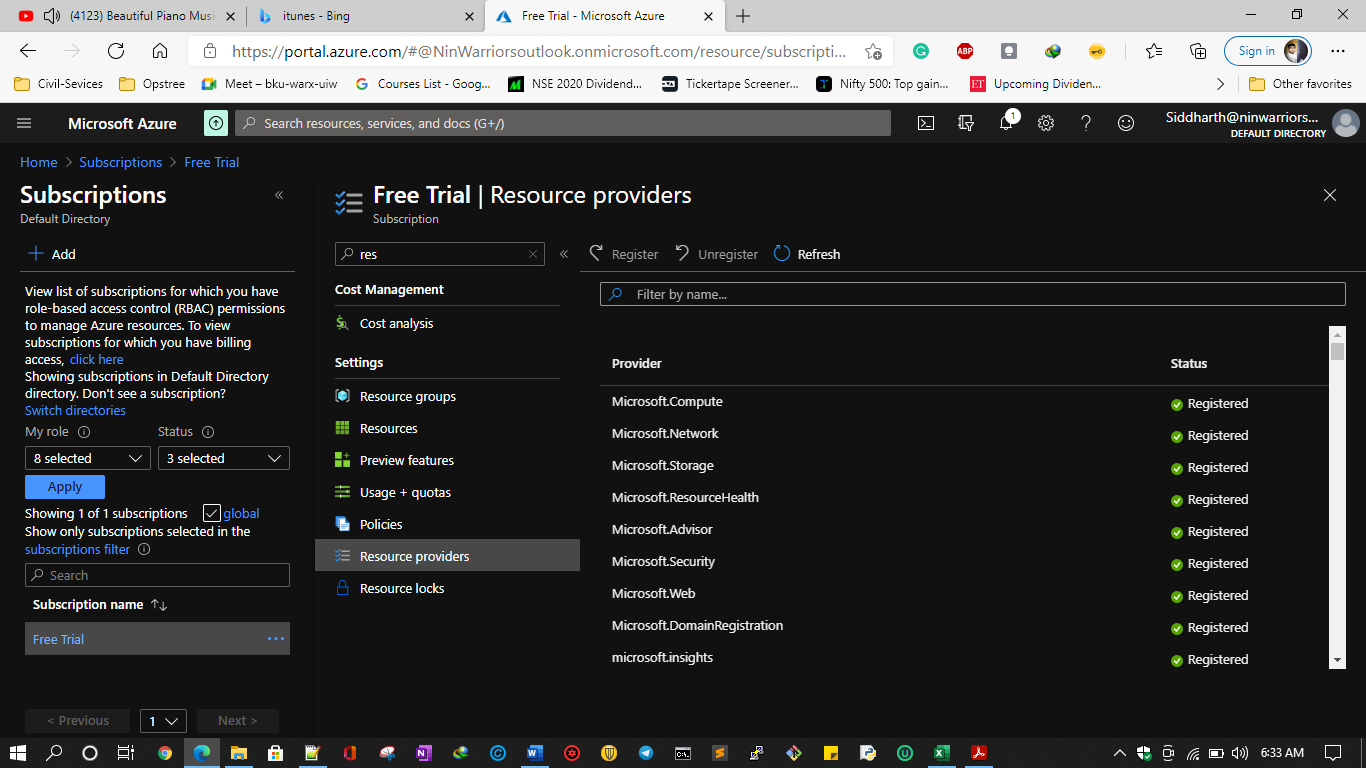
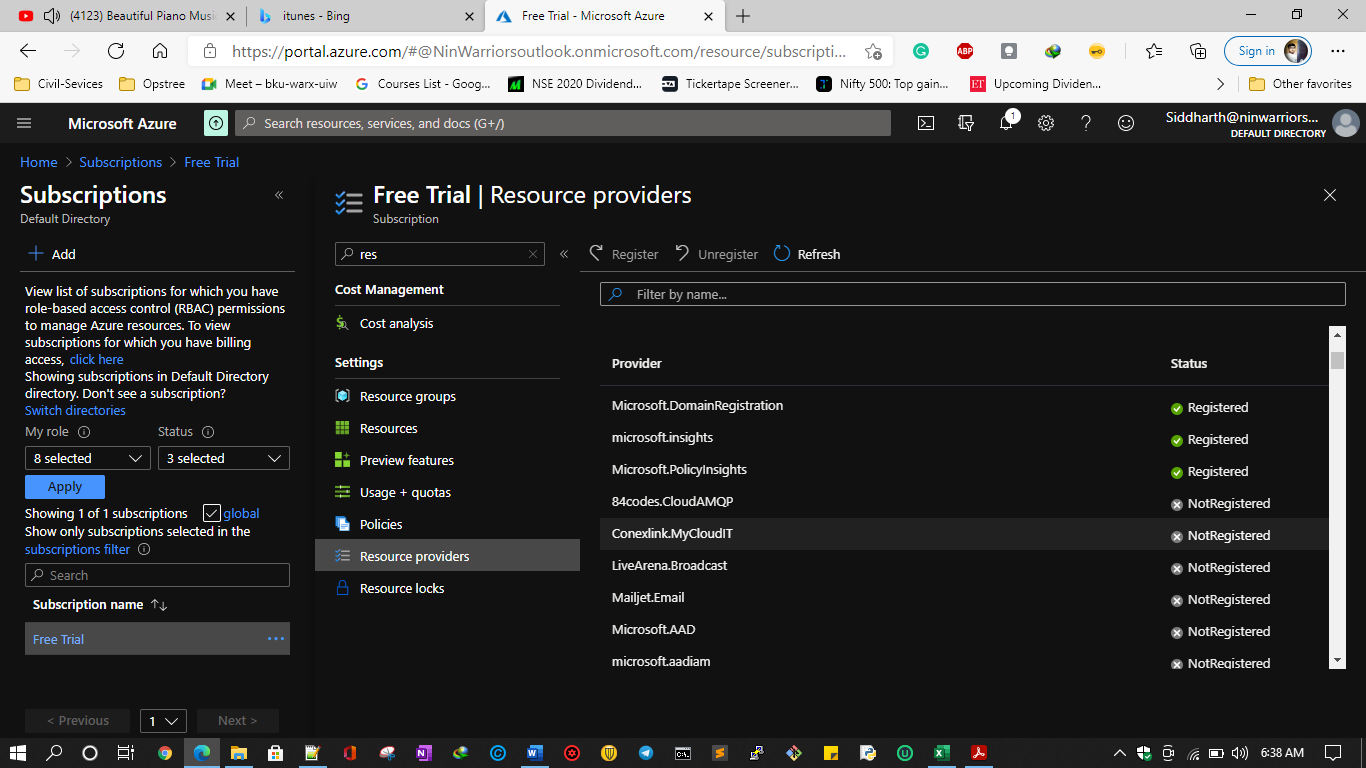
Doc: Microsoft Azure

