|  |
| --- |
| PowerShell Boot Camp Survival Guide |
| June 15, 2015 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# Signing up and Connecting to Azure

Refreshers

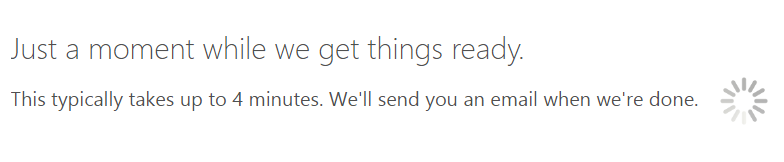
* While you do have to provide your credit card number to sign up, I promise that you will not be charged a single cent unless you click the button saying ‘Allow me to be billed’
* The VM class we’ll be using cannot possibly incur any charges, even if left on for a month. Do not worry!
* You’ll need the Azure PowerShell cmdlets. If you don’t see ‘Microsoft Azure powerShell’ in your start menu, install this <http://bit.ly/DownloadAzurePS>
* We’re going to connect to Azure by downloading a .PublishSettings file which we import into PowerShell

## Step One - Sign up For an Azure Trial, if you don't already have one

Make an account here <http://bit.ly/BootCampAzureSignUp>

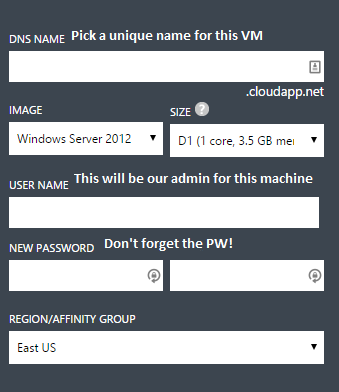
Follow the instructions and confirm you've provided a valid credit card (required to register, and only used if you decide let your VMs continue running after spending your free $200 credits)

Machine generated alternative text:
Microsoft Azure 
Payment information o 
Agreement 
COMPLETE 
to the agræ•æ-.t, offer deta'ls. and 
my email to Miwft 
We creatng Dc or ref-eh 

When you see this message, you're finished! Leave the window open in the background.

|  |  |
| --- | --- |
| My Azure Account Details | |
| USERNAME |  |
| PASSWORD |  |

## Step Two - Download your Azure Publish Settings file

1. Launch ‘Microsoft Azure PowerShell’. This is pretty much the normal console with a special .ps1 profile loaded that gives you some bonus tips about Azure. You could also just run Import-Module Azure
2. Run the cmdlet Get-AzurePublishSettingsFile
3. This will launch a web browser window and prompt you to sign in to Azure using your account details that we just created. Make sure to save the file when prompted, not open it.
4. Copy the .XML file to your C:\PowerShell directory
5. Now, use the CD command to change to the C:\PowerShell folder, then run Import-AzurePublishSettingsFile .\YourPublishSettingsFile.PublishSettings
6. When this is done, run Get-AzureSubscription to confirm that it all worked ☺

## Step Three – Create our VM

1. Go back to our Sign up window from before and click the green button that says ‘Start managing my Service’. Or use this URL if you closed the window https://manage.windowsazure.com
2. Click through the intro wizard, then Click Virtual Machines on the left, then **+ NEW** at the bottom.
3. Choose Compute -> Virtual Machine -> Quick Create
4. In the box that appears, fill out the following, and fill it in here so we don’t lose it!
5. When you’ve completed the new VM wizard, it will take a few minutes to spin up the VM, so this is a great bathroom break time.
6. When the VM status changes to ‘Running’ you can login to it by clicking Connect, which will download a .Rdp file.
7. When the login box appears, type in your logon name as VMName\UserName, so if you made a VM called PSBootCamp01 and the UserName of Jim, you’d logon with PSBootCamp01\Jim.
8. Click the Checkbox of ‘Don’t ask me again for connections to this computer’ when that box appears.
9. You’re now logged on to your first VM in Azure. Pretty cool, huh!?
10. You can close the session for now, we’ll come back to it later.
11. In PowerShell on your local PC, try running Get-AzureVM, get-AzureAccount and Get-AzureSubscription. Some of these values should seem familiar now ☺

