# User Guide for Azure Blockchain Hyperledger Fabric template (AKS based)

Contents

[User Guide for Azure Blockchain Hyperledger Fabric template (AKS based) 1](#_Toc22648391)

[Onboarding to Preview 1](#_Toc22648392)

[Support Forum 1](#_Toc22648393)

[Hyperledger Fabric Blockchain network setup 1](#_Toc22648394)

[Deploy the orderer/peer organization 2](#_Toc22648395)

[Build Consortium 6](#_Toc22648396)

[Chaincode commands 9](#_Toc22648397)

Azure Blockchain Hyperledger Fabric template provides customers a simplified experience to deploy the Hyperledger Fabric network components on Azure. It deploys the components on Azure Kubernetes Service thus simplifying the orchestration and management of the cluster. This user guide will describe the different input parameters that are required for deploying the template and take you through the steps required to setup your Hyperledger Fabric Blockchain network.

## Onboarding to Preview

The Azure Blockchain for Hyperledger Fabric (AKS based) template is enabled in Azure Marketplace for private preview customers only. Customers whose subscriptions are whitelisted by Microsoft team will be able to access the template.

To get yourself onboarded to the private preview please mail your subscription ID details along with your organization details to [azhlf@microsoft.com](mailto:azhlf@microsoft.com)

## Support Forum

All the private preview customers will be added to the teams channel Azure Blockchain Previews > [Hyperledger Fabric AKS template preview](https://teams.microsoft.com/l/channel/19%3ac3c902493eea4b88a23de3728d0469a2%40thread.skype/Hyperledger%2520Fabric%2520AKS%2520template%2520preview?groupId=e5e57fed-4b1b-49ca-bfed-ec8a18f6a8d3&tenantId=72f988bf-86f1-41af-91ab-2d7cd011db47). Customers can send out their queries/requests on this channel, we will be able to reply to your queries daily.

*Note: Please indicate if you do not want to be added to the support forum teams channel by sending a mail to* [*azhlf@microsoft.com*](mailto:azhlf@microsoft.com)

## Hyperledger Fabric Blockchain network setup

The Fabric deployment through this template mainly involves 2 major steps

1. Deploy the orderer/peer organizations
2. Build the consortium

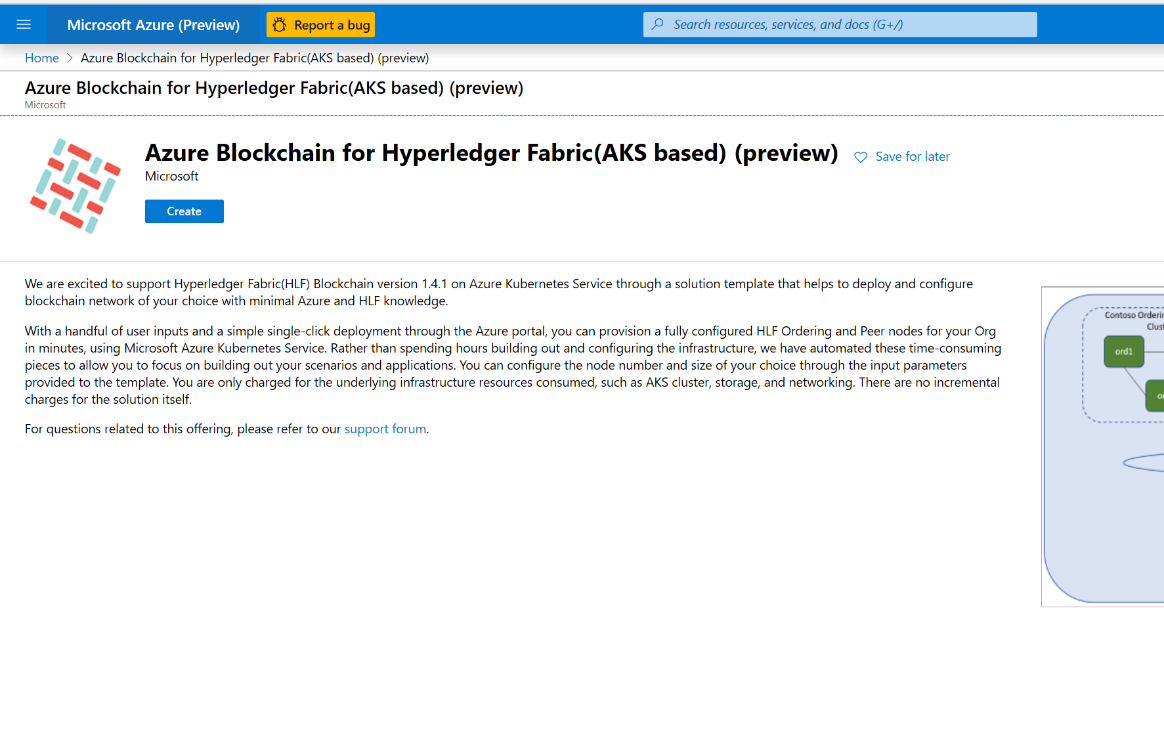
User will have to follow the below steps in the order mentioned to setup the Fabric Blockchain network.