Resources available with the current subscription:

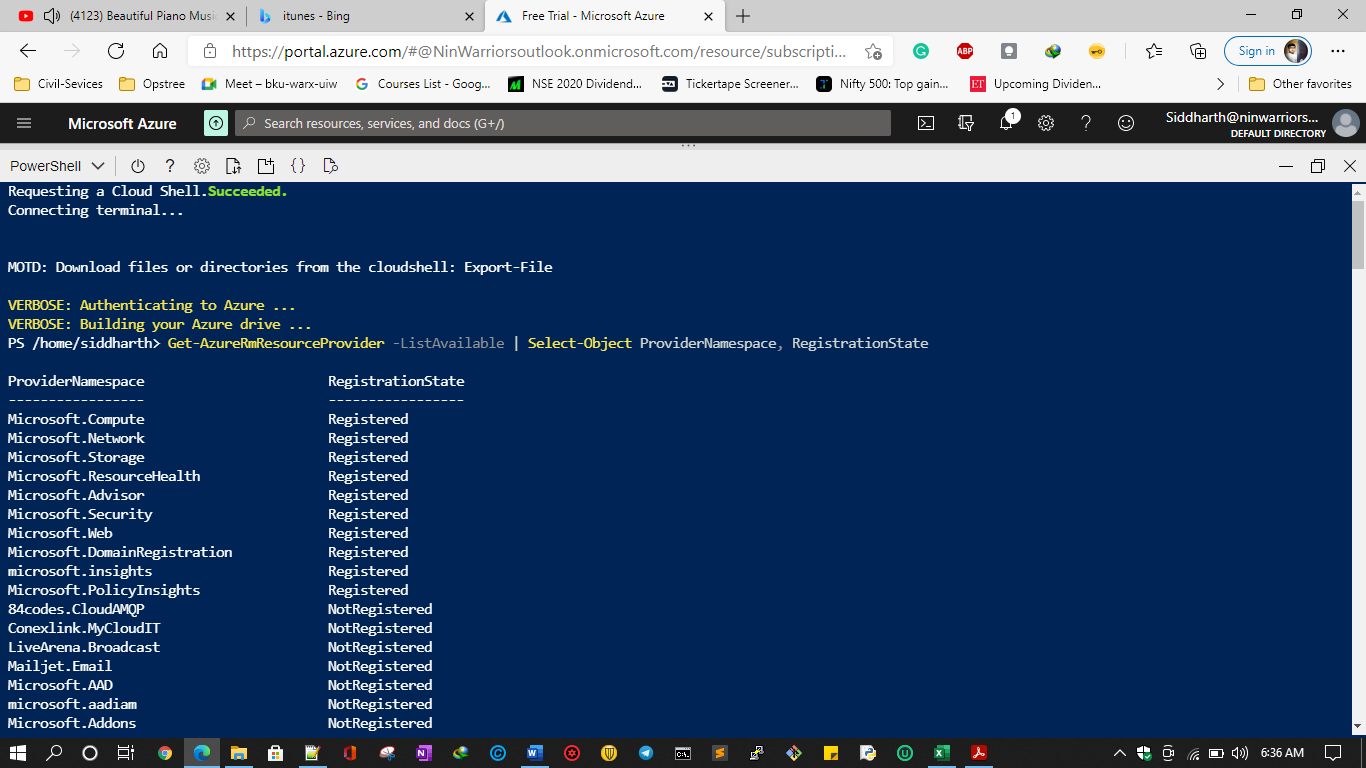
Navigation: *Subscription -> Resource Providers*

