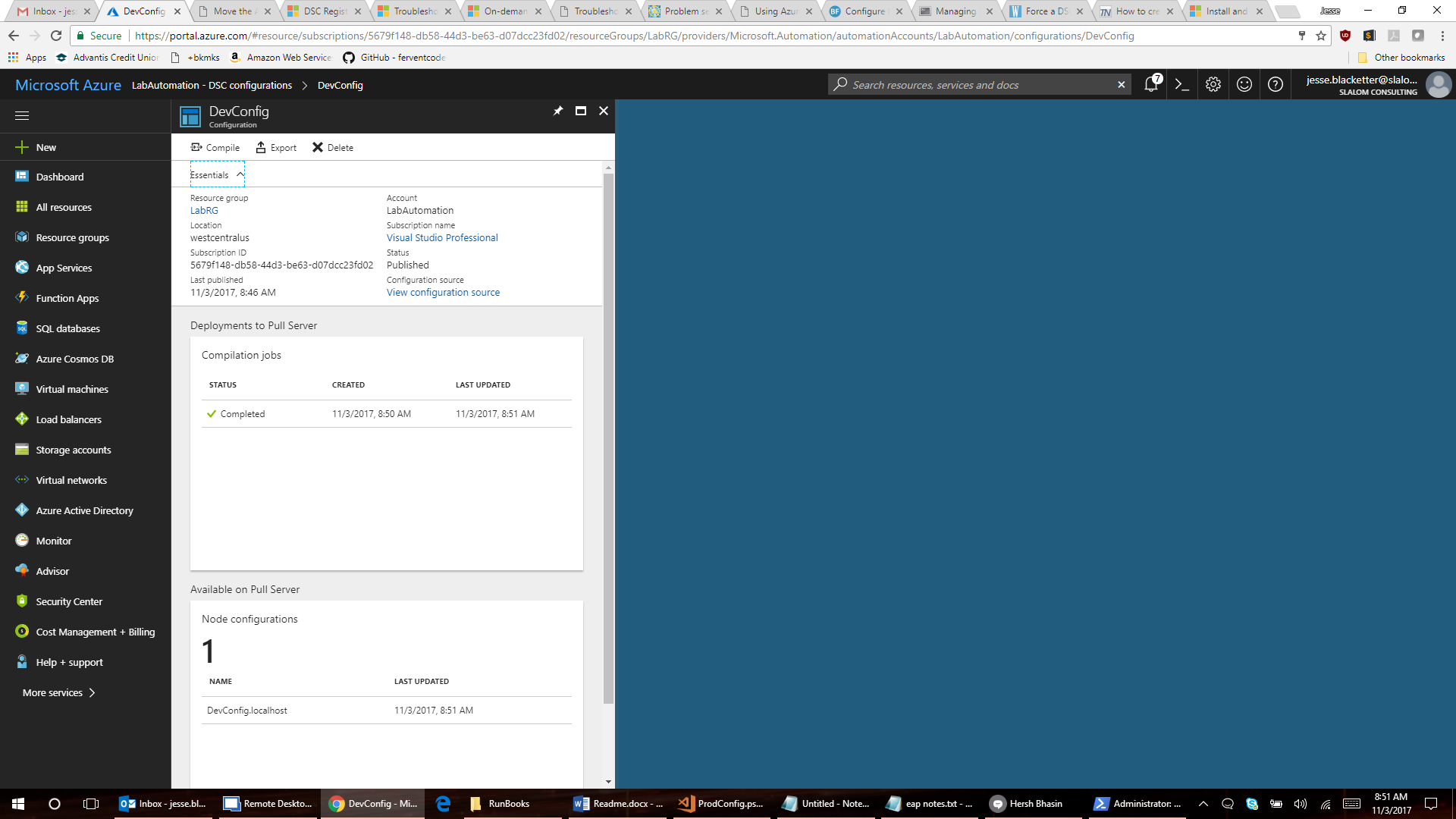


2.) When Imported, Click on it, compile it by choosing Compile on the Toolbar. Let it default for the ComputerName ("Default will be used").

