Chaincore Developer Edition

Contents

[Prerequisites 2](#_Toc482718854)

[Adding your object ID to the key vault 2](#_Toc482718855)

[Accessing the values from the key vault 6](#_Toc482718856)

[Single Generator, Multi Signer, Single Region 7](#_Toc482718857)

[Architecture Diagram 8](#_Toc482718858)

[Description 8](#_Toc482718859)

[Dataflow 8](#_Toc482718860)

[Single Generator, Multi Signer, Multi Region 9](#_Toc482718861)

[Architecture 9](#_Toc482718862)

[Description 9](#_Toc482718863)

[Dataflow 10](#_Toc482718864)

[Post Deployment 10](#_Toc482718865)

# Prerequisites

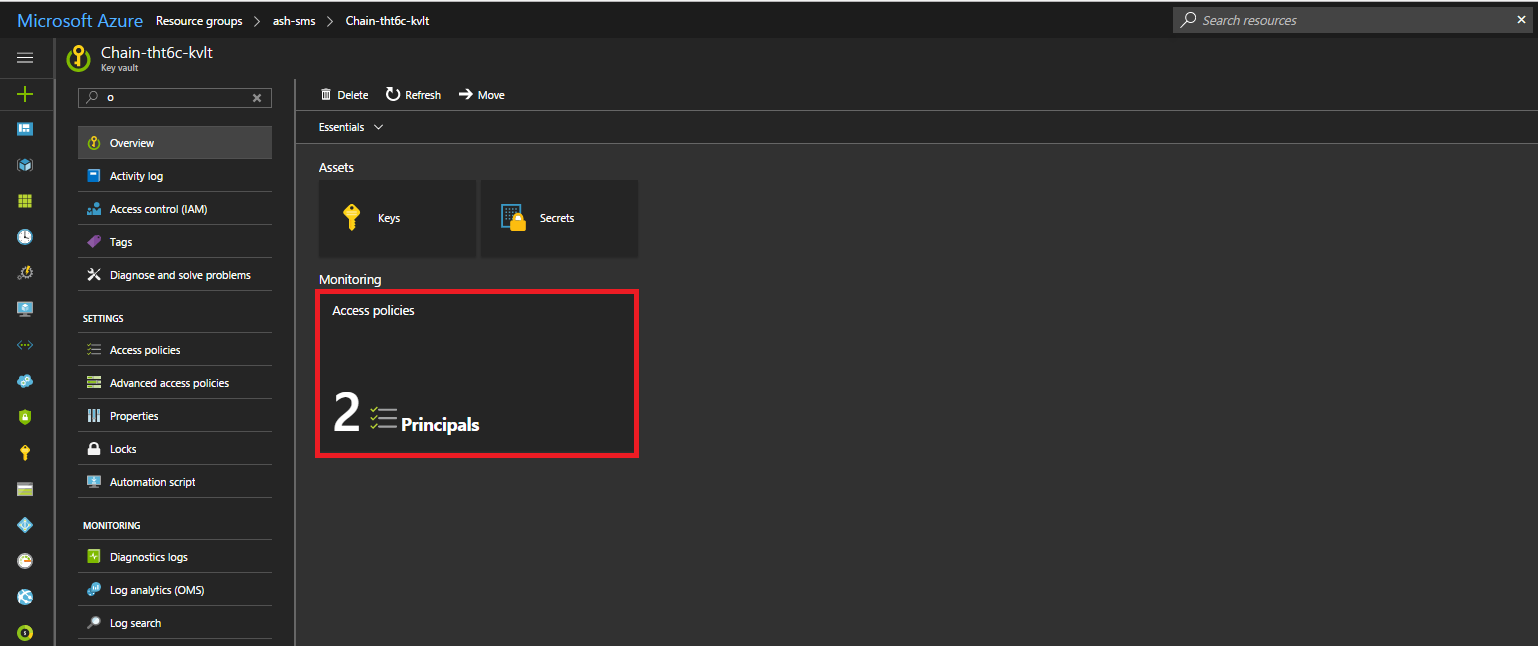
The following ARM templates deployment require few mandate prerequisites.

1. You need a service principal (ie Application ID) and its object ID to access the Key vault which is deployed as part of this ARM template.
2. If you do not have an existing service principal, you can create one by following this document [link](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal).
3. The object ID, is required in the access policies section of the key vault. This grants permission to that object ID to view the secrets and keys stored in the Access policies

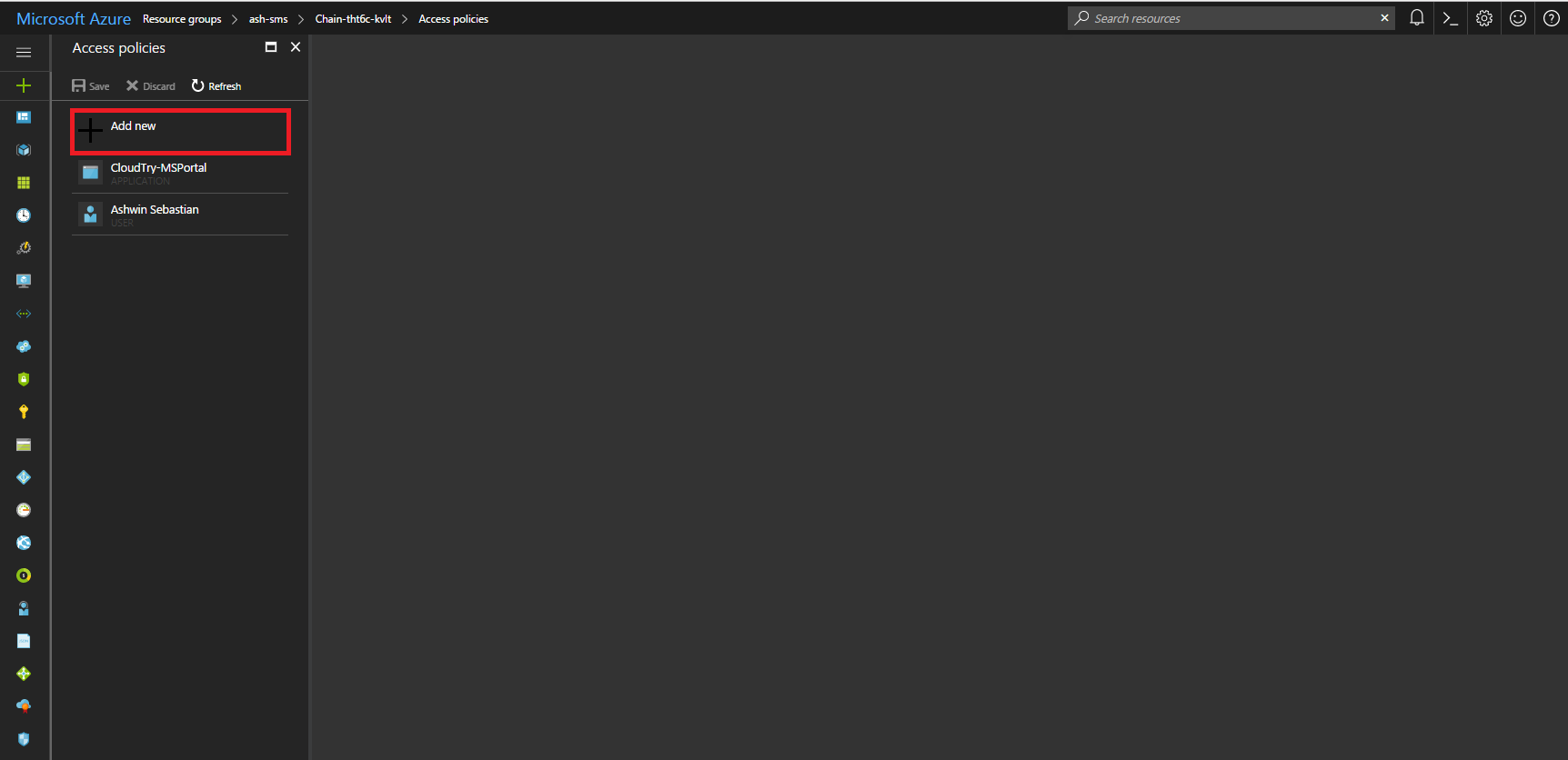
## Adding your object ID to the key vault