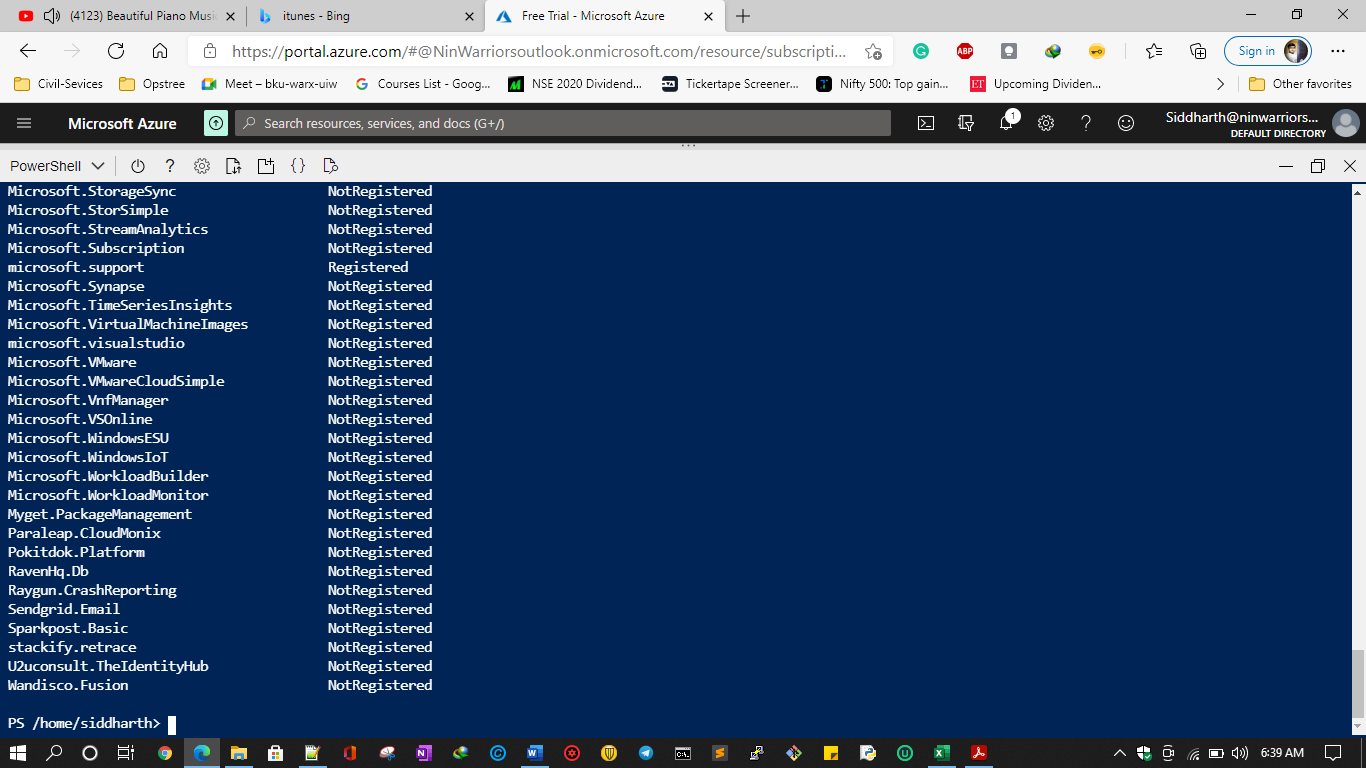




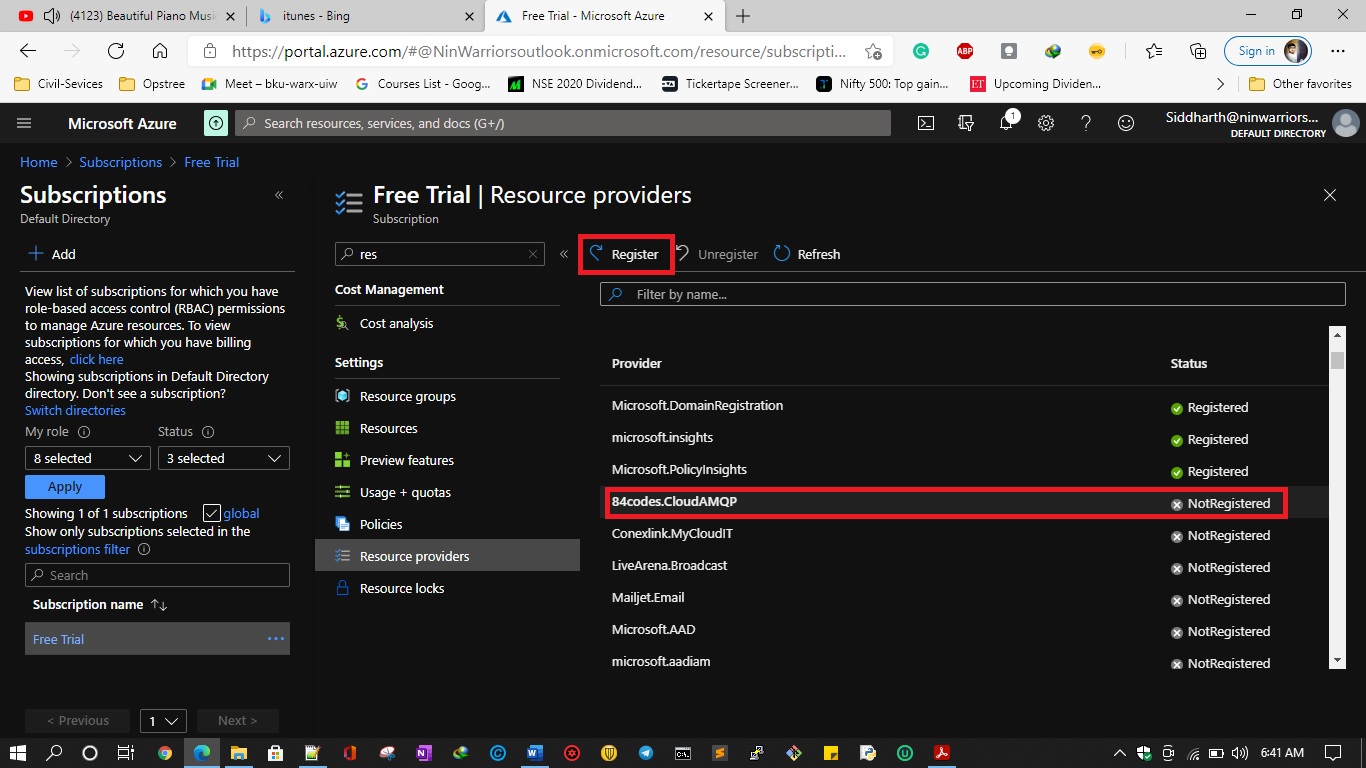
Command Let in PowerShell

*Get-AzureRmResourceProvider -ListAvailable | Select-Object ProviderNamespace, RegistrationState*





Register/Unregister a service



Equivalent Command let:

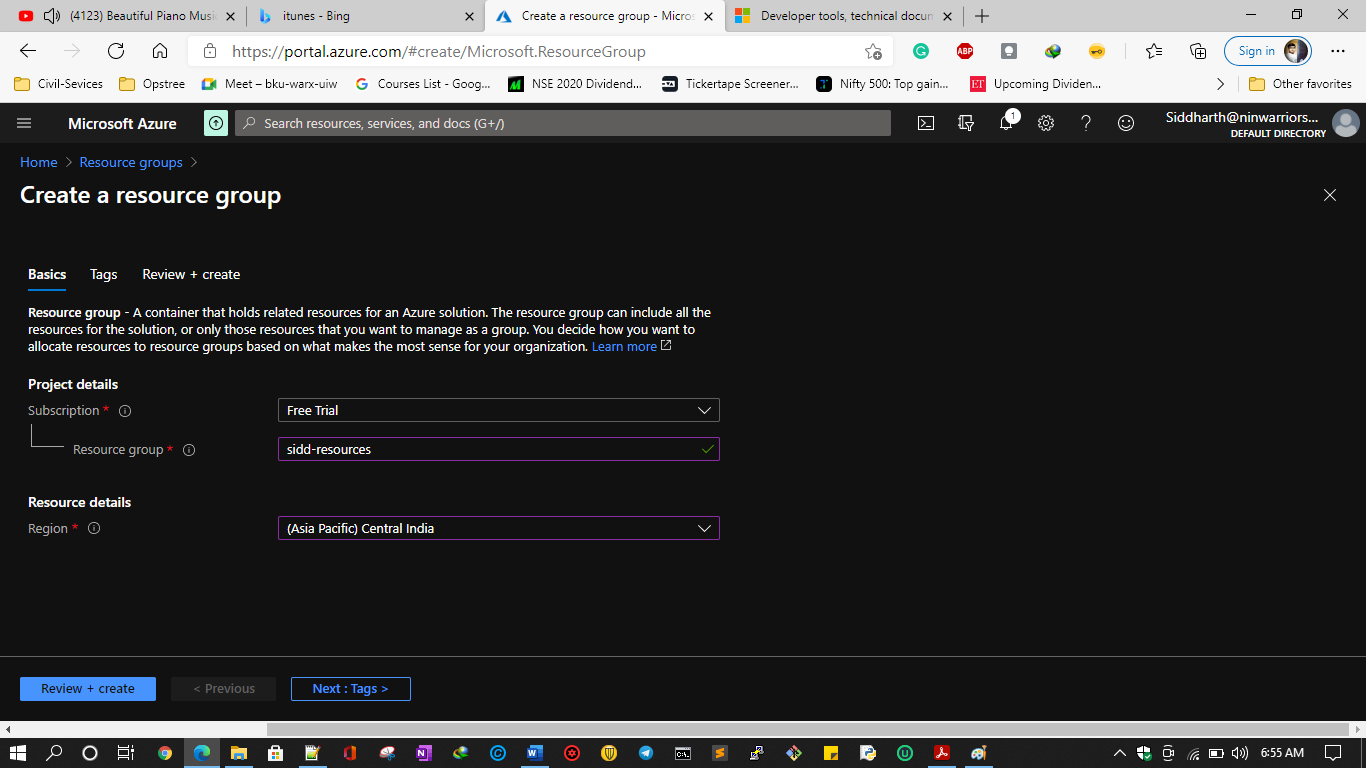
* For Registration  
  *Register-AzResourceProvider -ProviderNamespace Microsoft.BotService*
* For Un-registration  
  *Unregister-AzResourceProvider -ProviderNamespace Microsoft.BotService*

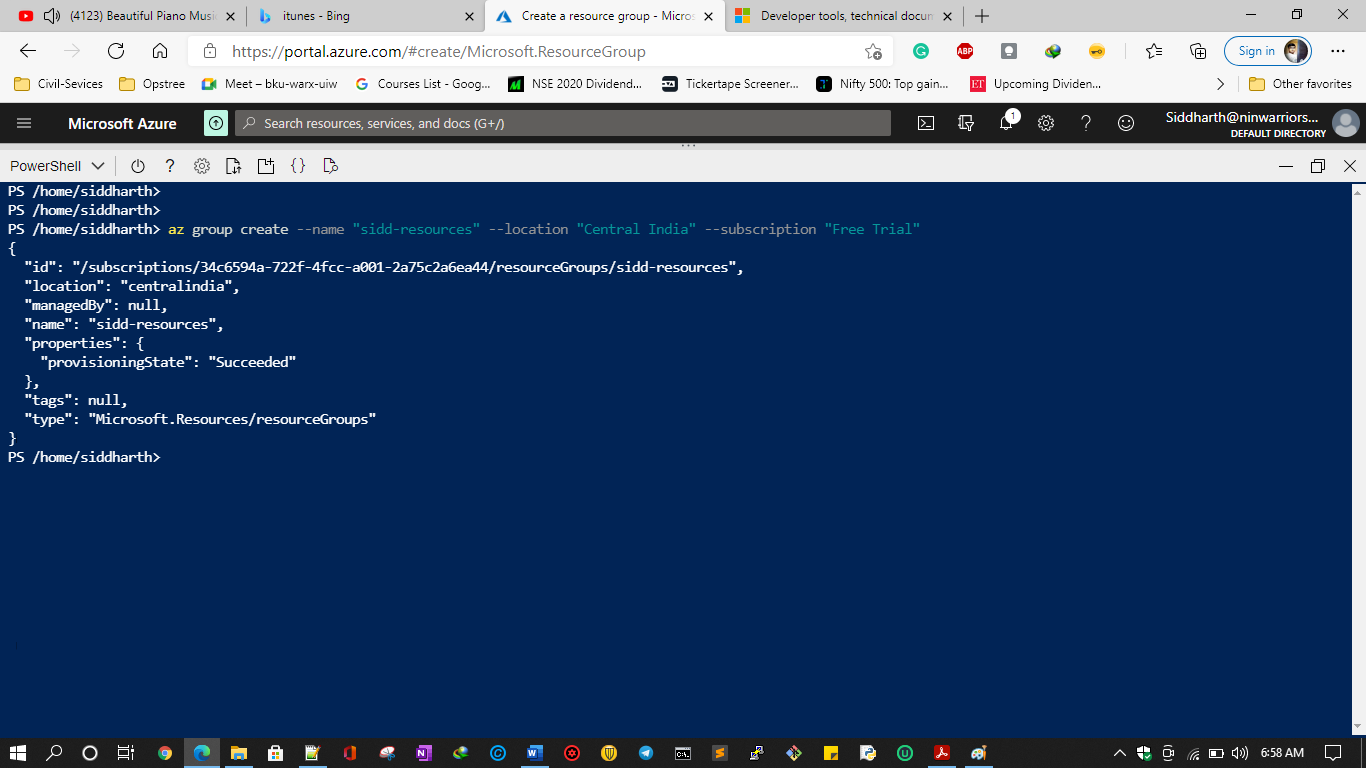
Resource providers are a simple way of separating your subscription from different kinds of resources and limiting access to them. If you are interested in how things really work, go to <https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview>, where you will find an overview of ARM.