# Configuring our test environment

Refreshers

* Look back to Chapter 7 if you forgot your Azure account info
* You can use Get-AzureVM to see if your VM is online or offline
* If you closed PowerShell, you might need to run the Import-PublishSettings file step again

1. Use Get-AzureVM to see if your VM is online or not. If it’s shutdown use, Start-AzureVM to fix that.
2. Look back to your info from Chapter 7 and then Connect to your VM again.
3. On your VM, launch PowerShell as an admin, and run ‘winrm QuickConfig’
4. Next, from PowerShell run this command : explorer $env:ProgramFiles\WindowsPowerShell\Modules
5. From your VM, open Internet Explorer and go to this URL http://bit.ly/BootCampDSC
6. Use the  button on the GitHub page and save to C:\PowerShell on your VM.
7. Copy the two folders in the ZIP to the Modules folder. You should have two folders, xActiveDirectory and xComputerManagement.
8. Double-click the BootCampMachineConfig.PS1 file
9. Hit F5 to run the Configuration Process. You’ll be prompted to provide a Domain Name.   
   **WRITE THIS DOWN IN THE BOX BELOW!** You’ll use this when you next connect to the VM, as we’re setting it up as a Domain Controller to have a safe place to do the rest of our lab work.  
   *Your domain must have a period in the name, and end in .test or .local*

|  |  |
| --- | --- |
| My Azure Account Details | |
| ComputerName *(something.cloudapp.net)* |  |
| First User Name/ Password (from ch7) |  |
| DomainName |  |
| Domain Admin UserName |  |
| Password |  |

1. You’ll be prompted next to provide a name for the first Domain Admin user. Make sure you double check that you’re spelling the name the way you want it, and triple check your password too. I’d even recommend copying and pasting your desired username and password from Notepad . Write them all down so you don’t forget it, or you’ll have to rebuild the machine later.
2. Watch as your machine is automagically rebuilt into a domain controller. We’ll explain how this worked later in the course. You’ll eventually see ‘Time to reboot the machine’ multiple times.
3. Reboot the machine, then hop over to Azure PowerShell on your local machine.
4. Use the Get-AzureVM cmdlet to check for the machine coming back online. When the Status goes ‘Stopped’ to ‘Running’ connect again
5. When you connect back to your VM (you can reuse the .rdp file you downloaded earlier), use your credentials you made in CH 7, but add your domain name. For instance, Foxdeploy.test\Stephen.
6. We’re going to mass create a bunch of users now.
7. Launch the PowerShell ISE as an Admin **ON YOUR AZURE VM** and open the C:\PowerShell\New-BootCampUsers.ps1
8. Break down the script line by line, and try to describe what each section is doing:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Real World Examples Below! Don’t be afraid to ask for help here ☺

Now that we have a for-real AD environment, let’s do some basic tasks.

1. Open up Active Directory Users and Computers from the start menu on your VM, and take a look around.
2. You’ll see that we have a few new Organizational Units (OUs) which are effectively folders for AD Objects. In AD, you use OUs to permanently store objects, normally based on their line of business or physical location.
3. When you need to group objects more flexibly, like for mailing lists or permissions, you use Groups.
4. Use your command finding skills to find a way to make new ADGroups, and make one for each of the OUs, Accounting, IT, Engineering, Marketing, called ‘All <Dept> Users’ .   
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Use your PowerShell skills to find a way to search for AD users based on their OU.   
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. Building on this cmdlet, add all of the users of each OU to the appropriate AD Group.  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
7. Try and make a new user called –FirstName Stephen –LastName Owen –SAMAccountName [Stephen.Owen@yourdomain.com](mailto:Stephen.Owen@yourdomain.com)   
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
8. Today I received an e-mail from your manager, informing me that I “was just not a good fit for your company, especially given my interesting dress code choices.” Use your PowerShell skills to disable my account.  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# PowerShell Remoting

Refreshers

* If you forgot your Azure VM information, double-check what you wrote down in the tables in chapter 7 and chapter 10
* In this section, we’re going to cover remoting into our Azure VM, so we can run PowerShell from our own desktop/laptop without needing to Remote Desktop into our VM anymore
* If you’re remoting for just a few commands, use Enter-PsSession to leave your PC behind and hop into a PowerShell window on a remote machine
* Store a session in a variable if you want to hop back and forth frequently.
  1. From PowerShell with the Azure module loaded, run Get-AzureVM. Is your VM Running? If not, start it up with Start-AzureVM.
  2. Use your get-command skills to find a command to Get Endpoint information about azure VMs
  3. Filter down so that the Name property is PowerShell and look for the Port number. Store that in $port
  4. Use that Port number to run the following command:

$cred = Get-Credential

Enter-PSSession -Comp yourVMname.cloudapp.net -Port $Port -Cred $cred -UseSSL

Did you run into an error saying ‘The certificate has the following errors…unknown authority’? That’s expected!