Steps to add your object ID to key vault

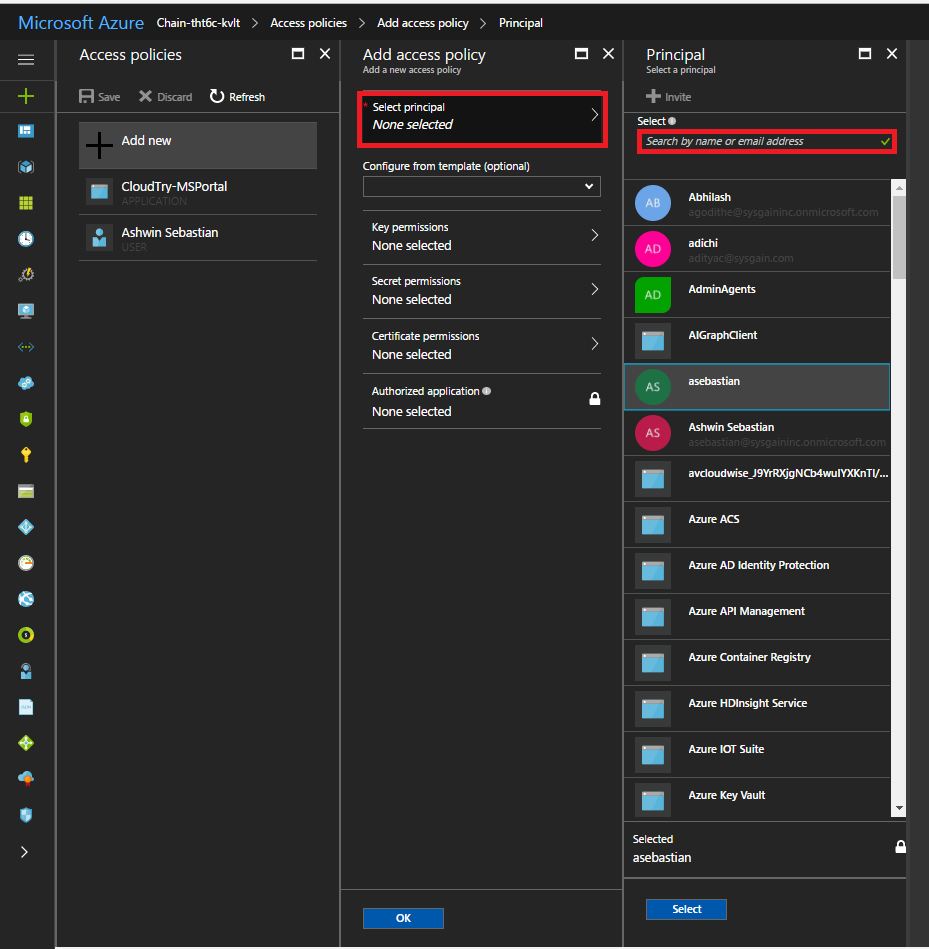
1. Open the ***key vault*** resource from your resource group.



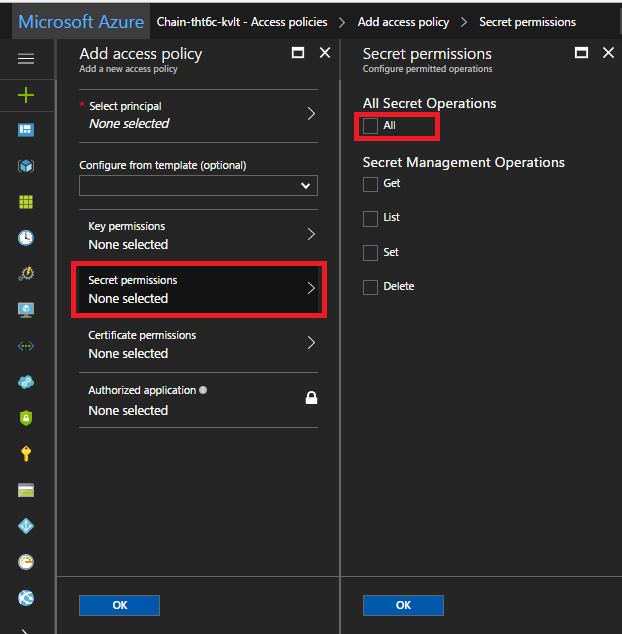
1. Click on the Access policies then click on ***Add new***, a new pane opens up called as Add access policy