Resource Groups:

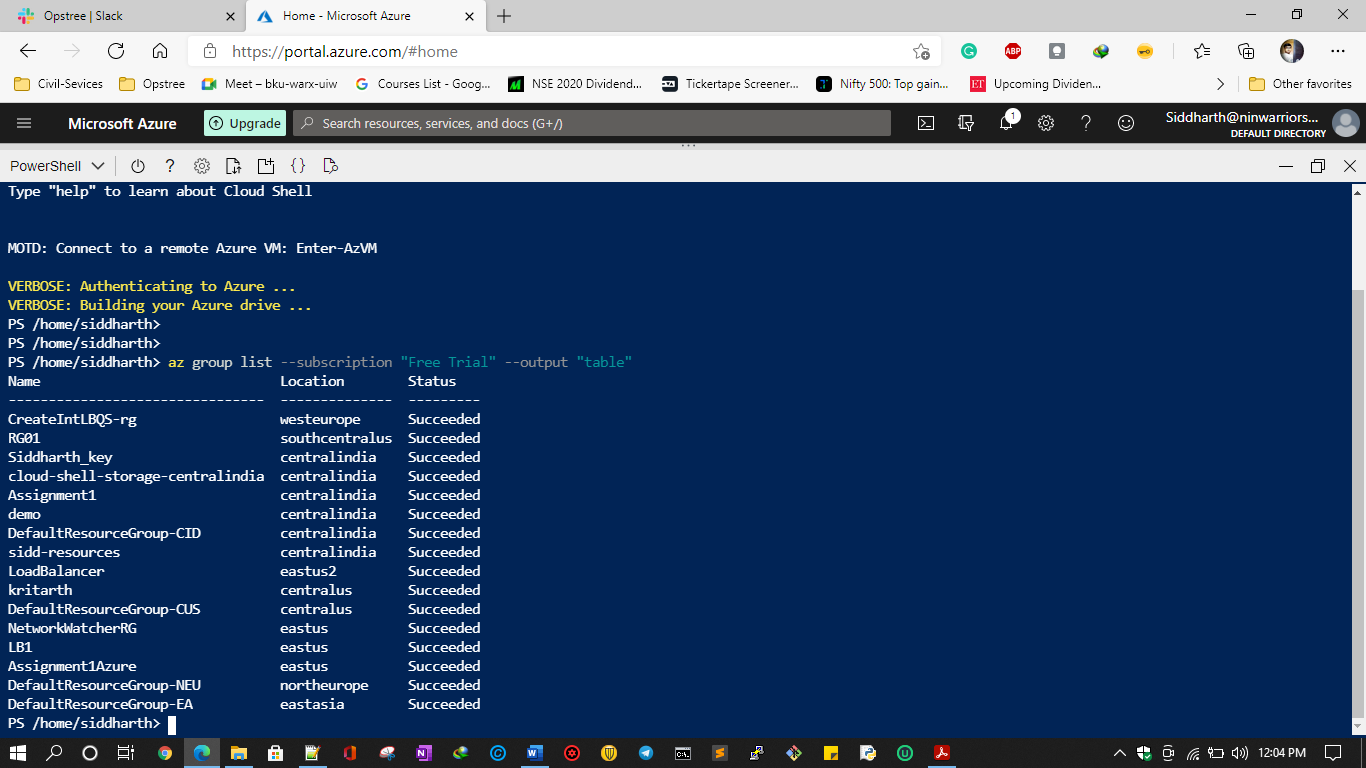
A resource group is the main logical component when it comes to governing resources provisioned in Azure. You cannot create a service without selecting one.  
To get started, you'll need a resource group. Creating one is one of the easiest operations in Azure – you can use the Azure portal, PowerShell cmdlet, or Azure CLI for this. If you prefer using the graphics interface, search for the Resource group term in the marketplace and click on the Create button. You will see a really simple form where you only need to provide two things:

* Resource group name
* Resource group location

The following is an example configuration for my resource group:  


CLI Command Let:  
*az group create --name "sidd-resources" --location "Central India" --subscription "Free Trial"*  


Browsing Resources   
*az group list --subscription "Free Trial" --output "table"*



ARM Template

One of the most important features of any computer system is the ability to introduce changes to it seamlessly and in an automatic fashion. When infrastructure is considered, it is especially handy if there is a way to code it. This enables us to replicate it anywhere at any time with a single command. This approach, often abbreviated as Infrastructure-as-Code (IaC), plays a major role in modern applications as it guarantees that all the components are scripted and that no manual work is required to restore them (for example, in the case of disaster recovery).  
This section addresses this problem by introducing Azure Resource Manager (ARM) templates, which are JSON files that are used to describe how a service should behave and be configured. There are multiple ways to prepare an ARM template, which can be reused and modified anytime you want. Before we get started, you have to understand how a typical template is structured.

Let's take a look at the following JSON file:

*{*

*"$schema":*

*"https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.*

*json#",*

*"contentVersion": "",*

*"parameters": { },*

*"variables": { },*

*"functions": [ ],*

*"resources": [ ],*

*"outputs": { }*

*}*

ARM templates are not the only way to manage resources via ARM. There are external tools that you can consider if you find this particular feature cumbersome or counterproductive. These tools are as follows:

* Azure Fluent: A set of helpful SDKs that enable you to programmatically call ARM APIs to provision resources. They are written for multiple different platforms (.NET, Java, Python, or Node.js).
* Terraform: A tool by HashiCorp where you use a YAML file to describe your infrastructure.
* Pulumi: A new project where you can use TypeScript, Go, or Python to write scripts that describe your infrastructure.

VNet (Virtual Network) Pairing

In cloud infrastructures, you do not have access to the physical layer of networking—instead, you can use software-based routers to help you segment your network. The resulting networks do not exist physically, hence the name virtual networks. There are moments when you need to connect two isolated VNets.

This gives you many crucial benefits—you can treat the traffic inside the networks as if it was a single ecosystem. This way, you can preserve the privacy inside the networks and achieve a low-latency and high-bandwidth connection thanks to your use of the Azure infrastructure as the backbone.

Created two Vnets in two separate geographical locations but in the same resource group:

