Azure Monitoring Hackathon Deployment Guide

Version 1.0

Author: Rob Kuehfus

Please send feedback to here - <https://github.com/rkuehfus/AzureMonitoringHackathon>

Contents

[Getting Started 2](#_Toc530988600)

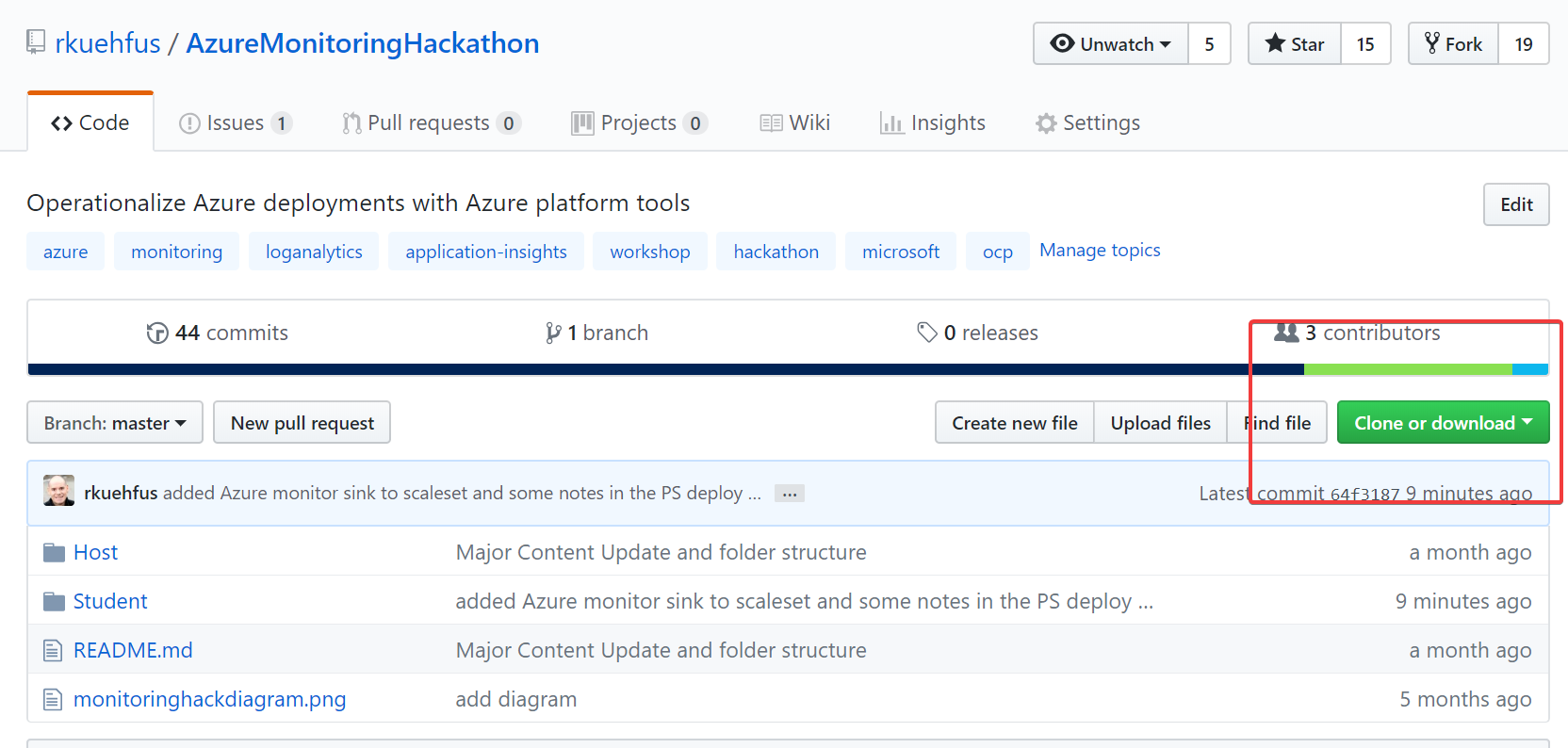
[Deployment Steps 4](#_Toc530988601)

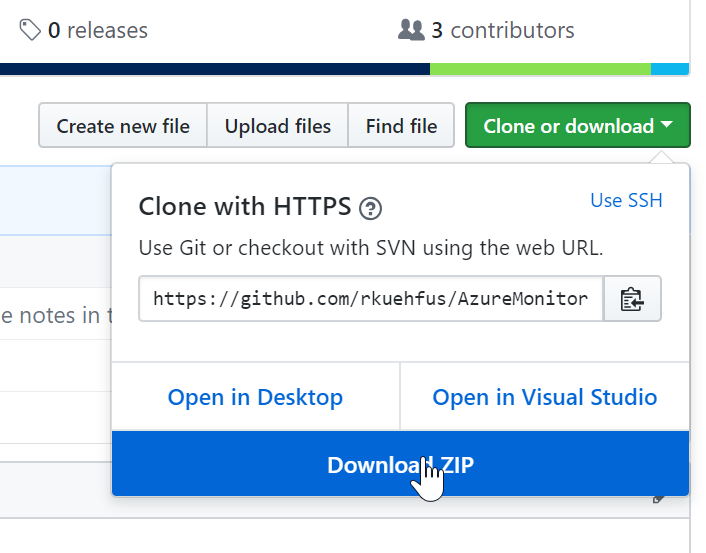
[Troubleshooting 7](#_Toc530988602)