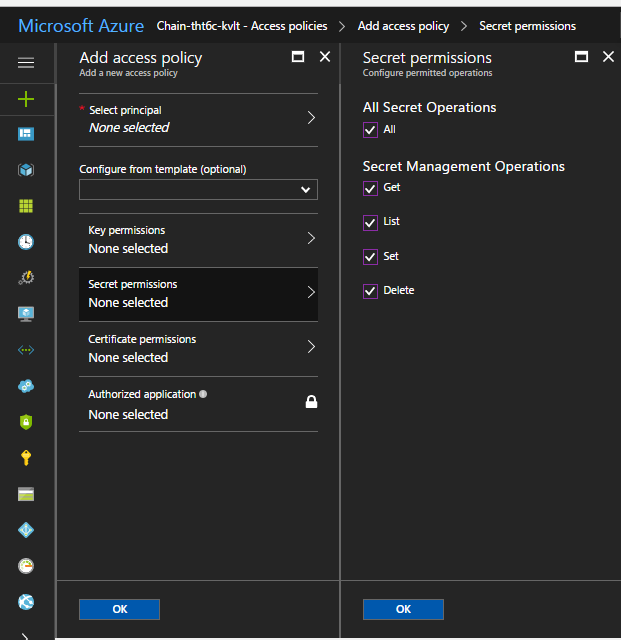


1. In ***select principal section***, select your username and click select.

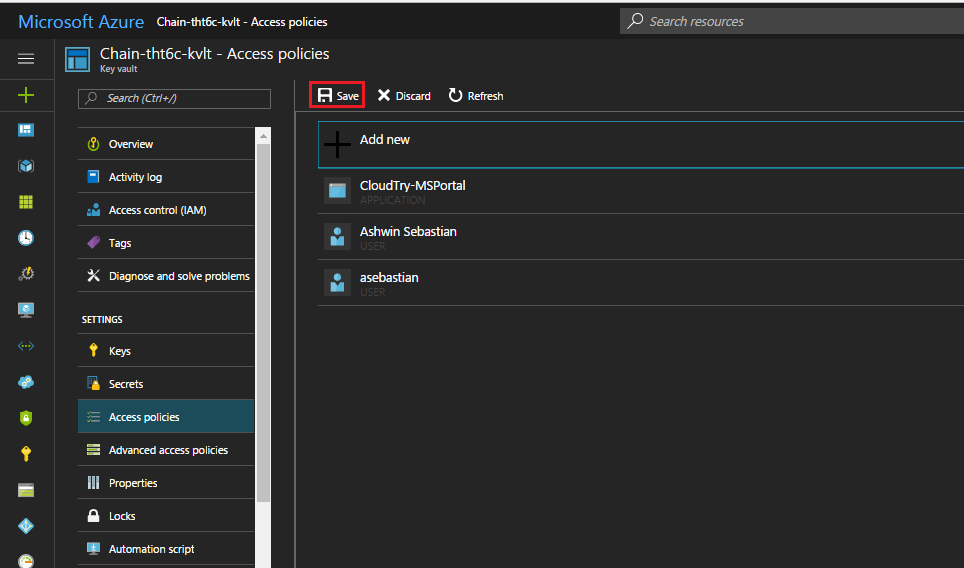


1. In ***Secret permissions section***, check the All checkbox to give all permissions to the concerned object ID and click **ok.**





1. Your user name will be listed under Access polices. Now click on ***save***.



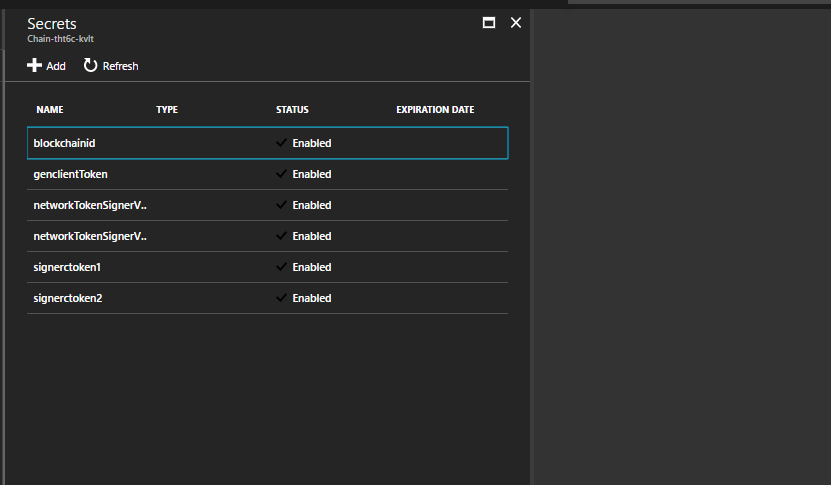
1. Thus you have successfully added your object ID to the

## Accessing the values from the key vault

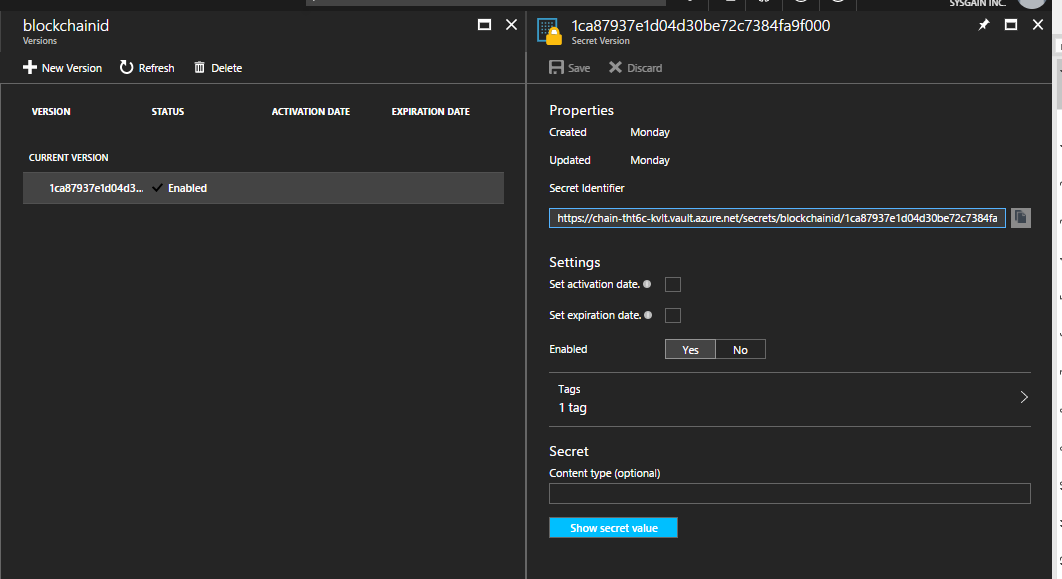
1. Open the ***key vault*** resource from your resource group.



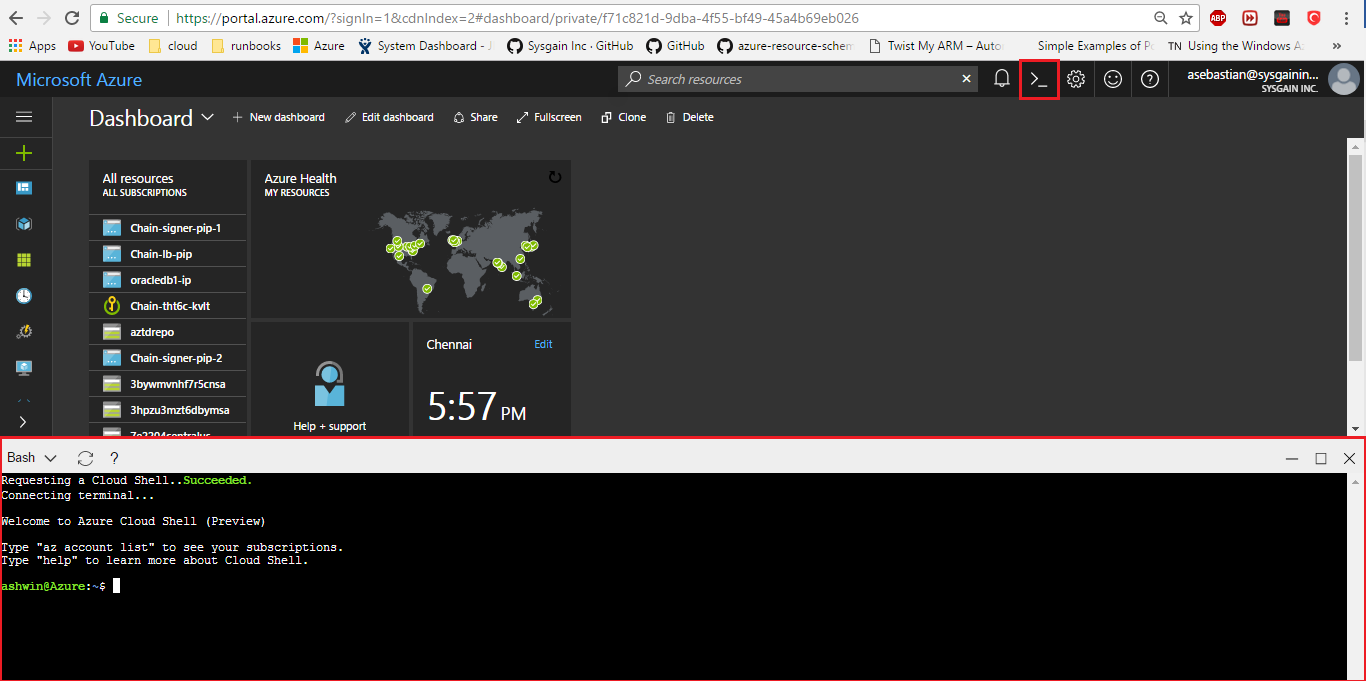
1. Click on the ***Secrets*** and you will be able to see the list of all the secret names.