## Deploy the orderer/peer organization

To get started with the HLF network components deployment, navigate to Azure portal marketplace offering link here <https://aka.ms/hlf>

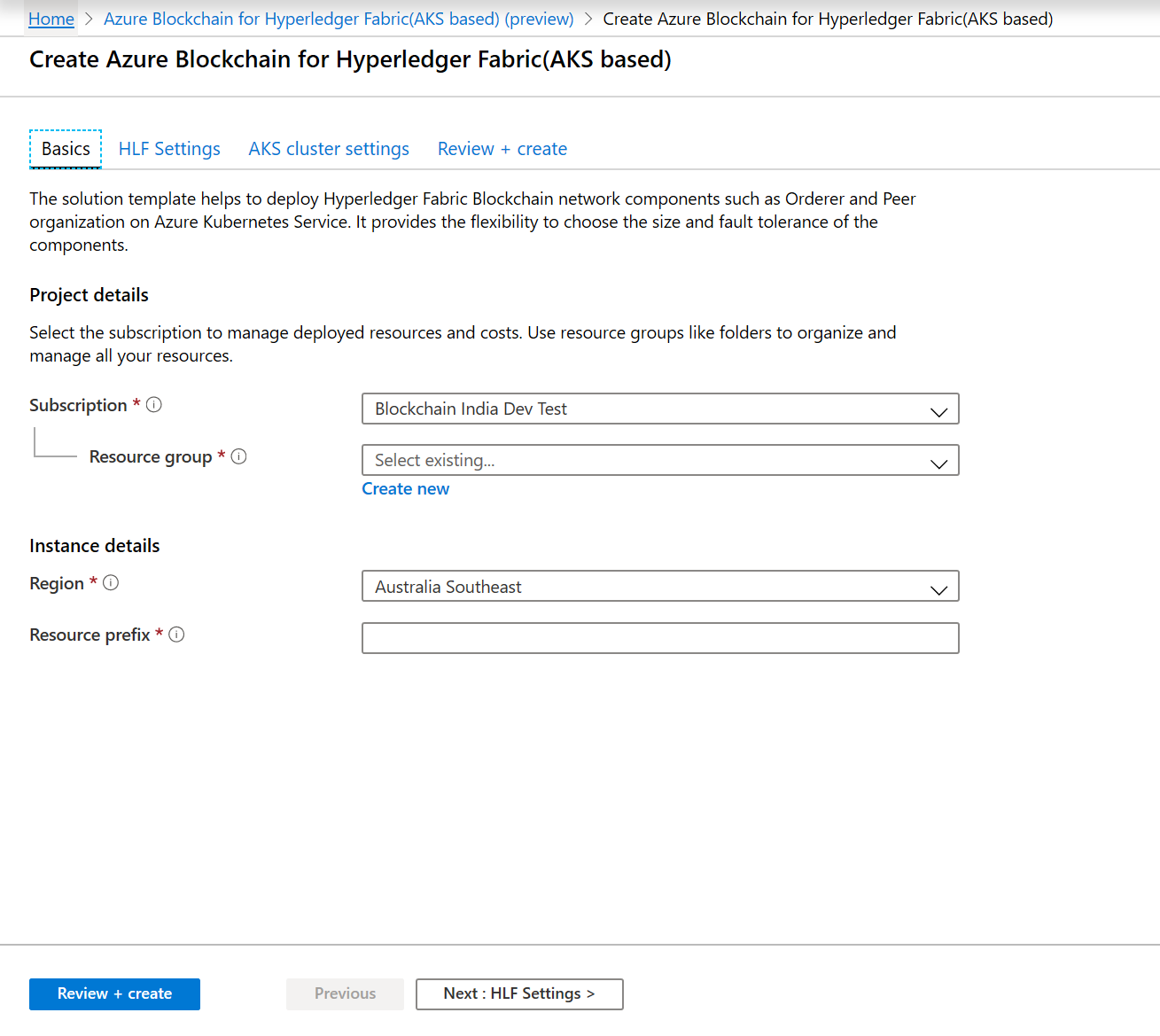
Please provide the below input parameters as per your requirements