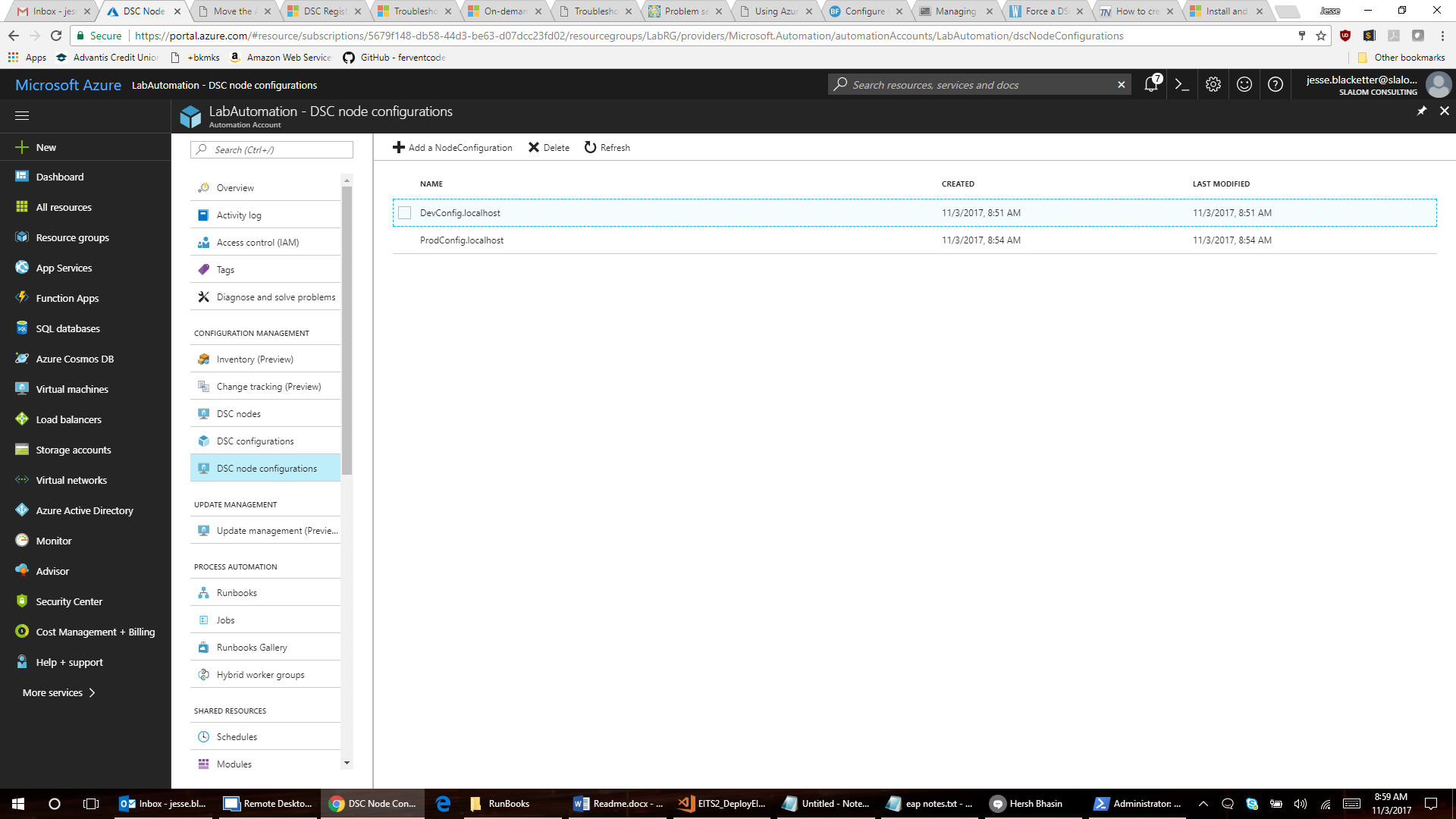




3.) The screen does not automatically refresh so close this blade by clicking the x at the corner of the blade to close it to return to the summary page of imported DSC configurations and wait for approximately one minute before clicking on the configuration again to check for status. You are looking for a completed status like this:

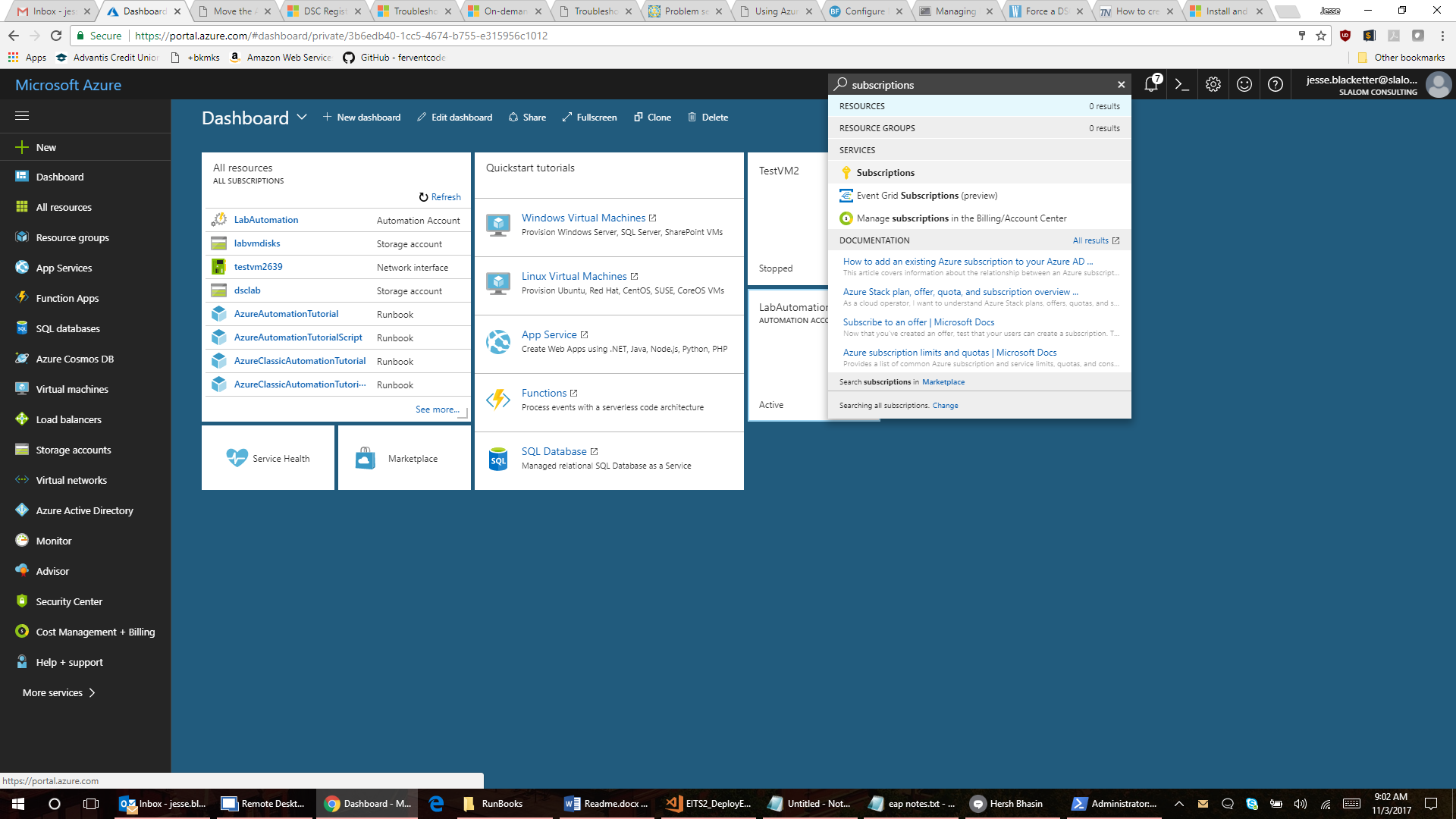


4.) After it is published, click on the link "DSC node configurations, note the resultant name of the configurations (DevConfig.localhost and ProdConfig.localhost). You will need to pass either of these by complete name as a parameter to the Runbook EITS2\_DeployEITS\_Env.ps1 to determine whether the VM being provisioned gets a Dev ccmsetup configuration or a Prod ccmsetup configuration. This parameter is named $NodeConfigurationName.