1. You can click on any secret and view its secret by clicking on the current version of the secret name and click on ***Show secret value***.



You can also retrieve the secret values from the key vault using the Cloud Shell feature of the Azure portal.

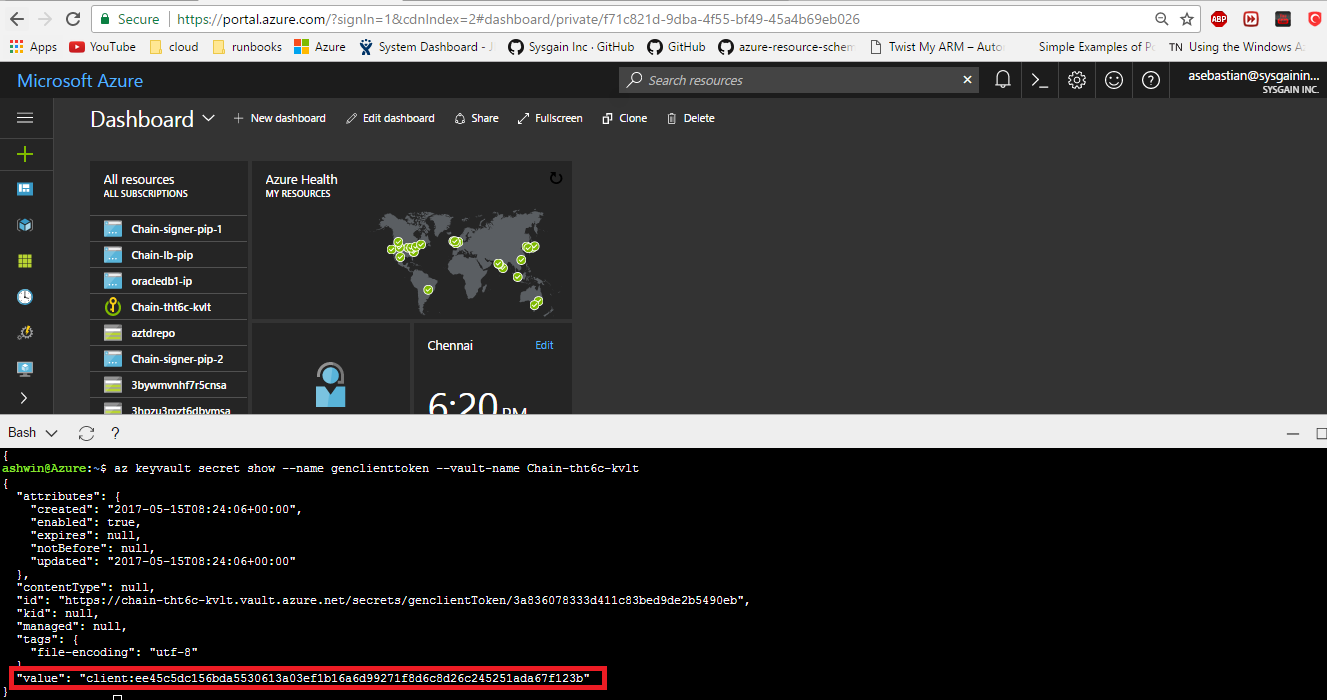


Use the following commands to access the key vault and view the secret name

***Note:*** These commands can only work when you have added your principal ie object ID to the access policies of the key vault.

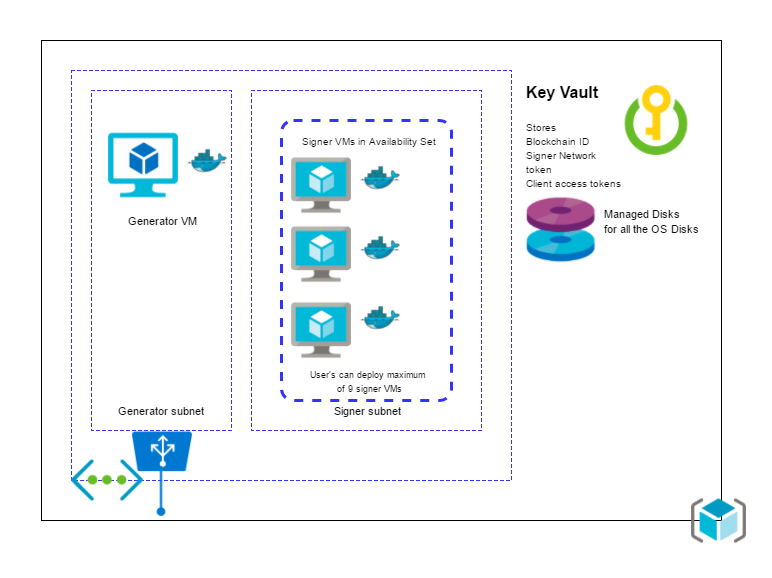
az keyvault secret show - -name <secret-name> - -vault-name <keyvault-name>

to view the secret value.



# Single Generator, Multi Signer, Single Region

## Architecture Diagram



## Description

1. A Generator VM (Ubuntu Canonical 16.04) which launches chaincore in port 1999 through chaincore Docker container.
   1. This VM is configured as a Block generator using a shell script as an extension to this VM.
   2. This VM is provisioned in a separate subnet ie generator subnet.
   3. It has an external load balancer from where internet traffic reaches the VM.
2. Next is the Signer VM (Ubuntu Canonical 16.04) which launches chaincore in port 1999 through chaincore Docker container.
   1. User has the option to provision up to 9 block signer VM’s
   2. These VM’s are configured as a Block signers using a shell script as an extension to the VM.
   3. These VM’s are provisioned in a separate subnet ie signer subnet.
3. A key vault to store the blockchain ID, network token for signers and client tokens for all the containers.
4. We use managed disks for provisioning the OS disks of the VM.

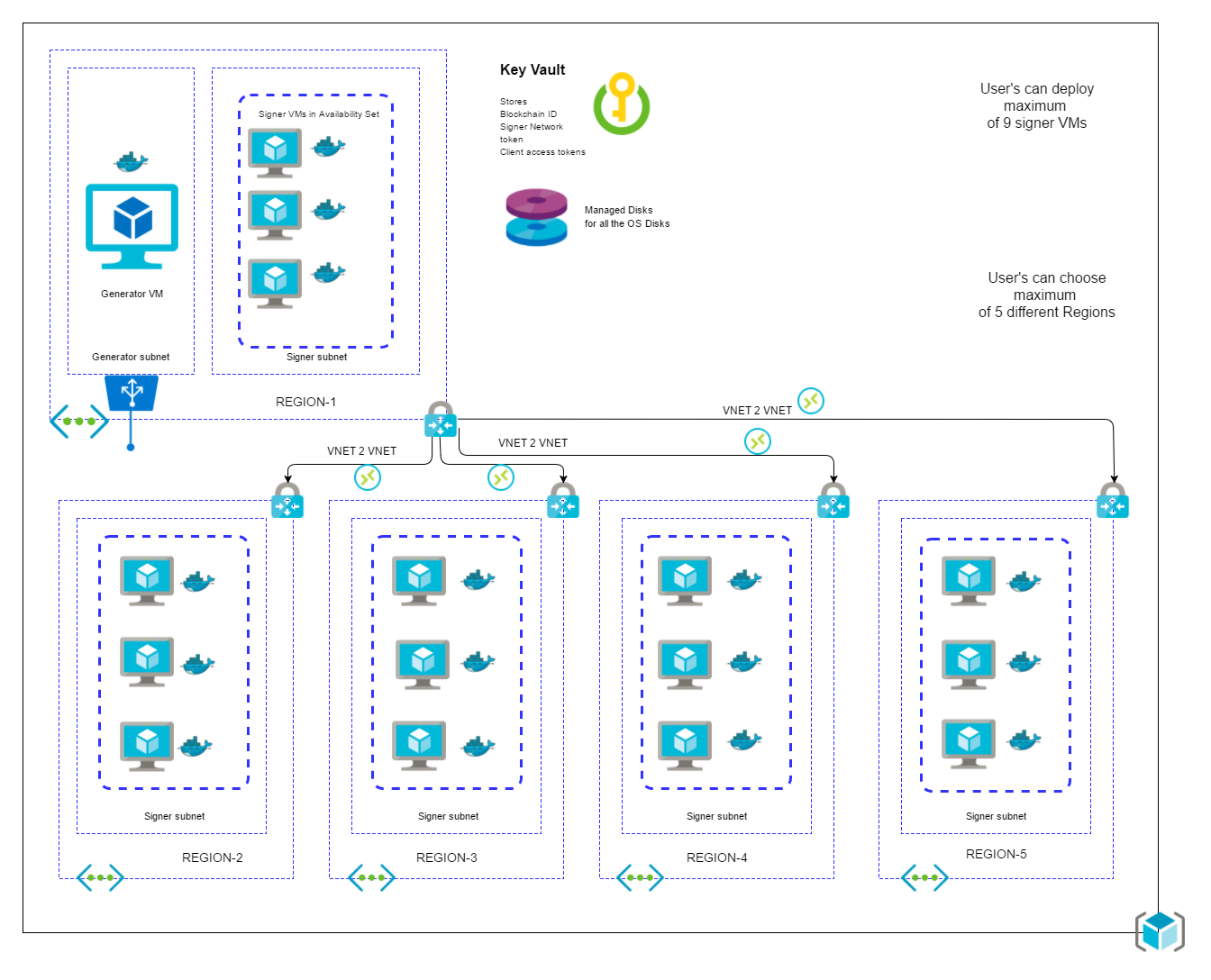
## Dataflow

The blockchain ID, network token for signers are stored in key vault from generator VM.

The Signer VMs retrieve blockchain ID and network token from the key vault and configure block signer and store signer client token in the key vault

# Single Generator, Multi Signer, Multi Region

## Architecture



## Description

1. A Generator VM (Ubuntu Canonical 16.04) which launches chaincore in port 1999 through chaincore Docker container.
   1. This VM is configured as a Block generator using a shell script as an extension to this VM.
   2. This VM is provisioned in a separate subnet ie generator subnet.
   3. It has an external load balancer from where internet traffic reaches the VM.
   4. Its deployed only in first region.
2. Next is the Signer VM (Ubuntu Canonical 16.04) which launches chaincore in port 1999 through chaincore Docker container.
   1. User has the option to provision up to 9 block signer VM’s
   2. These VM’s are configured as a Block signers using a shell script as an extension to the VM.
   3. These VM’s are provisioned in a separate subnet ie signer subnet.
   4. Its deployed in 5 regions based on number of regions selected by User.
3. A key vault to store the blockchain ID, network token for signers and client tokens for all the containers.
4. We use managed disks for provisioning the OS disks of the VM.
5. It deploys VNET gateways for each VNET in each region.
6. These VNETs communicate with each other using VNET2VNET connections

## Dataflow

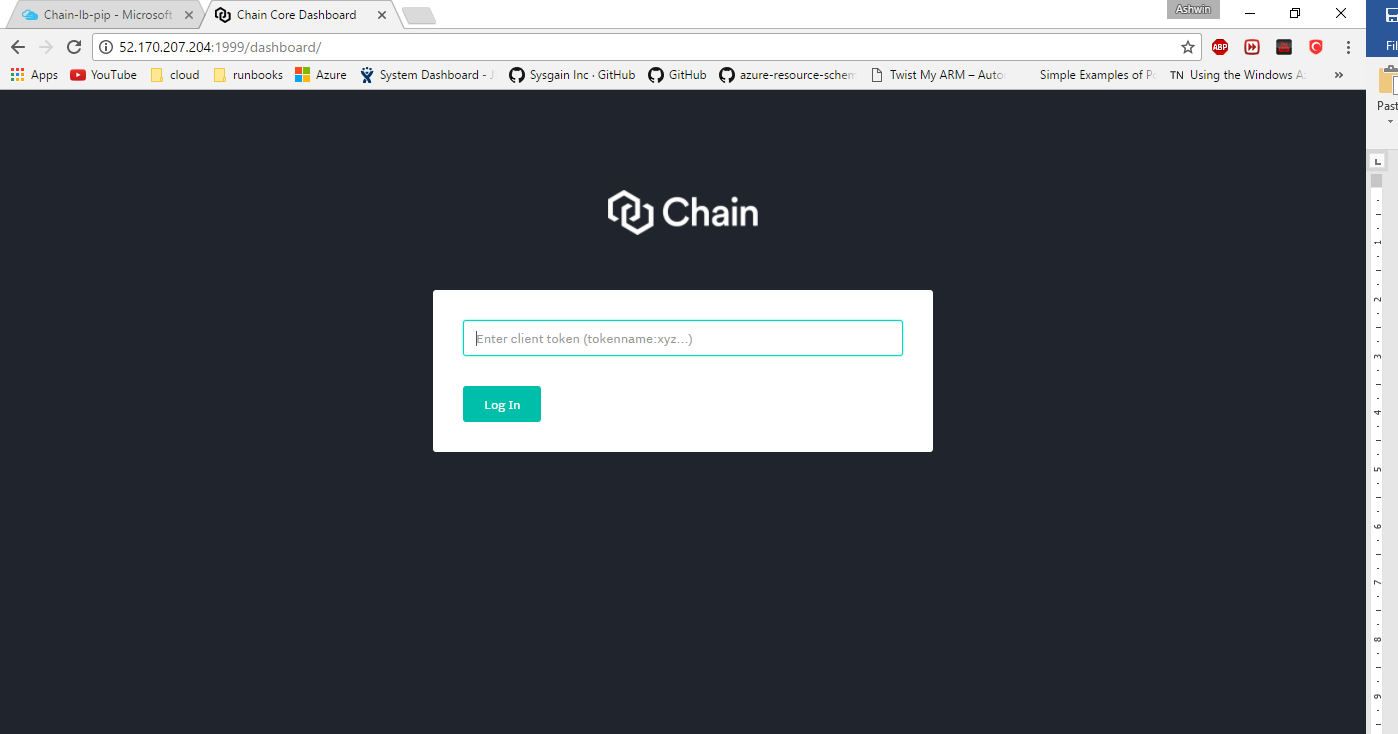
The blockchain ID, network token for signers are stored in key vault from generator VM.

The Signer VMs retrieve blockchain ID and network token from the key vault and configure block signer and store signer client token in the key vault

# Post Deployment

After the template deployment has completed you can access the generator dashboard by accessing generator VM’s IP address at port 1999

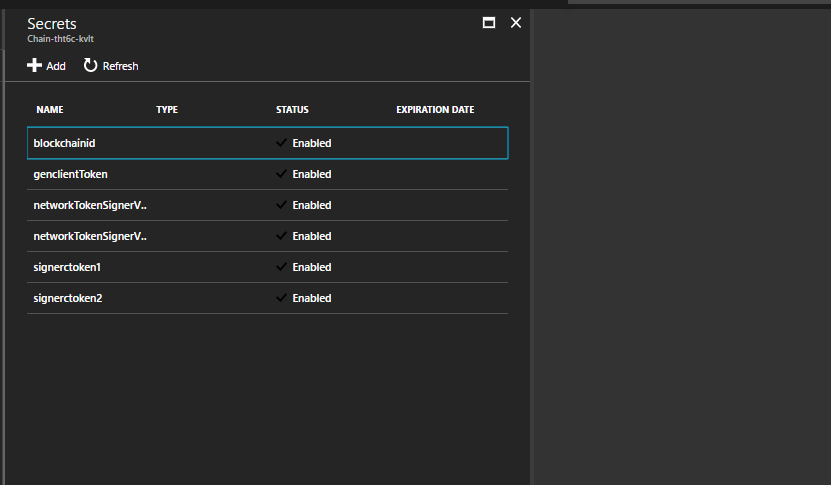
http://<generator\_LB\_IP\_address>:1999

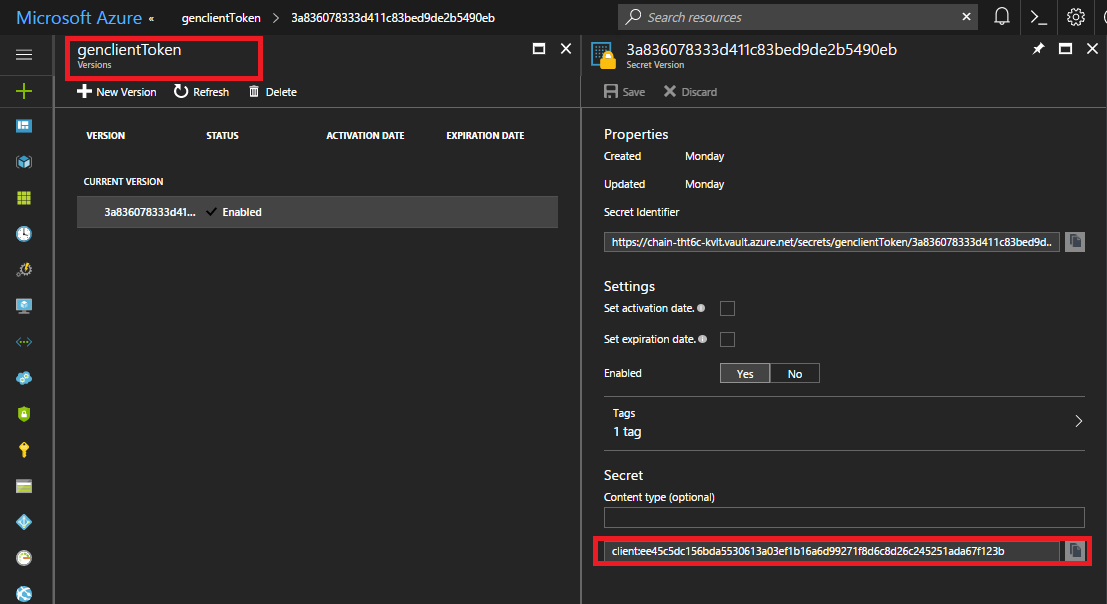


You can get the client token from the key vault secrets

The secret name of the generator client access token is ***genclienttoken.*** Click on the secret name and then click on ***show secret value***.

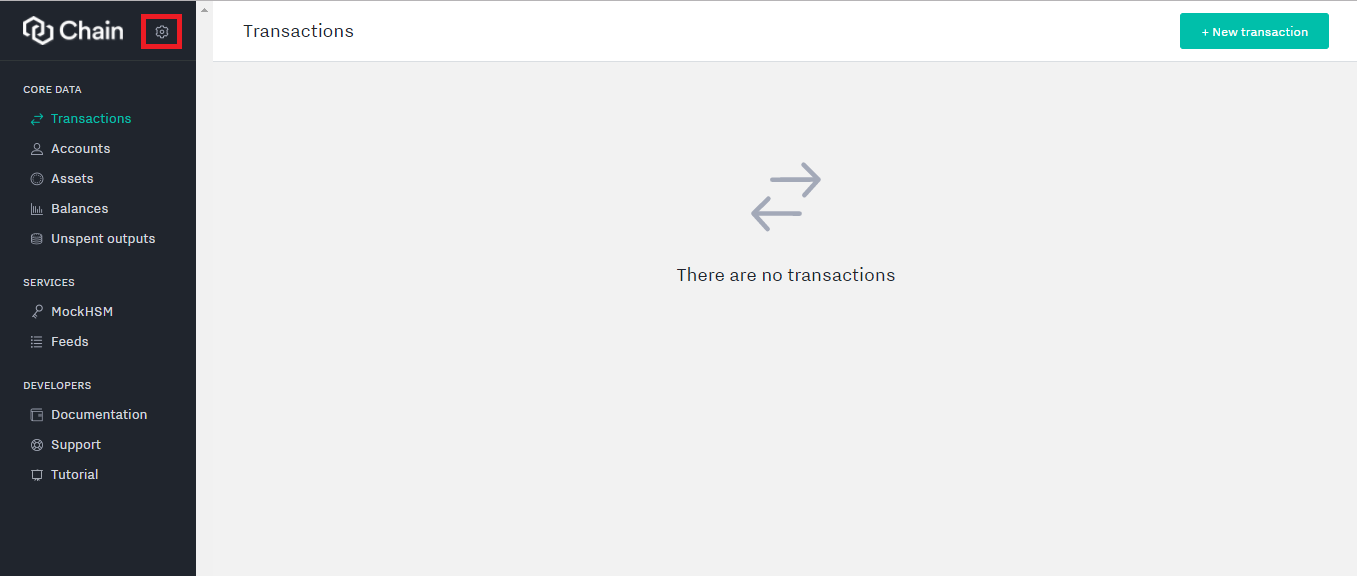




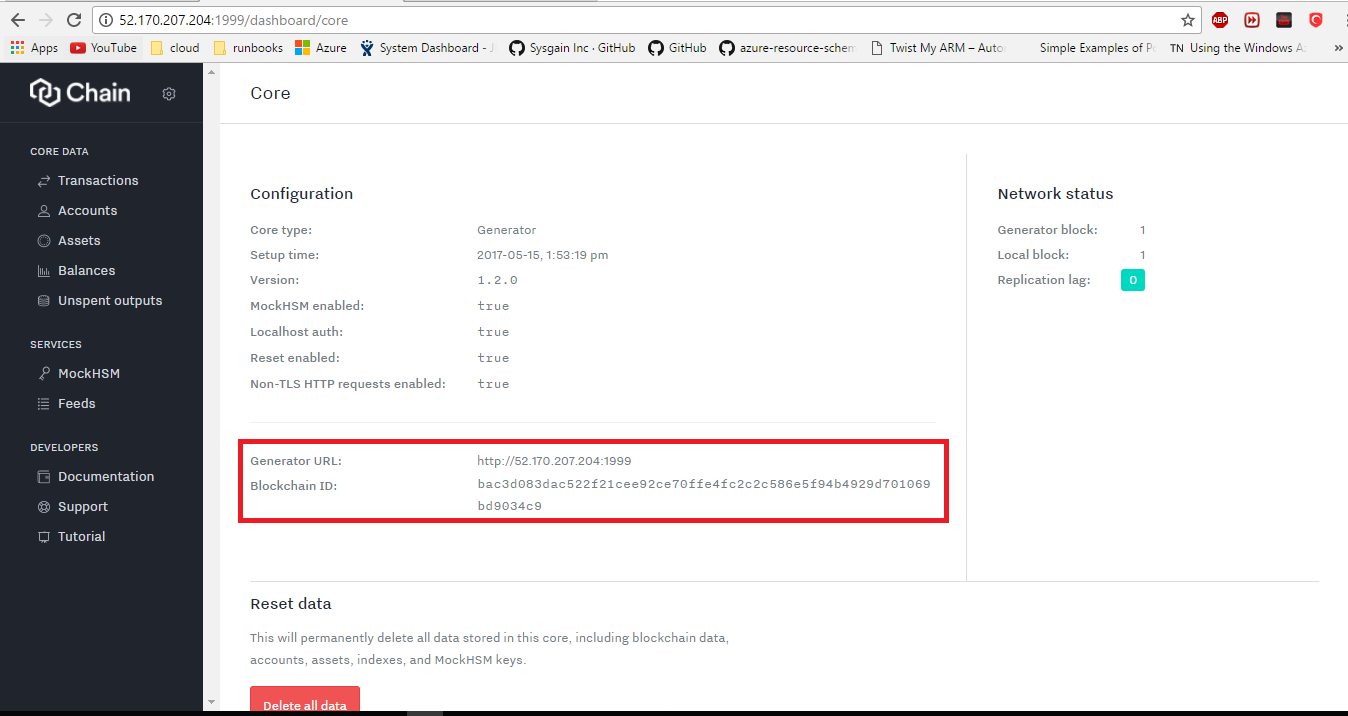


This shows client access token which you need to use to open the dashboard of generator.

Go ahead and copy the token and paste it in browser.



Click on the settings as shown in above screenshot, and select the core status to view the generator VMs

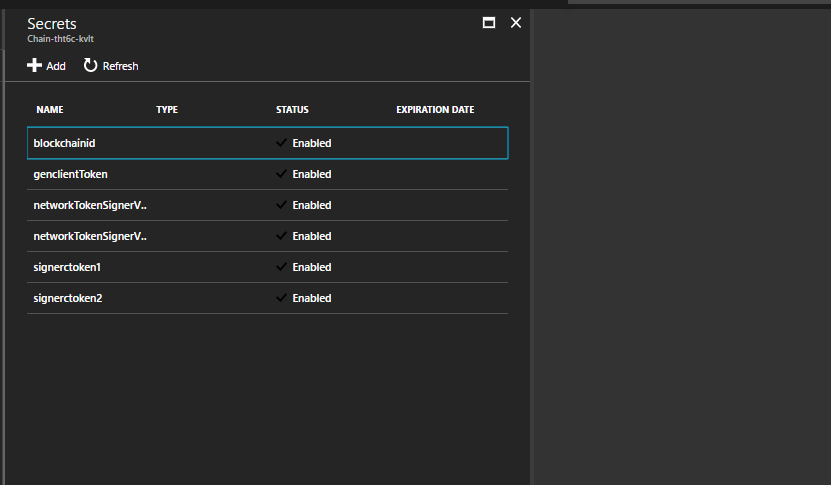


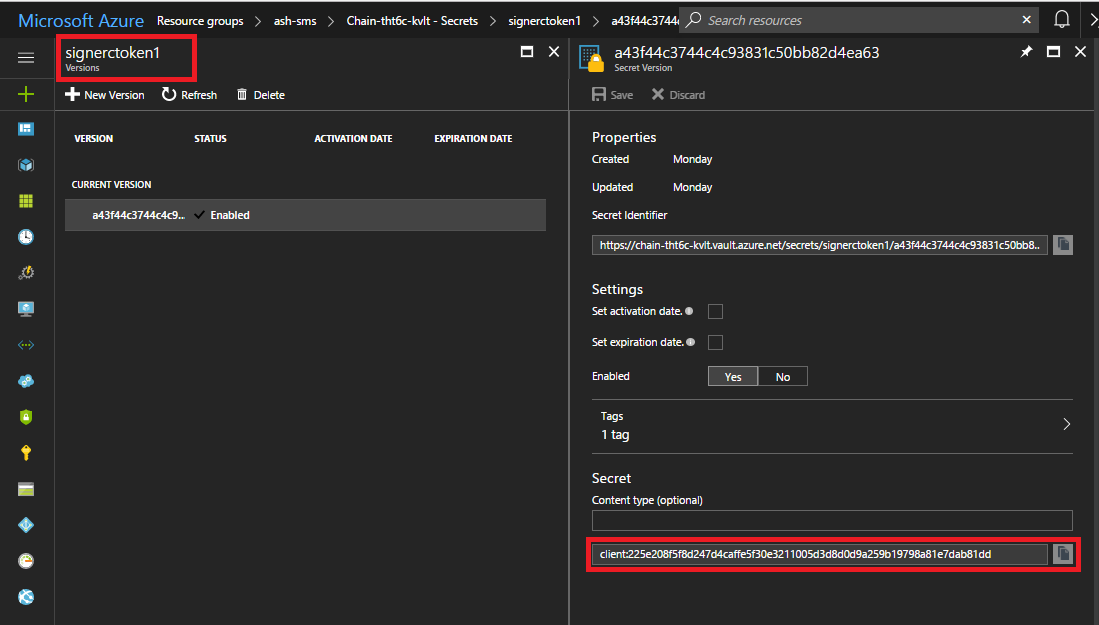
Here you will be able to see the blockchain ID generated in the generator VM.