* 1. The error happened because we our PC doesn’t trust the Certificate from the remote machine. In order to trust it, we need to Open a web browser and download the cert.
  2. From PowerShell run START https://yourVMName.Cloudapp.net:$port. You should get a message /Chrome error saying ‘This connection may not be private’. That’s expected, your mission is to download this certificate, normally done by right clicking the SSL Lock icon in the address bar and choosing to ‘View Certificate info’ in Chrome, then choosing ‘Copy to File’ on the Details page.
  3. Choose the top option ‘DER Encoded Binary’ and save this as C:\PowerShell\AzureVM.cer.
  4. After that you’ll need to open the certificate and choose to install it under Current User > Trusted Root Certification Authorities. Press ‘Yes’ to continue and you’re done.
  5. This whole process was a one-time thing, from now on, you can just hop right in!
  6. Try remoting into your VM and see how it goes. Ask for help if you get stuck!
  7. We’ll be hopping into and out of the VM quite often, try $Session = New-PSSession and copy the parameters from #4
  8. Now that you’ve made a $session object, you can enter and exit it repeatedly using Enter-PsSession $session and then Exit-Pssession from the remote machine.
  9. How does the UI change to show that you’re in a remote machine? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  10. Use the Invoke-Command –ScriptBlock {“this command is running on $env:ComputerName”}
  11. Now try the cmdlet again and add –Session $session. Did your command run against the remote machine?

# Real World PowerScripting

Refreshers

* If you have a property like $user.Name > ‘Owen’, and want to select just part of the value, you can index into the value itself, e.g: $user.Name[0] > ‘O’, to select just certain letters

1. Open PowerShell

**Background:**  
It’s Friday afternoon, 4:30 PM and you’re about to leave to go out of town on vacation, you need to beat the traffic. Your boss walks in the room and drops a list of new users starting on Monday morning at 6:00AM!

1. You’ve just recoiled from the eleventh hour news that you have a whopping 20 new users starting on Monday! You’ve been given a usersnames.csv file that has the users first and last name only. You know that you need to create and provision a Windows user account for each of them.

**Task 1**  
Using your PowerShell scripting skills, connect to your VM then take the information in C:\PowerShell\usernames.csv to create a new account for these users using the appropriate AD cmdlet.   
  
Be certain to define the SAMAccountName for the user as FirstLetter.LastName. Also, fill out the GivenName/(FirstName), SurName(LastName) and Display Name properties too  
  
Show the teacher the output of your command and write out the command used here  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IMPORTANT** If you run into any errors like ‘The Specified Account Already Exists’, don’t worry about it for now

**Task 2**

Oh crud! You just noticed that some of the users had similar names, and your code wasn’t set up to handle this sort of situation.

Management has decreed that if a new user account is to be created which would conflict with an existing users name, the new user’s SAMAccount Name should be $FirstName.$lastName instead.   
  
*Delete the users you just created (either using PowerShell or by hand, using Active Directory Users and Computers)*

**Task 2 (Continued…)**

Using your PowerShell logic skills, add logic to your user creation Script, and this time, check to see if the name would conflict or not. If so, use your scripting knowledge to specify a different user name instead.  
  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 3**

Traffic is starting to back up outside, and oh crap! You just realized that the list also had a department column too! Dang, the users are all in the wrong places.

Using your PowerShell logic skills, iterated through the users however you like, either by deleting them and starting over from your initial code, or make a new script to check your existing users against the .csv and place them in the right place. If an OU doesn’t exist for a user, make the OU using New-ADOrganizationalUnit  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 4**

One last thing to do, then you can run out the door and go on vacation, ensure that the user accounts are all enabled for the new users, and give them all a password of PowerShellisCool!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_