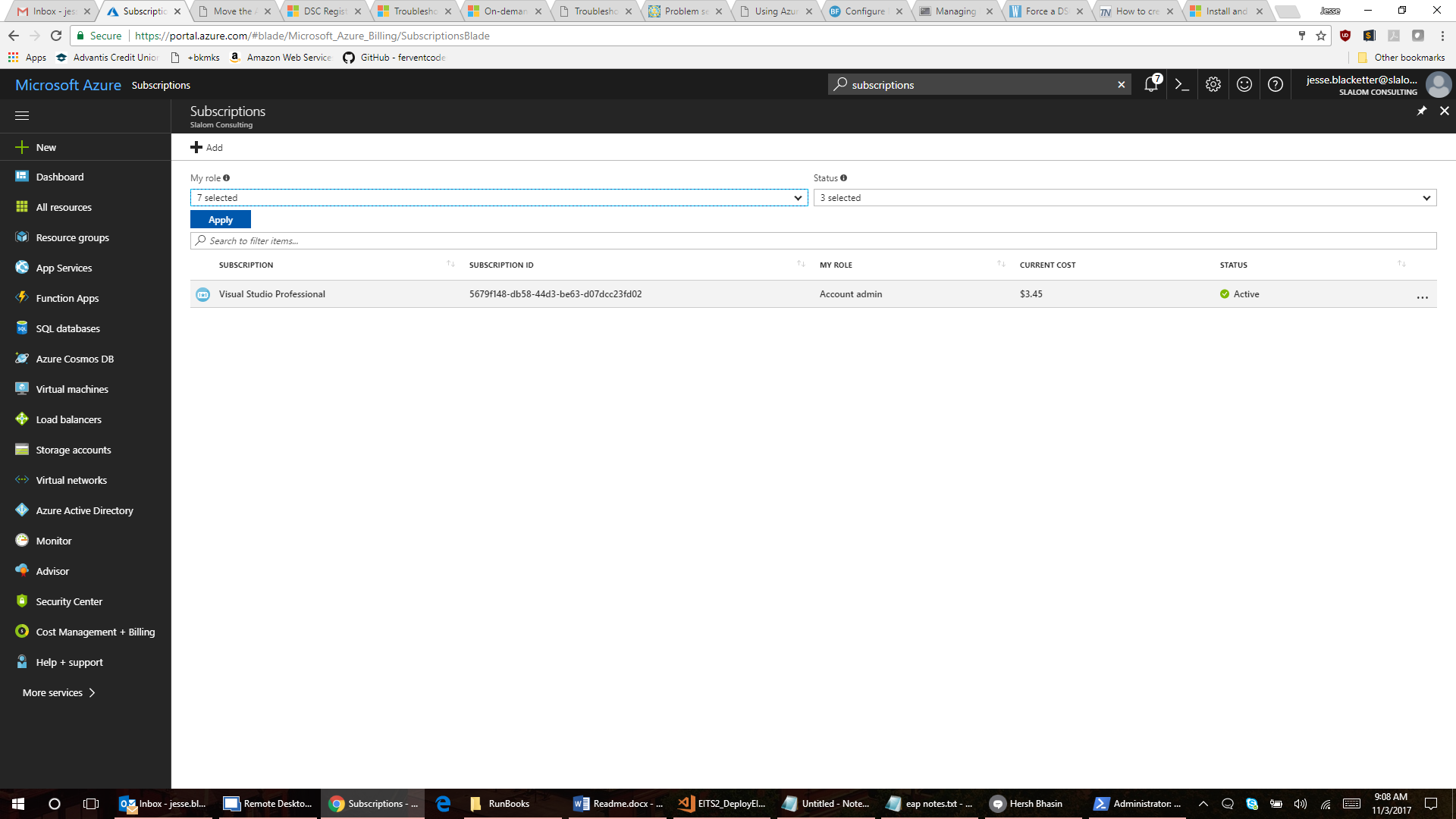


# E. Assign the automation account to Owner Role

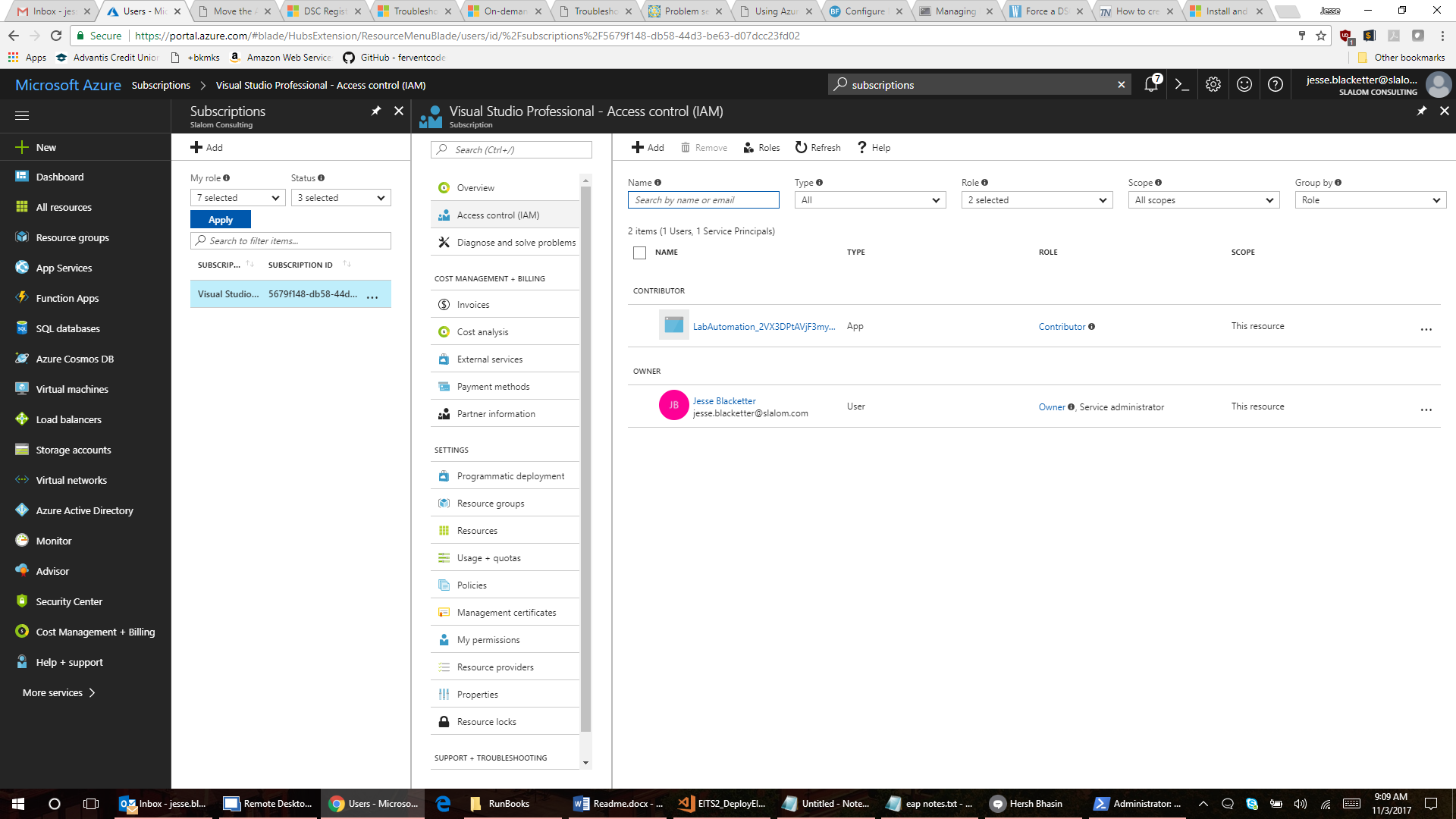
1.) In Azure Portal go to Subscriptions. It is