Now similarly open any of the signer VM to see if that VM is in sync with the generator VM

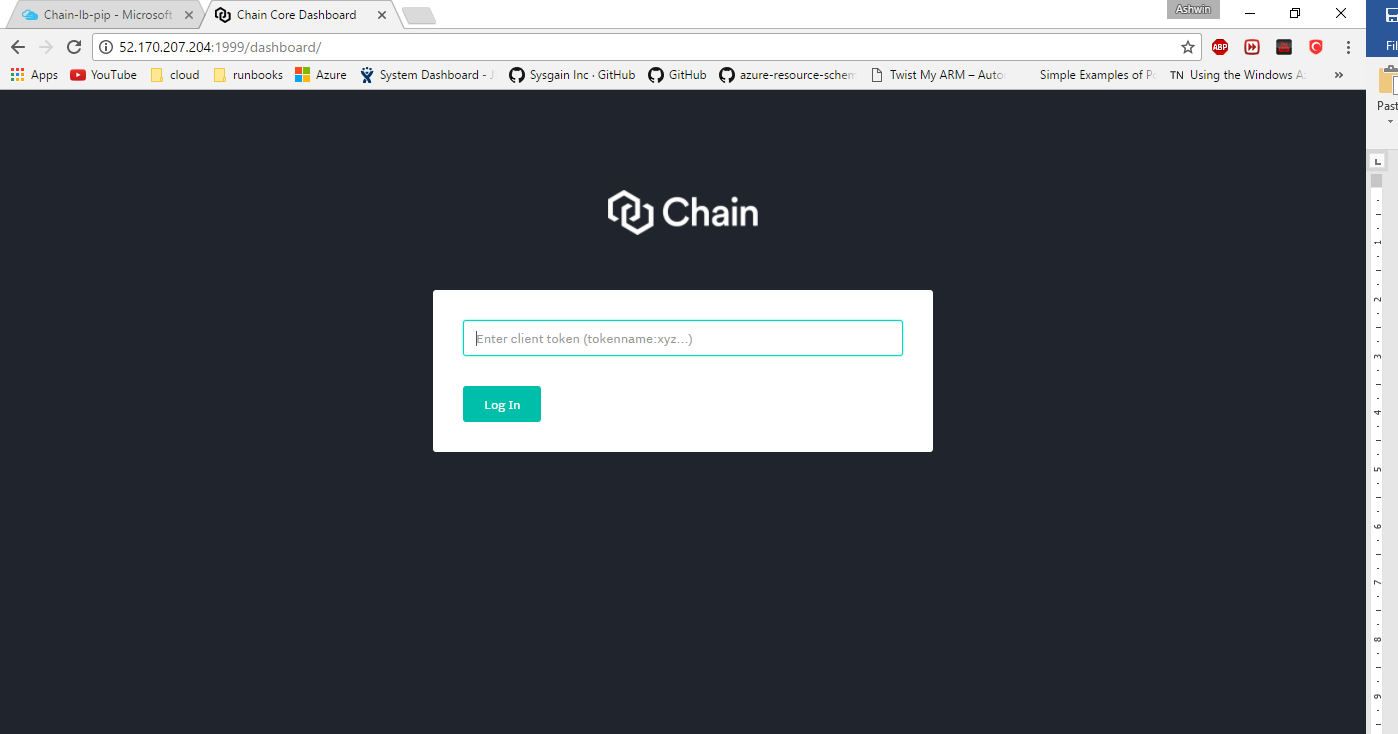
Open the dashboard of one of the signer VM using its client token which is stored in the key vault as part of this deployment.

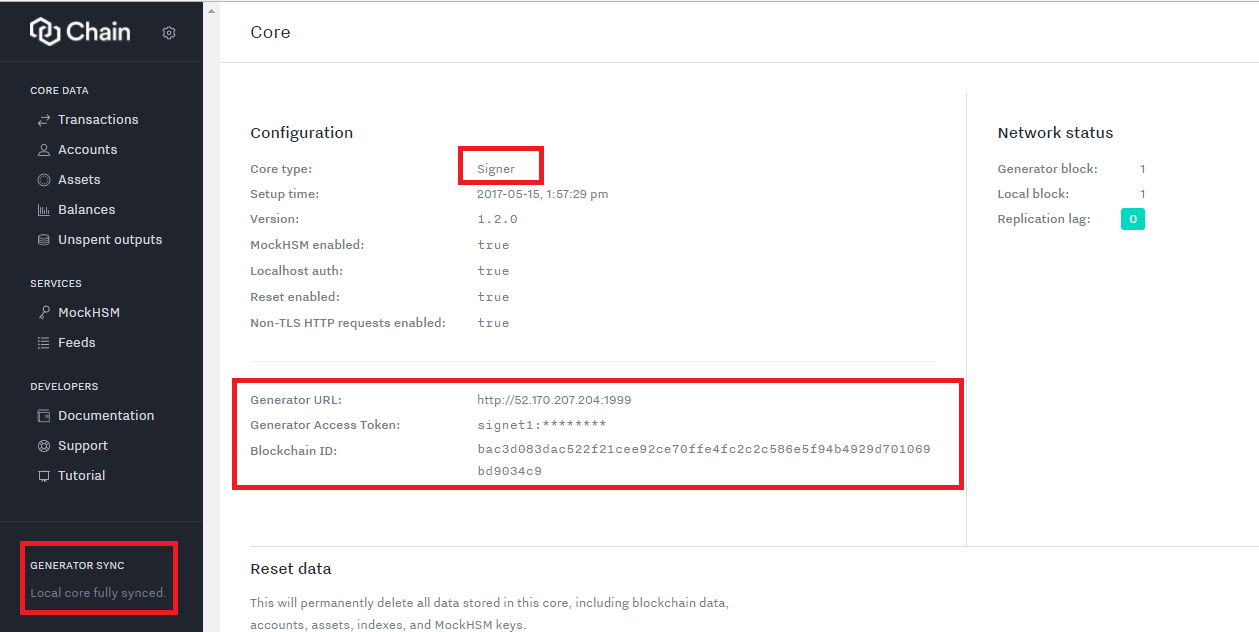






Open the signer dashboard using signer client token and open the core status to see that it is synced with the generator VM.





Thus you can see that the signer VM is in sync with the generator VM.