[Validation 8](#_Toc530988603)

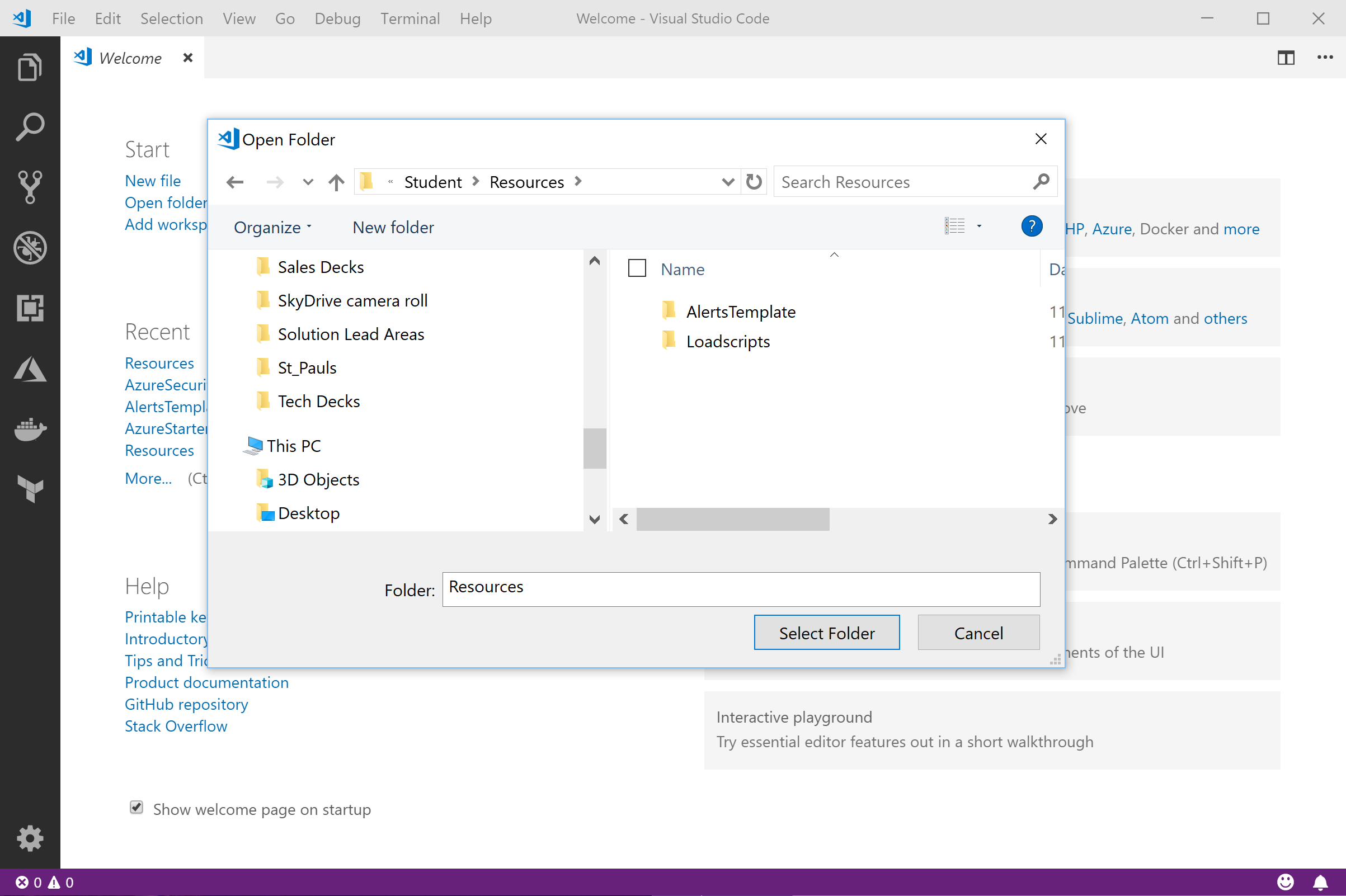
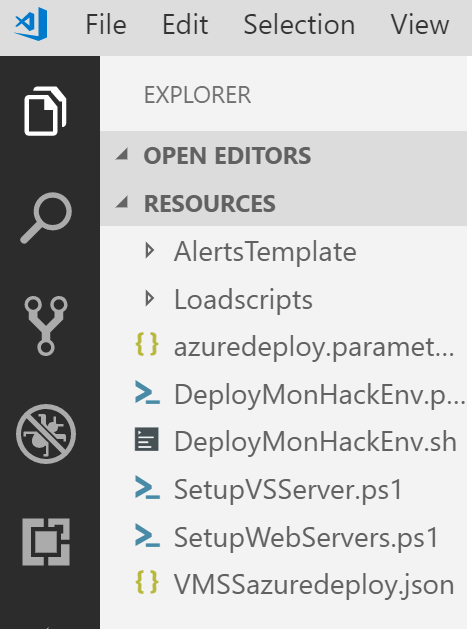
# Getting Started

Navigate to <https://github.com/rkuehfus/AzureMonitoringHackathon> and download the repository as a zip file to your local disk





Unzip the contents to a local folder on your machine.  