easiest to use the search function:



2.) Select your subscription

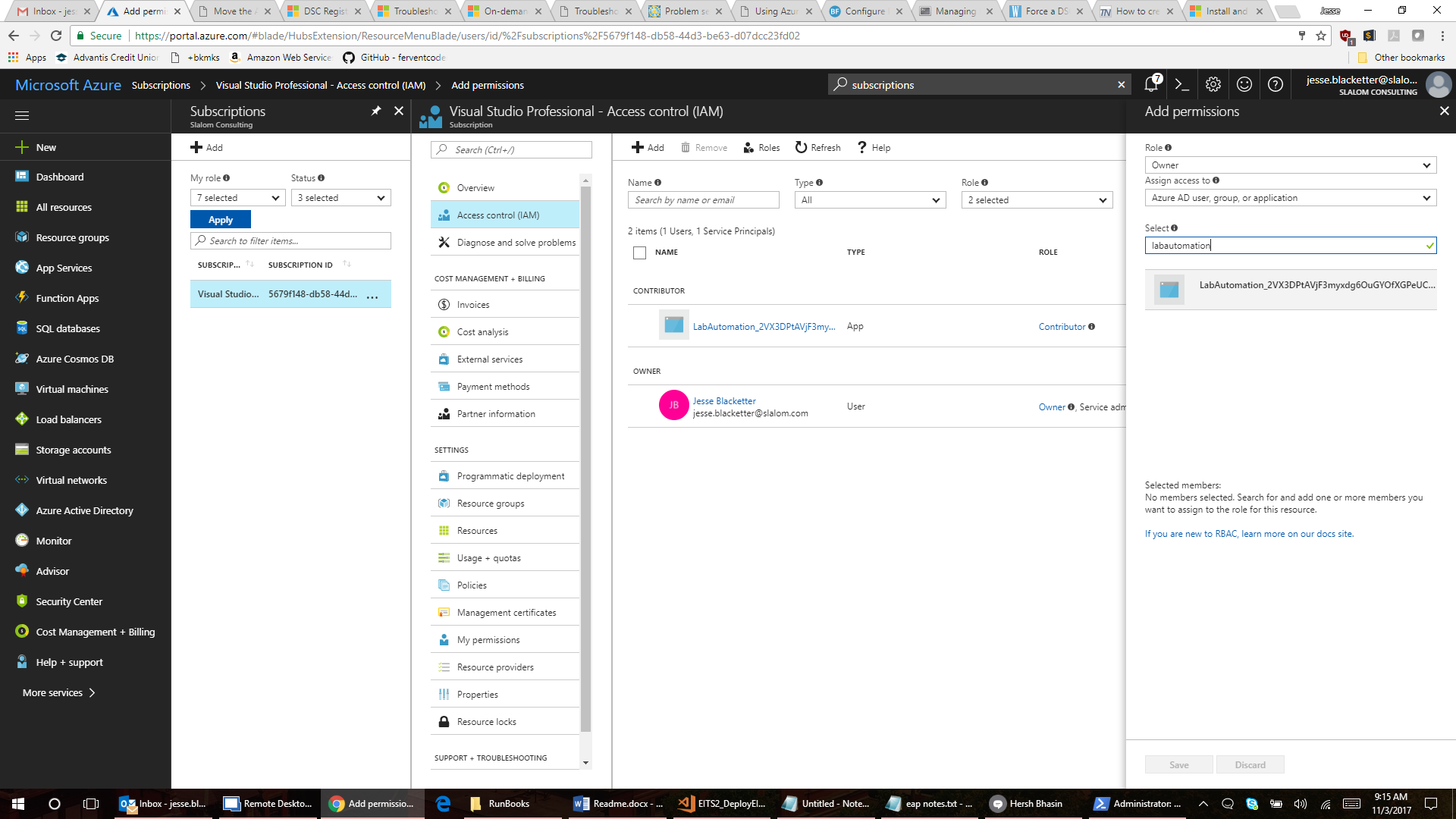


3.) Click on Access control (IAM) in the center blade and click on Add.



5.) You will need to set permissions with the following details and know the name of the Automation account you created earlier:

Role: Owner; Assign Access to: "Azure AD user, group, or application", In select, start typing the name of your automation user (imp. start typing else it is hard to find)

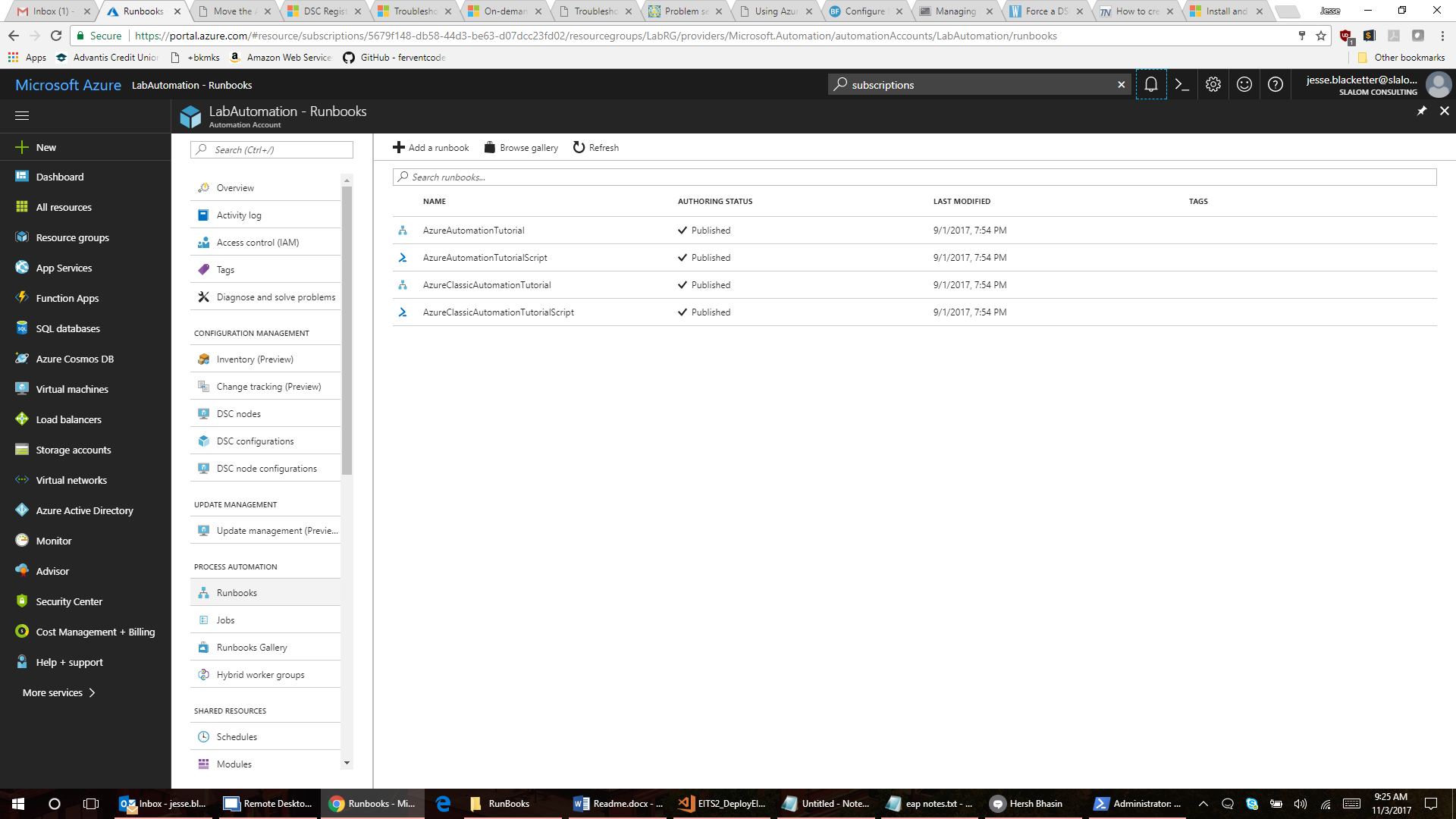


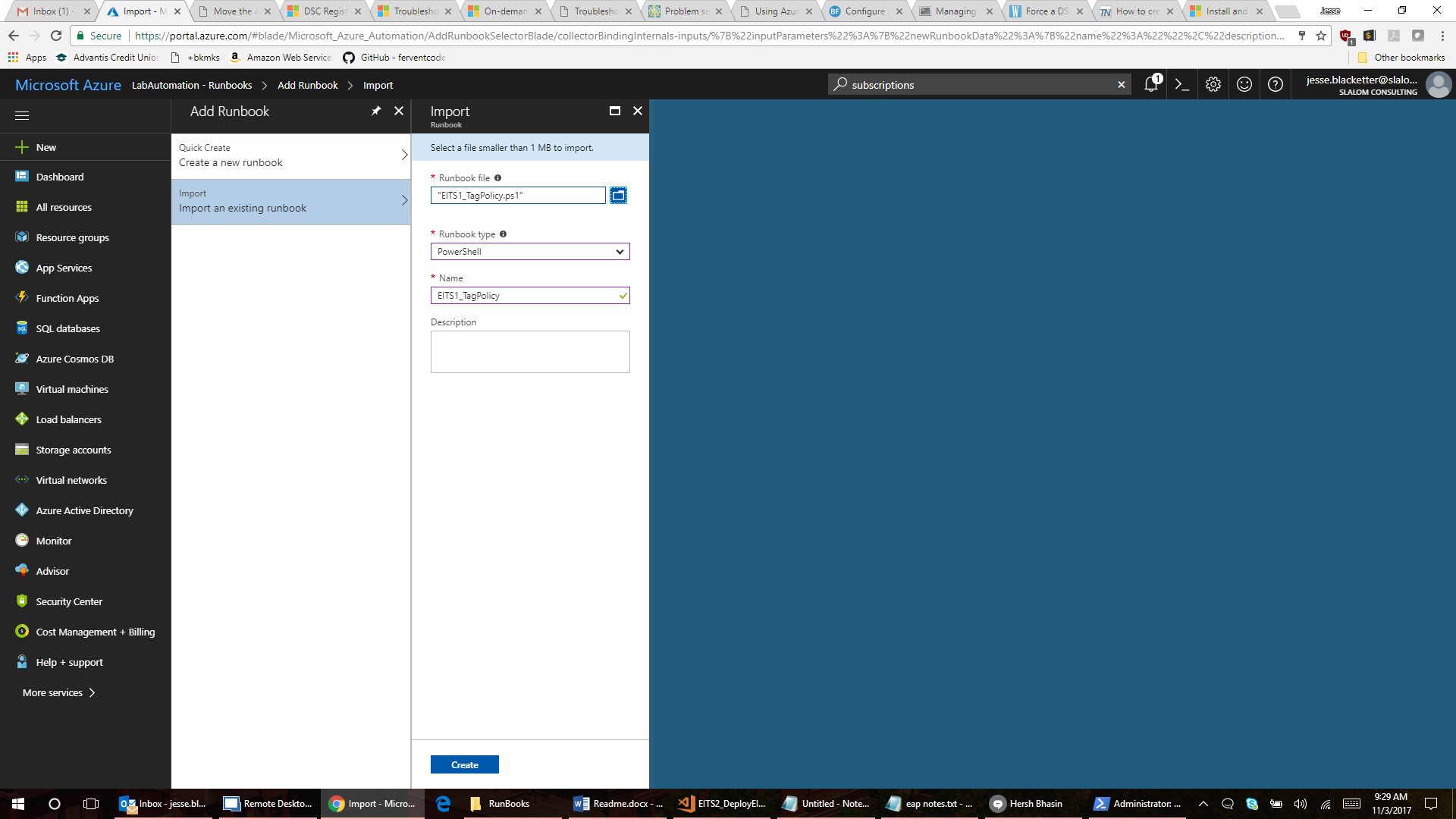
