Azure Notes

# Introduction

We’re going to use this document for holding information about neonKUBE and Azure that will likely turn into formal documentation when we ultimately release this thing to the public.

# Azure Links

[Azure Identity client library for .NET](https://docs.microsoft.com/en-us/dotnet/api/overview/azure/identity-readme?view=azure-dotnet)

# Terminology

Microsoft Azure is a popular cloud provider and may be used to host neonKUBE clusters. Azure cluster setup is currently performed via neon-cli. In the future setup, will be allowed from neonDESKTOP and perhaps eventually a cloud integrated service. Before we dive into describing how to deploy a cluster to Azure, let’s review some terminology:

Account Azure allows their users to manage one or more accounts with each account having their own subscriptions (billing) and cloud resources. For example, you may have access to a work account hosting resources for your company and another for your own personal off-work activities. Many Azure users will only have one default account.

Active DirectoryMicrosoft’s enterprise authentication/authorization solution which is deployed internally by a very large number of companies to manage their employee and application identities and is also integrated into products like Office365 for smaller companies.  
  
Azure includes an Active Directory (AD) implementation and one or more instances of an AD can be created for any given Azure account and these are used to control who is able to create and manage resources within Azure. Each AD instance in an account is known as a tenant.

Application Applications are registered with Azure to hold application properties like its name, home page link, password, and application specific secrets and certificates. Applications are created within an Azure AD tenant which is known as the application’s home tenant.

Azure-CLI Azure’s command line interface tool. This is named az and can be used to manage all Azure resources. Alternatively, Microsoft also publishes a large number of Azure related cmdlets integrated into Powershell that can be used as well. We’re going to use az for these instructions. You can find instructions on how to install the Azure-CLI here:  
  
<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest&tabs=azure-cli>

Service Principal This is essentially the username an instance of an Azure application uses to authenticate itself with Azure to gain access to resources. Service principals reference an application and are also managed within an AD tenant. One way to think about application vs. service principals is that the application defines global properties of the application including its password and service principals are used to assign resource management rights to individual application instances.  
  
So, each application instance will be associated with a service principal which is associated with an application. The service principal will be assigned specific Azure resource management rights which will ultimately be used by the application to obtain the API token that will be presented to the Azure REST API to control access to the account’s Azure resources.  
  
Service principals are identified by a UUID known as its service-principal-id.

SubscriptionAzure organizes cloud resources like VMs, storage, and networks, etc. into subscriptions. A subscription is simply a billing relationship. When you signup for and Azure account, a default subscription will be created for you that uses your credit card for billing (or your quota for free plans). For many users, the default subscription will be all you ever need. More advanced users may want to add multiple subscriptions to isolate production and test resources for billing purposes and perhaps also for security reasons.  
  
Azure subscriptions are identified by a UUID known as its subscription-id.

Resource Azure refers to everything that can be deployed within their cloud as resources. This includes IaaS things like virtual machines, virtual disks, storage, load balancers, and private networks, as well as SaaS things like CosmosDB, Redis, Azure, Data Lake; an ever-growing list of services.

Resource Group Azure allows individual resources to be organized into a named group. This is useful for helping track costs for a group of resources that collectively implement a larger application, controlling access to the resources, as well as making it easy to discard related resources by simply deleting the resource group.  
  
neonKUBE deploys each cluster within separate resource groups by default. This makes it easy to control access to these resources, track per cluster costs, and deleting the cluster is easy: simply delete its resource group in the Azure portal.  
  
neonKUBE also allows a cluster to be part of an existing resource group for more advanced scenarios.

Tenant An individual Active Directory instance within an Azure Account.

# Prerequisites

You’ll need to do some preparation before you’ll be able to deploy a neonKUBE cluster to Azure. This section will walk you through simplified steps to get up and running quickly. Advanced users may need to customize this process.

1. Password Manager: We highly recommend that you install and use a password manager to record your Azure credentials as well as the neonKUBE application related credentials we’ll be provisioning below. We currently use Dashlane ourselves but there are several alternatives that all look reasonable.  
     