Open Visual Studio Code (download [here](https://code.visualstudio.com/download) if you do not have it installed), File 🡪Open Folder…  
  
Navigate to the location where you unzipped the files and open the **.\ AzureMonitoringHackathon-master\Student\Resources** folder  


From here, open either the **DeployMonHackEnv.ps1** or **DeployMonHackEnv.sh** depending on what you are most comfortable with. **Note:** The remainder of this guide will focus on the PowerShell deployment script, but you can easily translate this to the bash script.

From the DeployMonHackEnv.ps1, we need to modify a few items before working through the script.

**Note:** Make sure you have the latest Azure PowerShell Modules installed. If you do not, don’t worry just uncomment this line  
  
**Install-Module -Name AzureRM -Force -Scope CurrentUser -AllowClobber**  
  
Place your curser anywhere on the line and hit **F8** on your keyboard to run it. *No need to run as Administrator as the scope is set to CurrentUser*

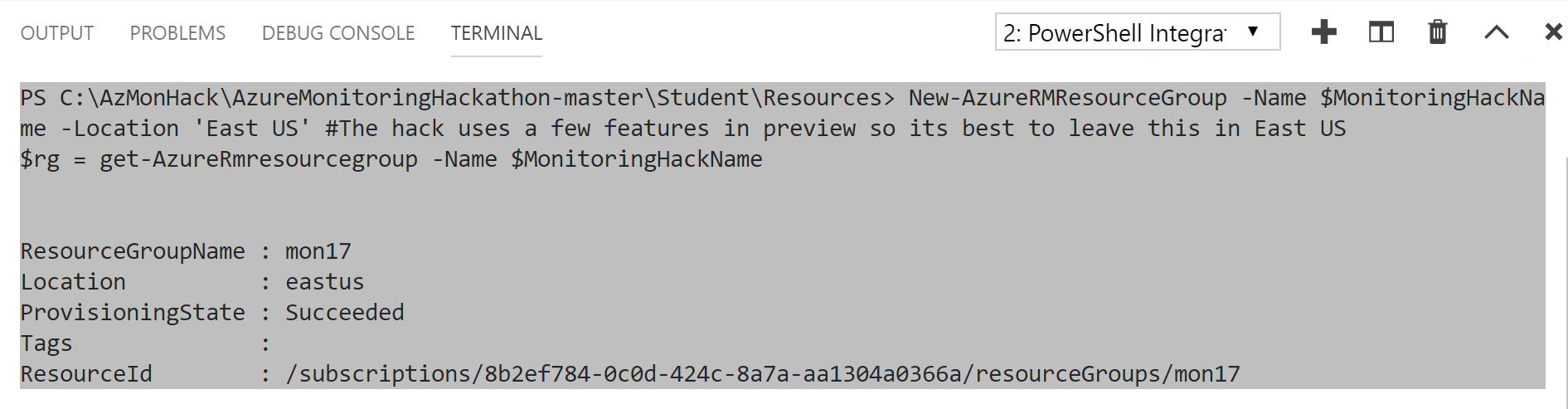
Now you are ready to proceed.