1. Click on create to start the template deployment



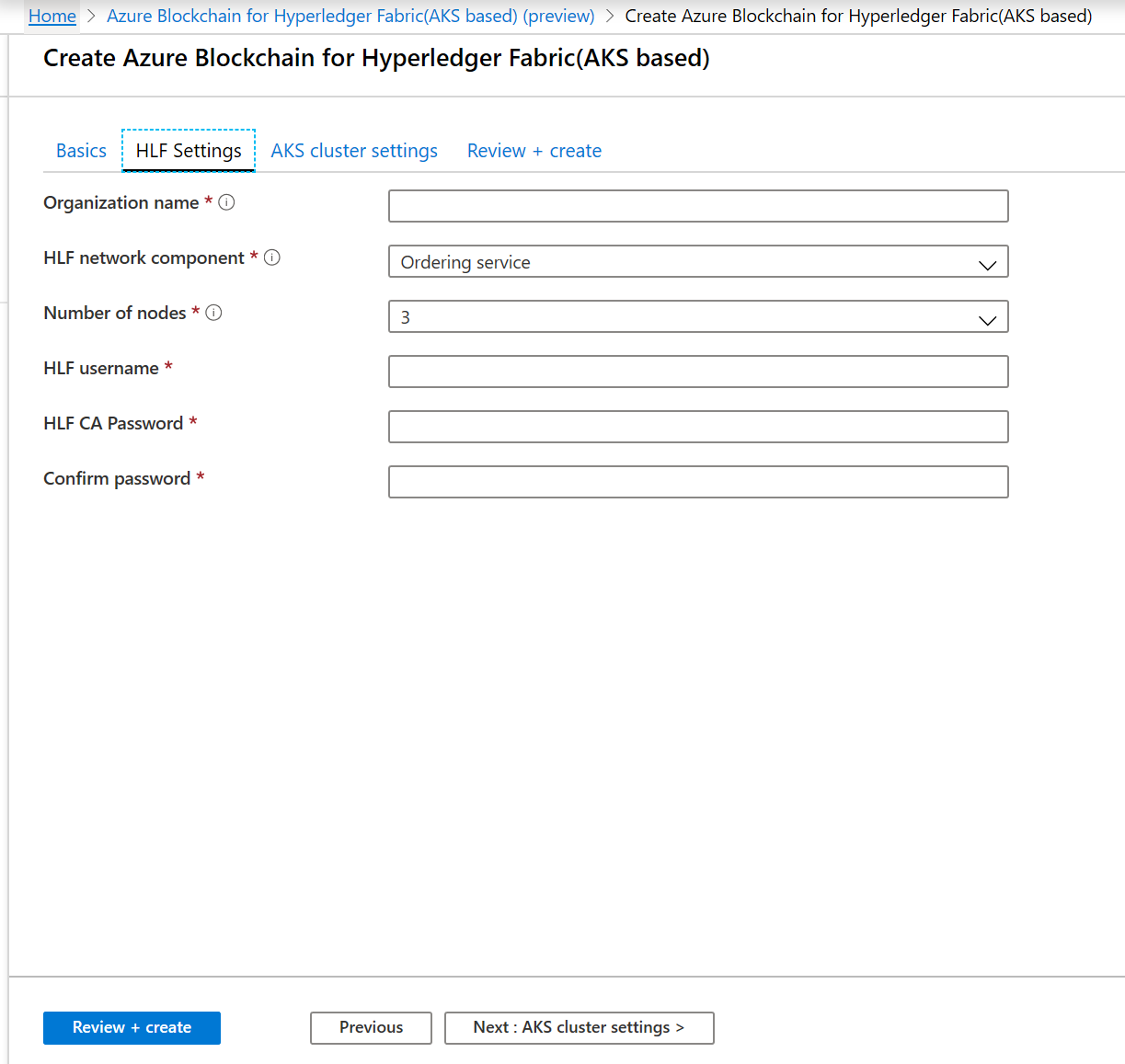
1. Provide the input parameters in the Basics tab

* Subscription: Choose the subscription name where you want to deploy the HLF network components
* Resource Group: Either create a new resource group or choose an existing empty resource group, this resource group will hold all resources deployed as part of the this template
* Region: Choose the Azure region where you want to deploy the Azure Kubernetes cluster for the HLF components
* Resource prefix: Prefix for naming of resources that will be deployed. This should be less than 6 characters in length, including lower case alpha and numbers only.



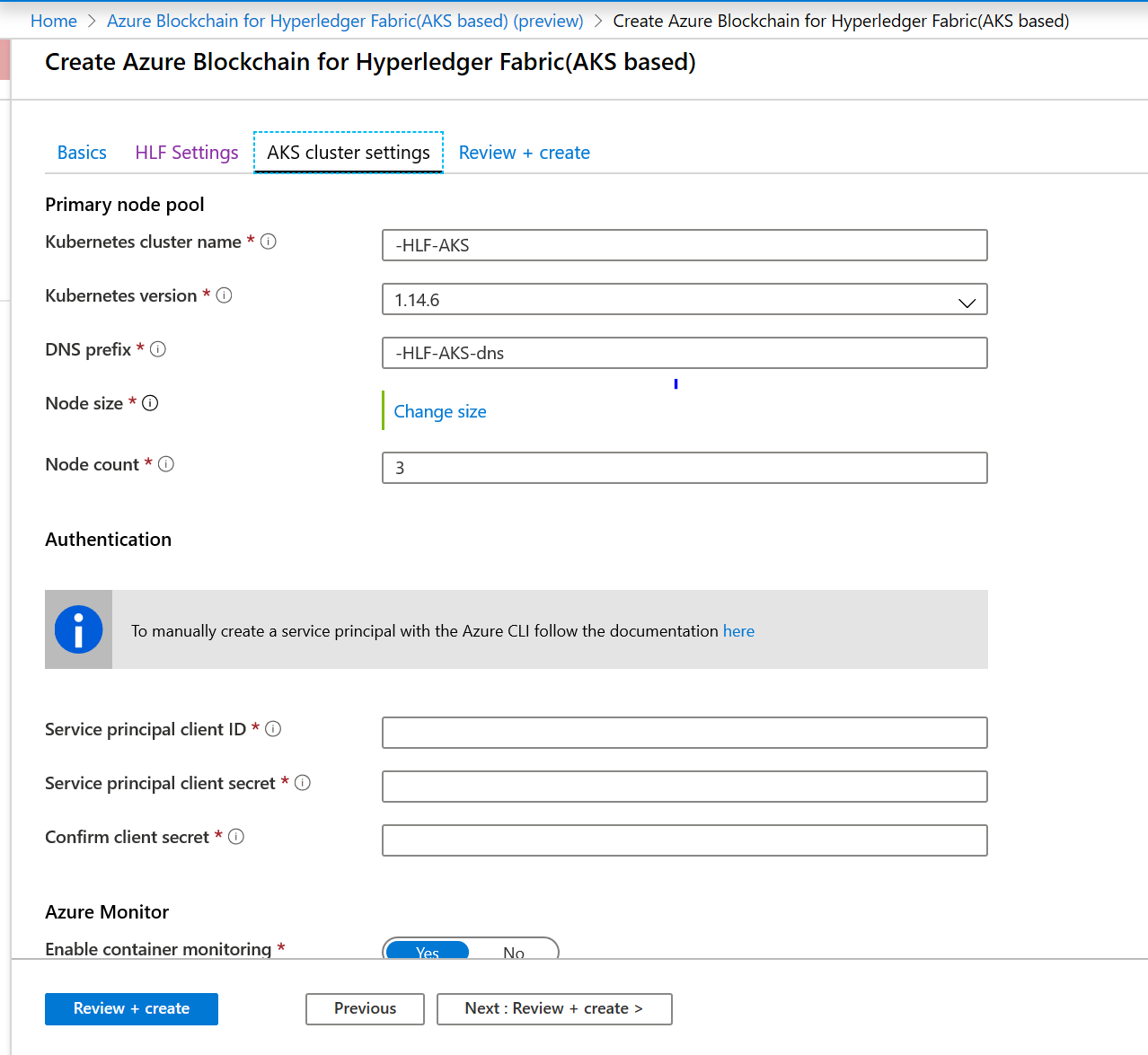
1. The next set of input parameters define the HLF network component that will be deployed