   In leu of a password manager, you’ll need to write these credentials down on paper or to a file that you’ll keep secure some other way.
2. neonDESKTOP: Download and install the latest release of neonDESKTOP on your workstation from [here](https://github.com/nforgeio/neonKUBE). This includes the Neon-CLI.
3. Azure-CLI: Install the Azure-CLI from [here](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest&tabs=azure-cli) if you haven’t already done so. We’ve tested against v2.10.1 so we recommend that you upgrade if you have an older version. You can verify the installed version via:  
     
   az --version
4. Azure Account: You’ll need an Azure account if you don’t already have one. You can open a free account [here](https://azure.microsoft.com/en-us/free/). Note that each new Azure account will start out with a default subscription.  
     
   Save your account credentials to your password manager.
5. Azure Login: Login using the Azure-CLI via the command below. This will bring up a browser window where you’ll select and enter your Microsoft credentials:  
     
   az login
6. Identify the Target Subscription: We’re going to need to know the subscription UUID where the neonKUBE cluster will be deployed. You’ll use the az account list command. This command actually returns an array of JSON documents that describe the subscriptions the currently logged in account can access (this command seems misnamed to me; it seems that something like   
   az account list-subscriptions would make more sense but what can you do?).  
     
   Execute the command below to list your subscriptions:  
     
   az account list  
     
   The command will return something like:

[  
 {

"cloudName": "AzureCloud",

"homeTenantId": "7dc1b8ce-8aa7-4e02-a801-01ea2f24c717",

"id": "d4bb8764-cd58-40ab-946e-f4273b4f8e33",

"isDefault": true,

"managedByTenants": [],

"name": "MySubscription",

"state": "Enabled",

"tenantId": "7dc1b8ce-8aa7-4e02-a801-01ea2f24c717",

"user": {

"name": "user@myemail.com",

"type": "user"

}

}

]

Most users will see only one subscription listed. If you see more than one, you’ll need to identify the one you want to use. The id property (d4bb8764-cd58-40ab-946e-f4273b4f8e33 in this example) is the subscription ID. Make a note of the subscriptionId and tenantId in your password manager; we’ll need those below.

1. Create neonKUBE app: This Azure application will specify the password Neon-CLI and neonDESKTOP will use to authenticate with Azure to obtain the API token these tools will use to create and manage the Azure resources underpinning neonKUBE clusters. First you need to generate a secure password. Password managers can do this or you can use Neon-CLI:  
     
   neon password generate  
     
   Make a note of this password and save it in your password manager as appPassword. Now execute the command to create the neonKUBE app, passing the password you just generated:  
     
   az ad app create --display-name neonKUBE --password *<appPassword>*  
     
   This will return JSON describing the new Azure app. Look for the appId property and add that to your password manager notes.
2. Create neonKUBE app service principal: The service principal is the place where the necessary Azure access rights will be assigned to neonCLI and neonDESKTOP. Run the command below to create new service principal (passing the new appId just created above):  
     
   az ad sp create --id <appId>  
     
   The command will return JSON describing the new service principal. Look for the objectId property and add that to your password manager notes as the servicePrincipalId.
3. Grant service principal access rights: The command below grants the neonKUBE app rights to create and manage most cloud resources. You’ll need to pass the servicePrincipalId and subscriptionId   
     
   az role assignment create --assignee *<servicePrincipalId>* --role Contributor --subscription *<subscriptionId>*  
     
   This should work fine for people trying neonKUBE out on a dedicated subscription but more advanced users and enterprises will probably want to restrict these rights to one or more Azure resource groups to absolutely prevent the app from messing with resources that don’t belong to it.

Now that these prerequisites are behind us, you’ll be able to create and manage one or more neonKUBE clusters within your subscription. You’ll need to repeat these steps to enable neonKUBE for other subscriptions.

Note that Neon-CLI and neonDESKTOP will require the subscriptionId, tenantId, appId, and app appPassword values identified above to provision and manage a neonKUBE cluster.