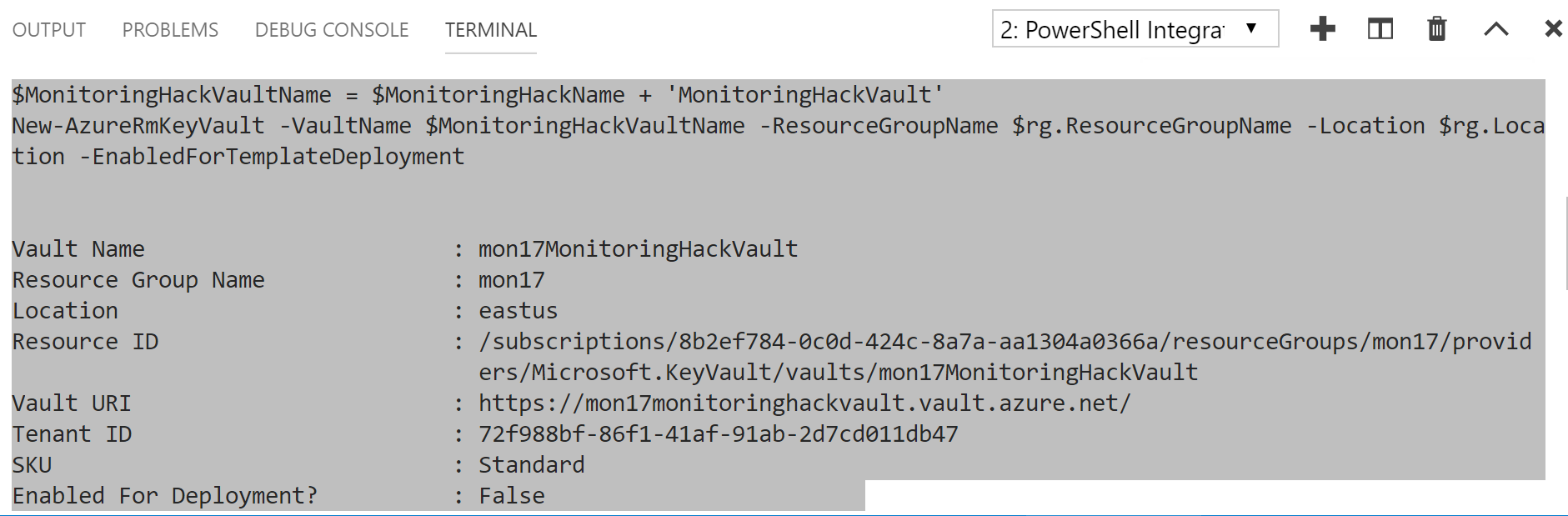
# Deployment Steps

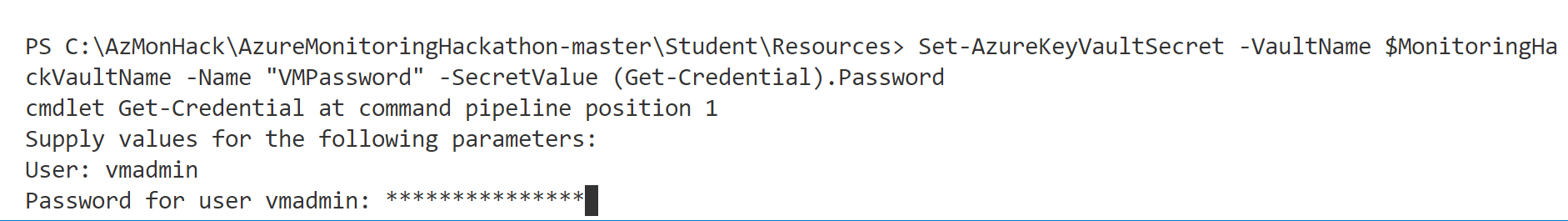
This is intended to be run one line at a time to place your curser on the line and hit **F8** as you work through each step.

**Step 1:** Update with your initials. Make sure to keep this to 5 characters and lowercase. The automation uses this in a few places including generating a storage account, so we need this to be consistent and to follow the rules. **Tip:** If you run into a deployment failure delete the resource group and change this to a different 5 characters as its possible someone else that deployed this hack used the same ones you did and left it running.

**Example:** $MonitoringHackName = 'mon17'

**Step 2:** If you are not yet connected to Azure from your PowerShell session run the **Connect-AzureRmAccount** line where you will be prompted for your credentials to connect to Azure. Make sure you are connecting to a subscription you have contributor access. If you are already connected skip this line and run the next two to create the resource group and set a variable (we will be using throughout the remainder of the deployment).   
  


**Step 3:** Run the two lines to create the Key Vault. This is uses to store the password used for creating each of the VMs and to access the database that is deployed. Remember we never hardcode passwords in ARM templates or PowerShell Scripts  
  


**Step 4:** Run this line and you will be prompted for a username and password. Use the username of “vmadmin” (if you do change this make sure to change it in the parameter file). Make sure your password adheres to the Azure password policy.  


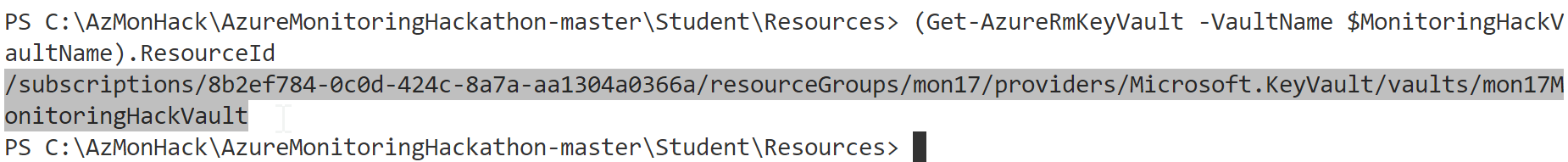
Passwords must be 12 - 123 characters in length and meet 3 out of the following 4 complexity requirements:

* Have lower characters
* Have upper characters
* Have a digit
* Have a special character (Regex match [\W\_])