# E. Credential Automation Asset of an AD User with "Owner Rights" for domain joining.

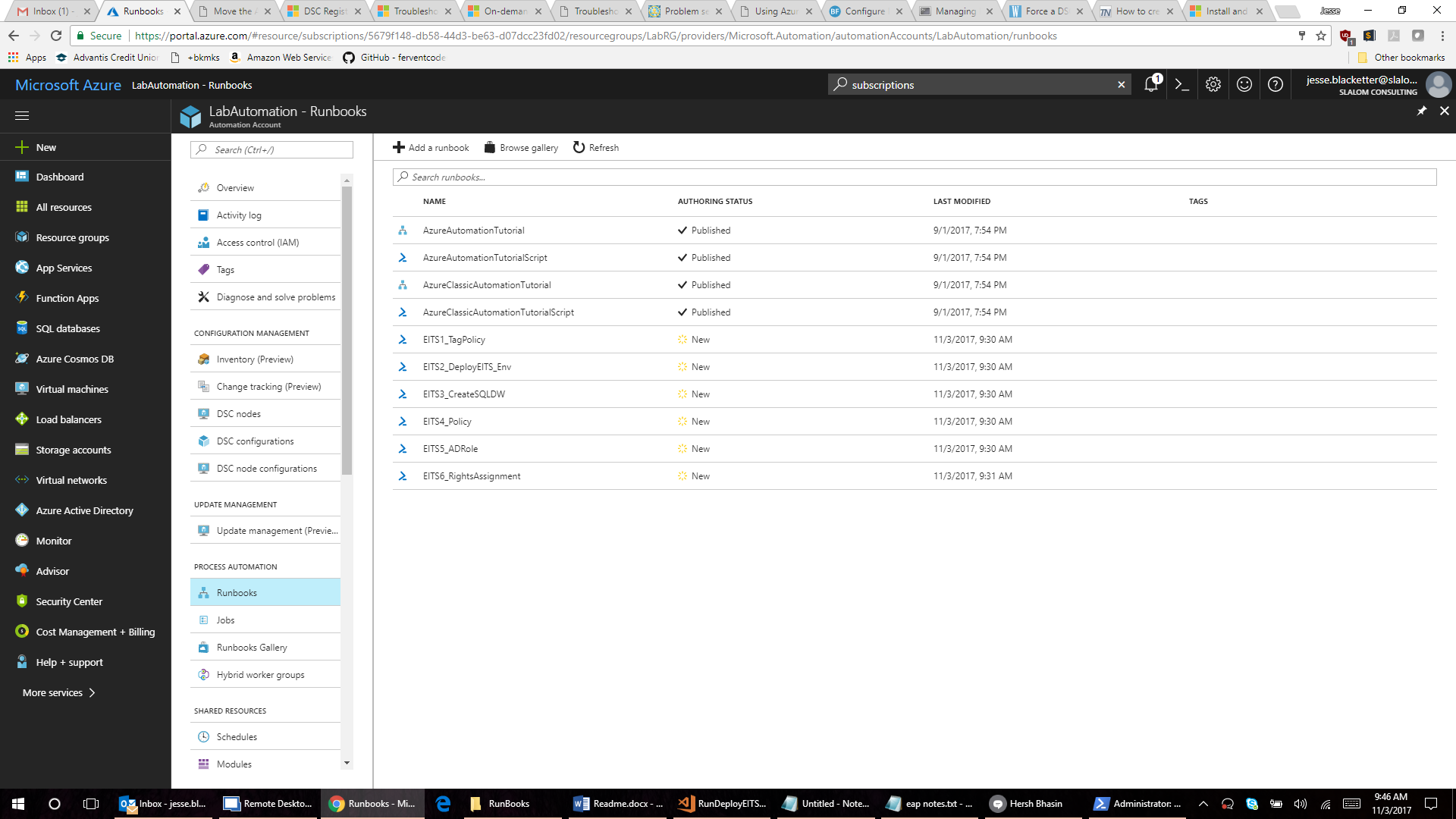
1.) You need to have an AAD user with Owner rights. This should be a service account with rights to join any resultant VM’s to the domain. Follow the steps above to give that account "owner rights" to the subscription.

# F. Import & Run Runbooks

1.) Return to your Automation account home and select Runbooks under the Process Automation section of the center blade column and then Add a runbook to import all of your runbooks.

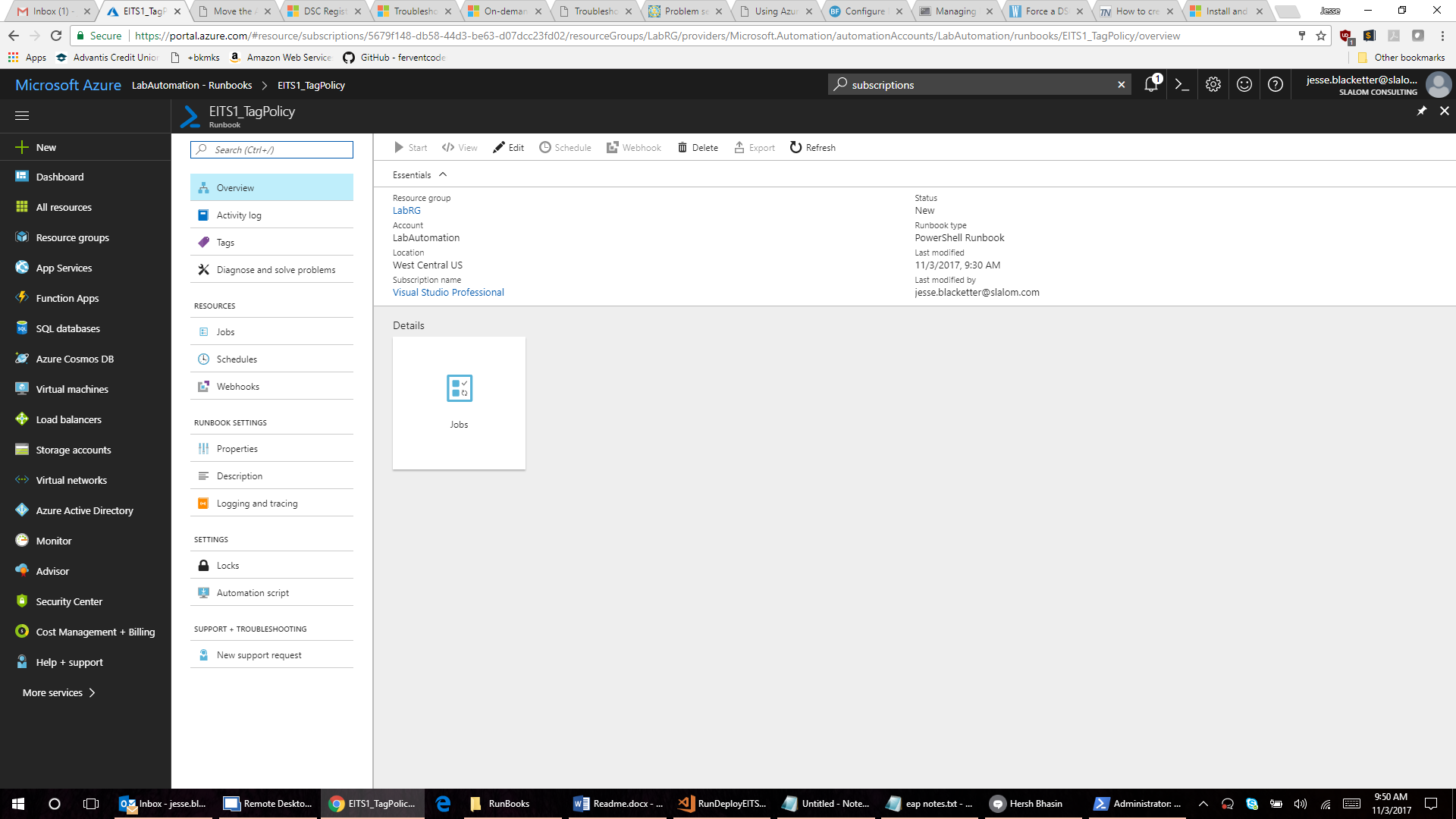




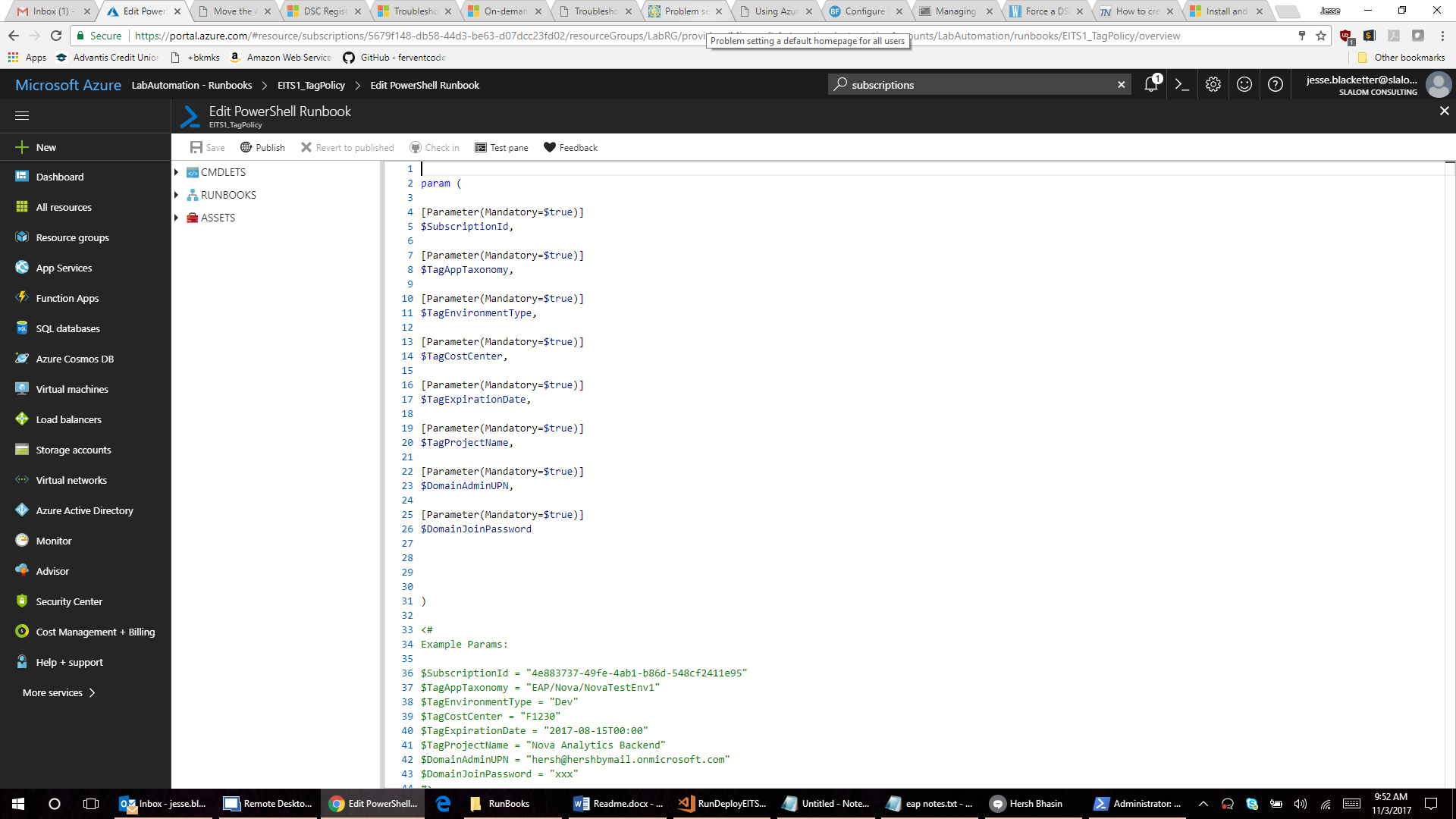


2.) The runbooks are numbered sequentially. There are 6 runbooks and they must be run in order.

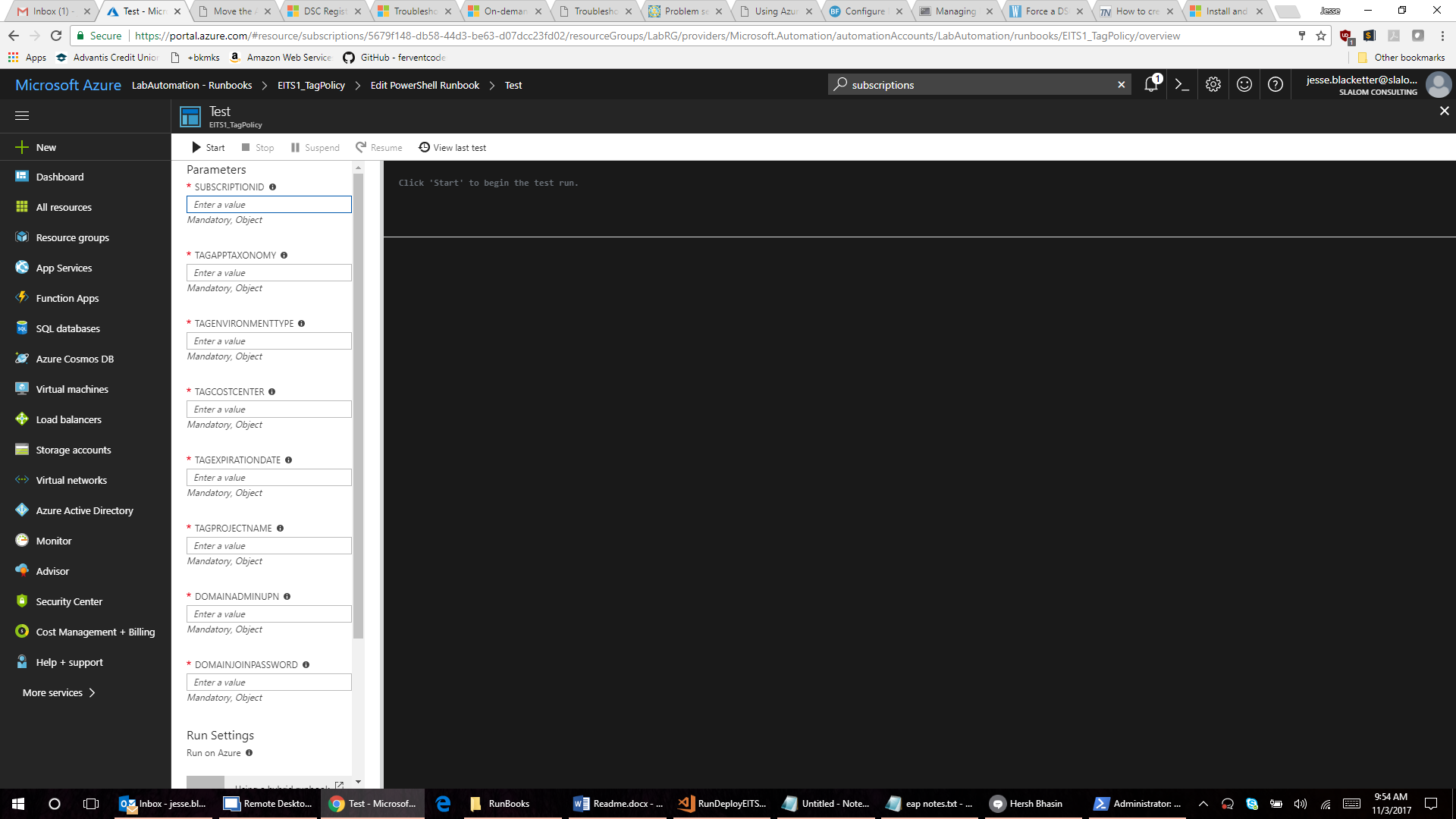
3.) Using EITS1\_TagPolicy as an example, click on that runbook and choose Edit.



4.) Now choose Test pane:



5.) Enter all required input for the parameters:



6.) When the parameters have been entered, you can click Start to execute the runbook.