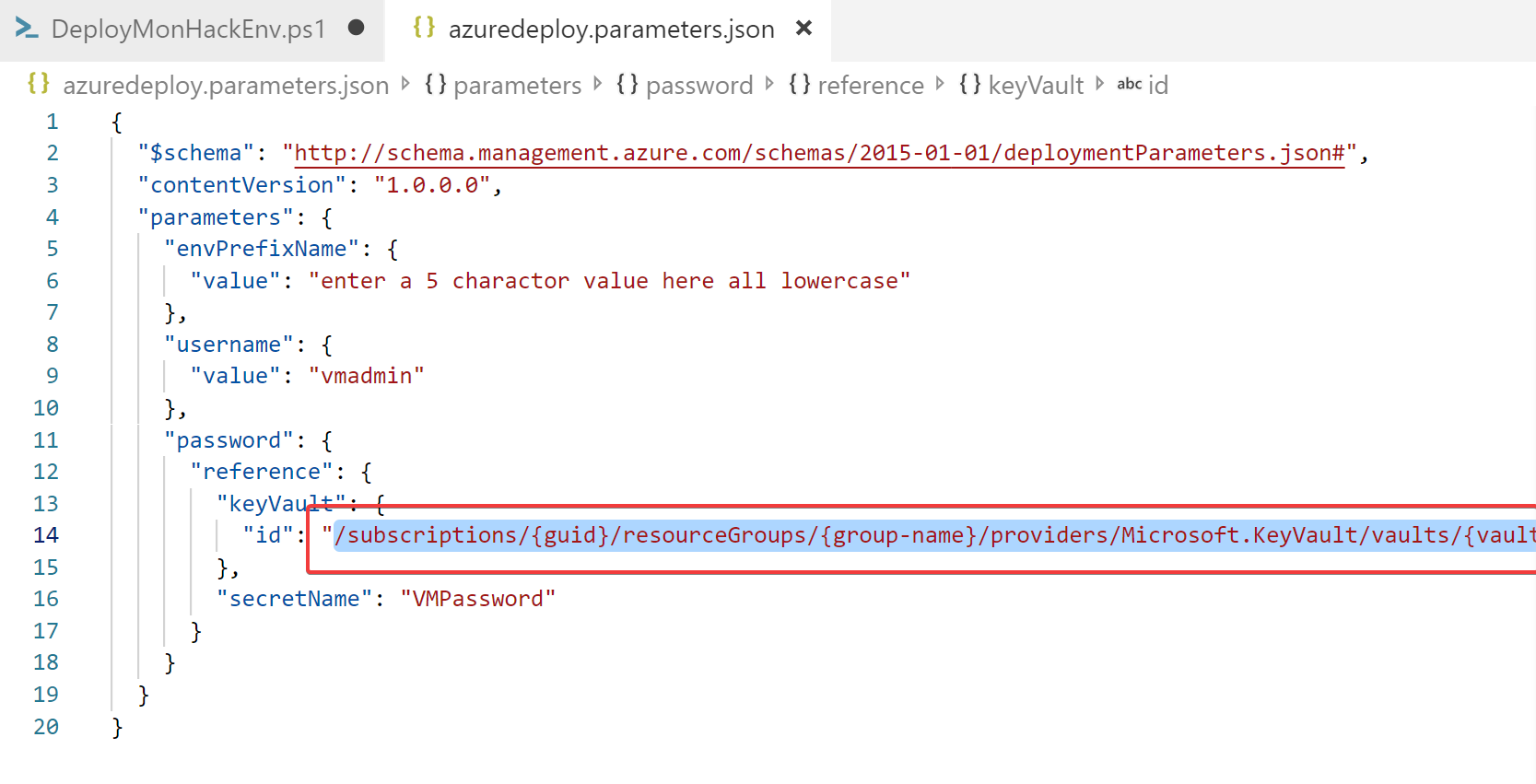
The following passwords are not allowed:

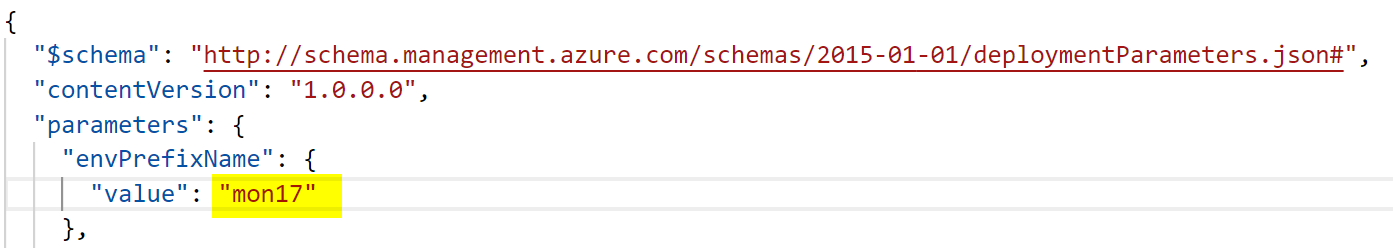
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abc@123 | iloveyou! | P@$$w0rd | P@ssw0rd | P@ssword123 |
| Pa$$word | pass@word1 | Password! | Password1 | Password22 |

**Step 5:** Run this line and copy the output results. Then paste this in the azuredeploy.parameters.json file under password.reference.keyVault.id.

Copy this -   


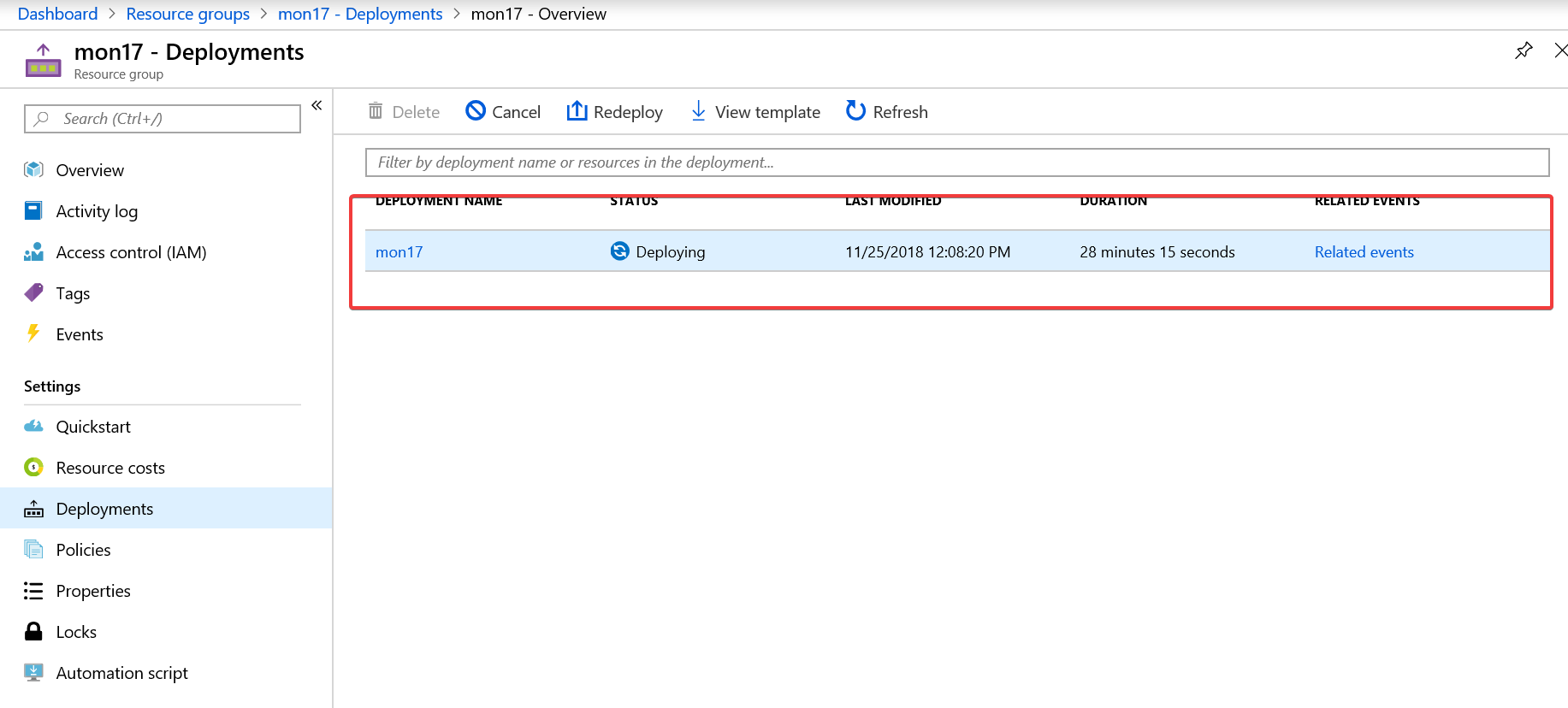
Paste here -



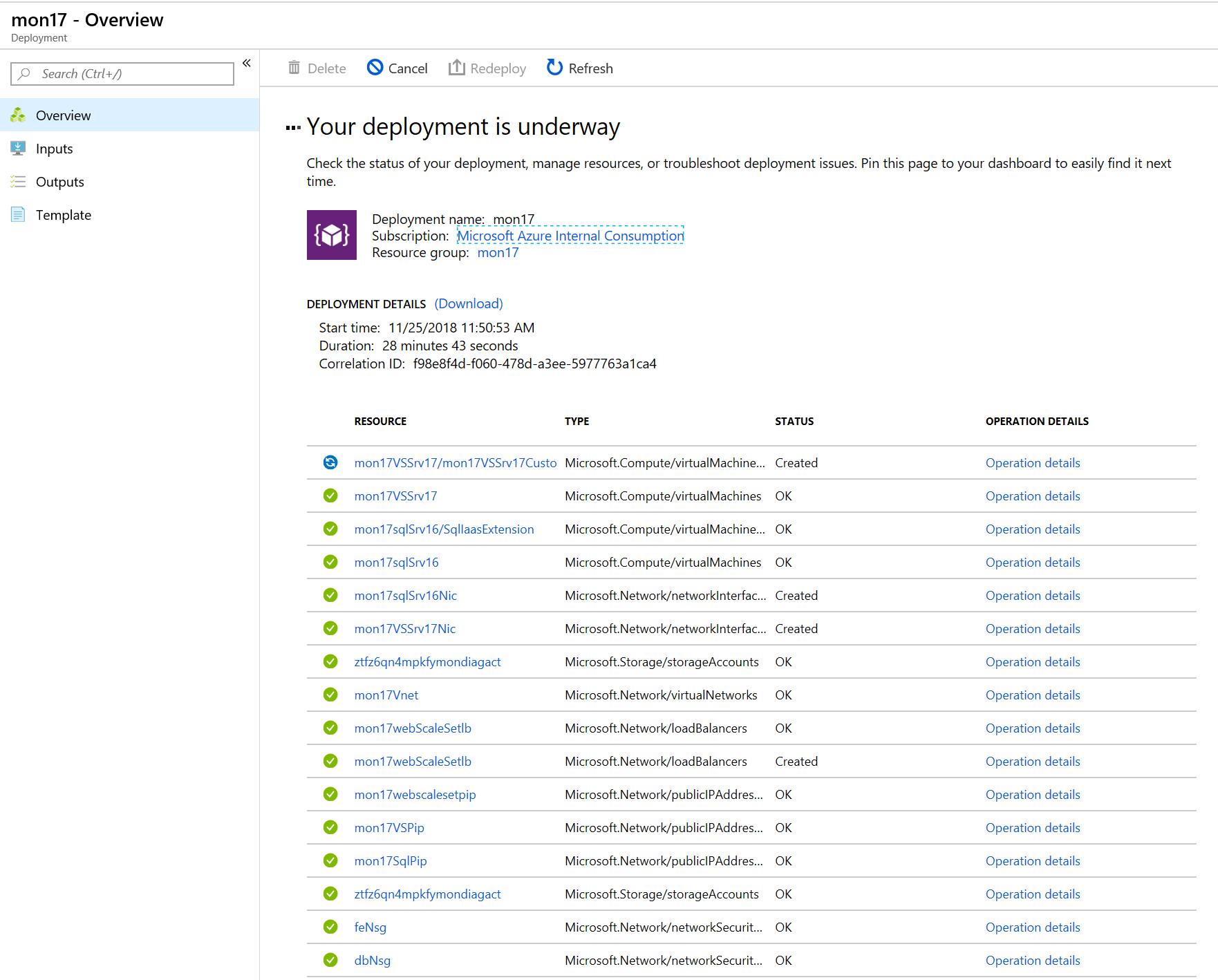
Also, make sure to update the envPrefixName to match what you specified in the DeployMonHackEnv file (yes, I know I could feed this in from the deployment script)  
  
  
**\*\*MAKE SURE TO SAVE THIS FILE BEFORE CONTINUING ON\*\***

**Step 6:** Run this line and away you go! Deployment takes around 42-45 mins due to dependencies.

You can check on the status of your deployment from the Azure Portal, by navigating to the resource group that was created and clicking on the Deployments blade.

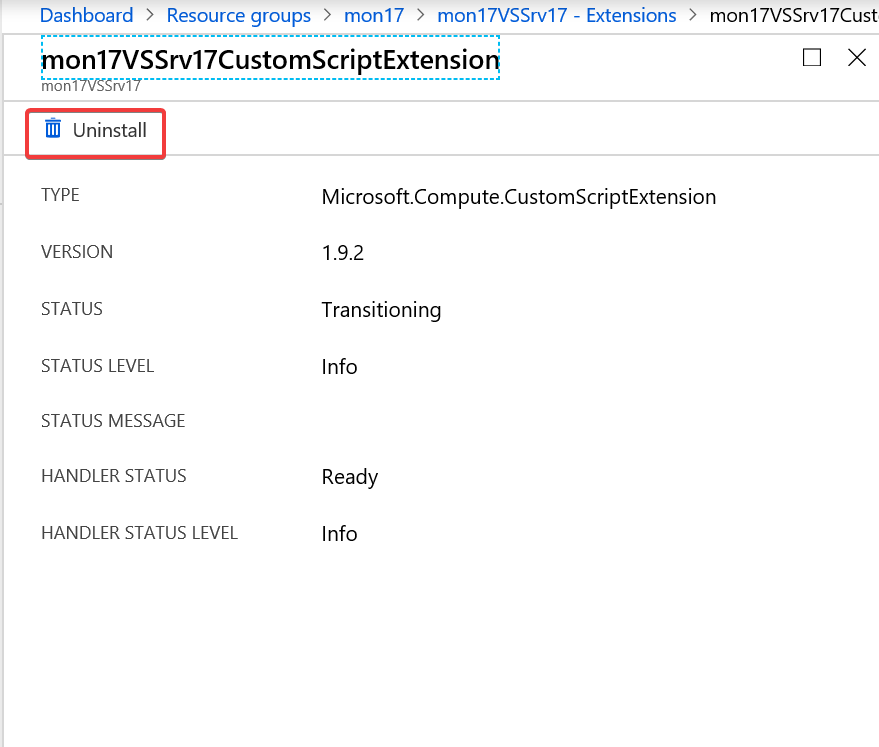


Click on the deployment name and check out the status.



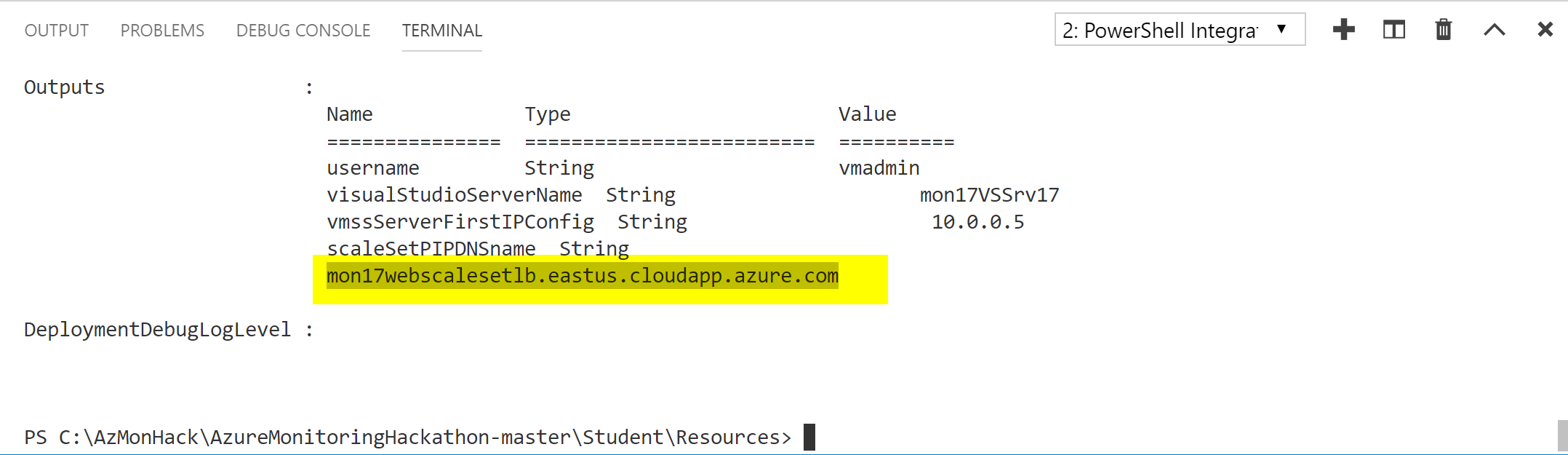
# Troubleshooting

* Make sure the 5-character name does not contain any uppercase letters
* Make sure the password used adheres to the [Azure password policy](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/faq)
* Make sure you are logged into the correct [subscription](https://blogs.msdn.microsoft.com/benjaminperkins/2017/08/02/how-to-set-azure-powershell-to-a-specific-azure-subscription/) and you have the at least contributors [role](https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles) access.
* Make sure you have the compute compacity in the region you are deploying to and request an increase to the [limit](https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits) if needed.
* Make sure you are using a region that supports the public preview for Azure Monitor for VMs - [link](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/vminsights-onboard)
* If you notice the deployment taking a long time

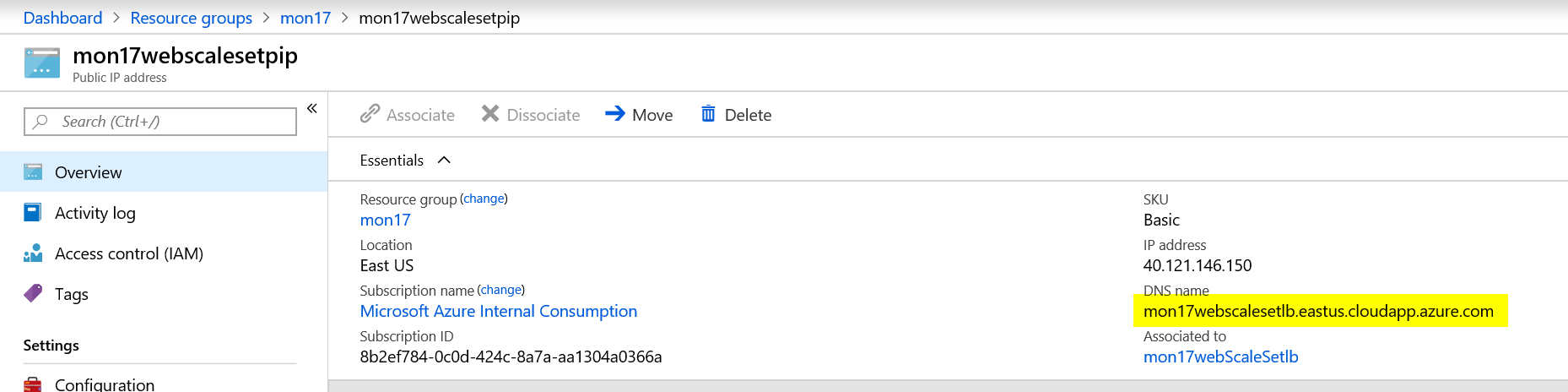
1. Look at the deployment details to figure out where it’s stuck
2. If you are stuck on the Visual Studio Custom Script extension (CSE)this is because the Microsoft Image was created with an older version of the CSE and has a bug.
   1. Workaround 1:The workaround has been to log on to the Visual Studio Server and navigate to “C:\Packages\Plugins\Microsoft.Compute.CustomScriptExtension\1.9.2” and double click on “enable” this will kick off the extension and the deployment should continue from here. If the script times out just rerun after you manually kick off the extension and it should finish
   2. Workaround 2: From the Azure Portal uninstall the CustomScriptExtension (which will fail your deployment).  
      
   3. Then rerun the ARM template and it will pick up where it left off.

Note: I will be opening a bug with the Product Group that owns the image.

# Validation

Once the script has completed in the Outputs section copy the scaleSetPIPDNSname and paste it in your browser.  


Or

Copy the DNS Name from the <5-char initials>webscalePIP resource in the Azure Portal  


You should render the eShop site