* Organization name: The name of the HLF organization, this is required for various data plane activities
* HLF network component: Choose either Ordering Service or Peer nodes based on what Blockchain network component you want to setup
* Number of nodes: In case of Ordering service, number of nodes provide the fault tolerance to the network, 3,5 and 7 are the supported orderer node count. In case of Peer nodes, you can choose 1-5 nodes based on your requirement
* Peer node world state database: For peer node database we currently support CouchDB only, hence this is preselected
* HLF username: Provide the username that can be used for the HLF network component created
* HLF CA password: Provide a password that will be used for HLF authentication for the username provided earlier



1. The next set of input parameters define the Azure Kubernetes cluster configuration which is the underlying infrastructure on which the HLF network components will be setup

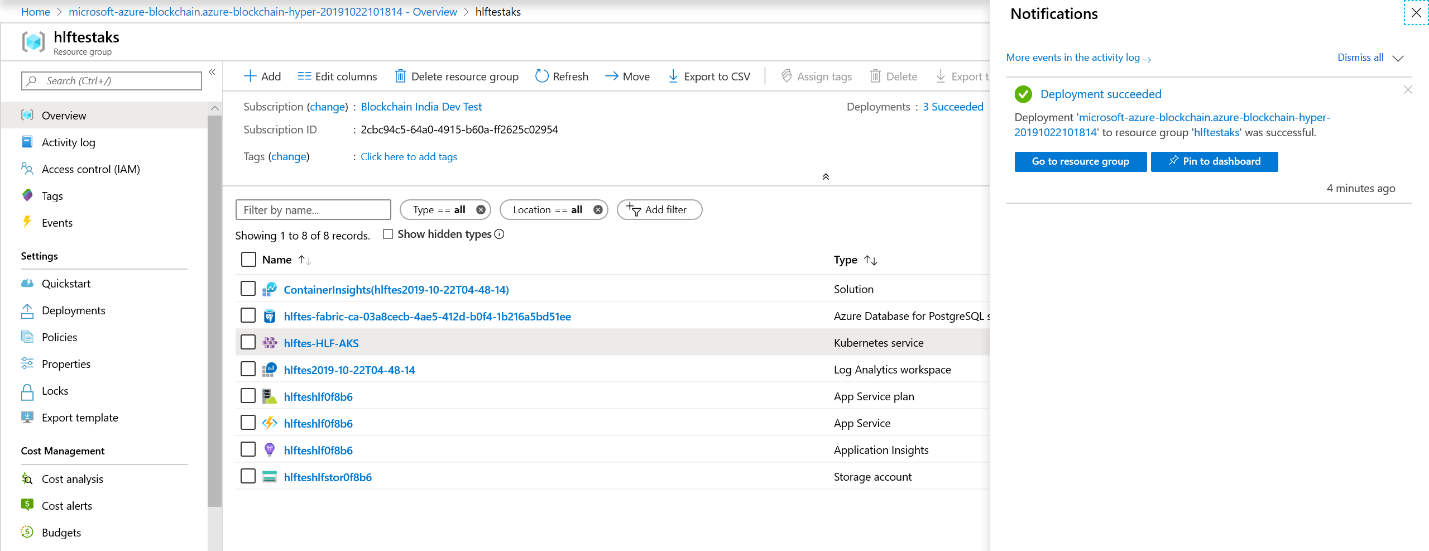
* Kubernetes cluster name: The name of the AKS cluster that will be created, this field will be prepopulated based on the inputs given earlier, you can change if required
* Kubernetes version: The version of the Kubernetes that will be deployed on the cluster created. Based on the region selected in the Basics tab, the supported version might change
* DNS prefix: DNS name prefix to use with the hosted Kubernetes API server FQDN. You will use this to connect to the Kubernetes API when managing containers after creating the cluster.
* Node size: The size of the Kubernetes node, you can choose from the list of VM SKUs available on Azure. For optimal performance we recommend Standard DS2 v2
* Node count: The count of the number of Kubernetes nodes to be deployed in the cluster. We recommend keeping this node count at least equal or more then the number of HLF nodes specified in the HLF settings
* Service principal client ID: Provide the client ID of an existing service principal or create a new one. This is required for the AKS authentication. You can refer to the documentation [here](https://docs.microsoft.com/en-us/azure/aks/kubernetes-service-principal#manually-create-a-service-principal) for steps to create service principal
* Service principal client secret: Provide the client secret of the service principal provided above
* Enable monitoring: You can choose to enable AKS monitoring, in which case the AKS logs will be pushed to the Log Analytics workspace specified
* Log Analytics workspace: This is prepopulated with the default workspace that will be created if monitoring is enabled



1. After providing all the input parameters click on Review and create, this will trigger validation of the input parameters provided, once the validation passes, you can click create.

The deployment usually takes 10-12 minutes, might vary depending on the size and number of AKS nodes specified.

After the deployment is successful, you would be notified through Azure notifications on top right corner, click on “Go to resource group” to check all the resources created as part of the template deployment. All the resource names will start with the prefix provided in the Basics setting in template creation.



## Build Consortium

To build the blockchain consortium post deploying the ordering service and peer nodes, you will have to carry out the below steps in sequence. Build Your Network(byn) script available [here](https://github.com/ravastra/ARM-template-for-Hyperledger-Fabric-based-on-AKS), will help you with setting up the consortium, creating channel and installing chaincode.

*Note: Build Your Network script provided is strictly to be used for demo/devtest scenarios. For production grade setup we recommend using the native HLF APIs*

Sample consortium details: the below steps are to build a consortium with an ordering service with 3 nodes and a peer organization with 2 nodes, both sharing a channel “testchannel”

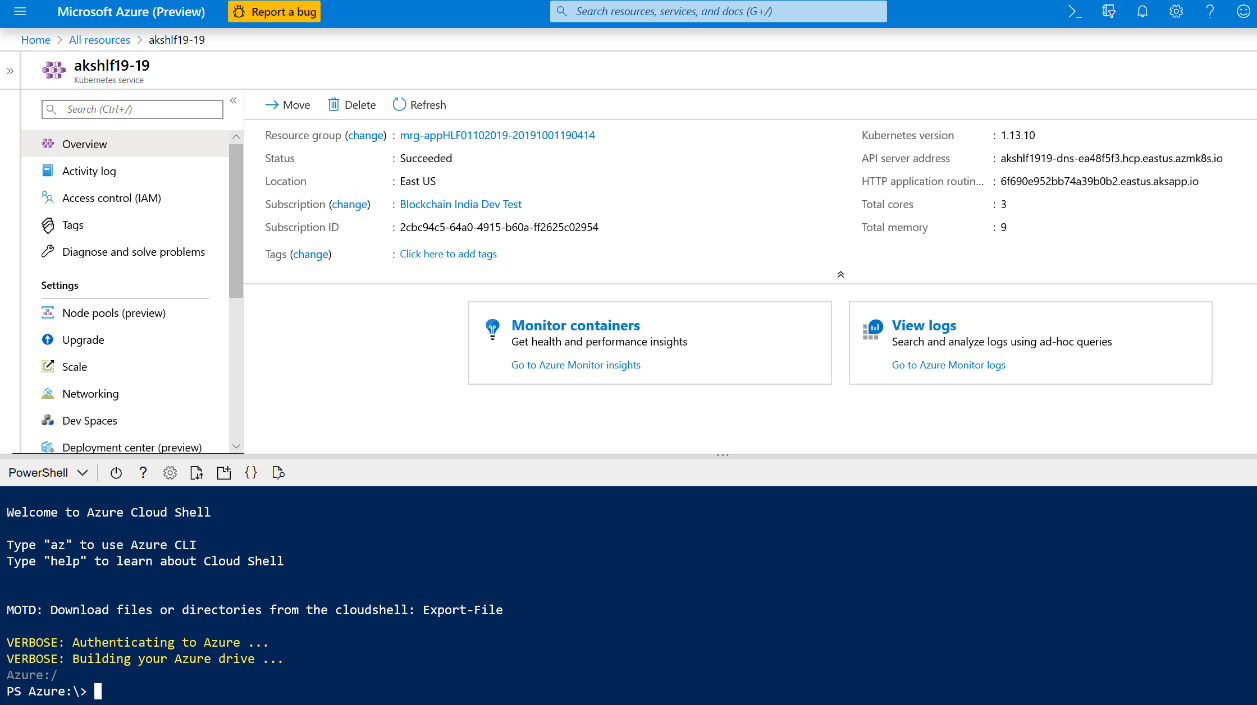
**Step 1:**

Create the ordering service with 3 nodes using the Azure Blockchain for Hyperledger Fabric(AKS based) marketplace offering as per the steps provided in section [Deploy orderer/peer nodes](#_Deploy_the_orderer/peer) above.

Create peer org1 with 2 nodes using the marketplace offering in peer org1’s Azure subscription. Create peer org2 with 2 nodes using the marketplace offering in peer org2’s Azure subscription. For dev-test scenarios you can create all 3 orgs in the same subscription.

***Note:***

*All the commands to run the byn script can be executed through Azure Bash CLI. You can login into Azure shell web version through the option*  *at the top right corner of the Azure portal. Once the command prompt comes up, type bash and enter to switch to bash CLI.*



**Step2: Orderer Org**

Connect to Ordering service Kubernetes cluster created.

# command to connect to an AK cluster

az aks get-credentials --resource-group “resource-group-name” --name “aks-cluster-name” --subscription “subscription-id”

Example

az aks get-credentials --resource-group hlftestaks --name hlftes-HLF-AKS --subscription 2cbc94c5-64a0-4915-b60a-ff2625c02XXX

You can quickly check how many pods are running in the aks cluster

# command to check the status of AK cluster

**kubectl get all**

To run all the consortium scripts in the aks cluster we need to start a test setup pod. This is solely to run the scripts; hence we recommend deleting the pod post the configuration of the consortium.

# command to create a 'test-setup' pod to run demo scripts, use the command as is

kubectl run -i --tty test-setup --image=akvhlftemplate.azurecr.io/fabric-tools:preview --serviceaccount="admin-account" --generator=run-pod/v1 --overrides='{ "apiVersion": "v1", "spec": { "imagePullSecrets": [{"name": "fabric-tools-secrets"}] } }'

Step3: Orderer Org

Create a channel, this should be executed on the orderer AKS cluster

# Command to create new channel

kubectl exec -it test-setup -- bash -c "/byn.sh createChannel \"${CHANNEL\_NAME}\""

Example

kubectl exec -it test-setup -- bash -c "/byn.sh createChannel \mychannel"

Now that you have the channel created, you can add peer organizations to this channel.

Step 4: Peer Org

To add a peer organization to the channel, you first need to upload the peer org MSP to Azure file storage. This command needs to be executed on the peer org AKS cluster.

# command to connect to an AK cluster

az aks get-credentials --resource-group “resource-group-name” --name “aks-cluster-name” --subscription “subscription-id”

Example

az aks get-credentials --resource-group myPeerRG --name PeerOrg --subscription 2cbc94c5-64a0-4915-b60a-ff2625c02XXX

As mentioned earlier, you will have to create a test setup pod in peer org as well, before running the byn script

# command to create a 'test-setup' pod to run consortium demo scripts, use the command as is

kubectl run -i --tty test-setup --image=akvhlftemplate.azurecr.io/fabric-tools:preview --serviceaccount="admin-account" --generator=run-pod/v1 --overrides='{ "apiVersion": "v1", "spec": { "imagePullSecrets": [{"name": "fabric-tools-secrets"}] } }'

Upload the org MSP to Azure file storage so that you can download it at the orderer org later

You can refer to the Azure storage documentation for creating a file storage. For example, you can use the below commands

# To create azure storage account

az storage account create -n MyStorageAccount -g MyResourceGroup -l westus --sku Standard\_LRS

# To create azure file share

az storage share create --name MyFileShare --account-name MyStorageAccount

# To generate azur file share URI

az storage share url --account-name MyStorageAccount --name MyFileShare

# To generate SAS Token for azure file with validity of 30 minutes

end=`date -u -d "30 minutes" '+%Y-%m-%dT%H:%MZ'`

az storage account generate-sas --account-name MyStorageAccount --expiry $end --https-only --permissions acdlpruw --resource-types sco --services f

Once you have the Azure file storage created and SAS token generated, use the below command to upload the peer org MSP.

# command to upload peer org MSP on Azure file storage

kubectl exec test-setup -- bash -c "/byn.sh uploadOrgMSP \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""

Step4: Orderer Org

Orderer org can now add the peer org into consortium and test channel by using the peer org MSP present in the file storage created earlier

# command to add peer org to consortium

kubectl exec -it test-setup -- bash -c "/byn.sh addPeerInConsortium \"${PEER\_ORG\_NAME}\" \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""

# command to add peer org to channel

kubectl exec -it test-setup -- bash -c "/byn.sh addPeerInChannel \"${PEER\_ORG\_NAME}\" \"${CHANNEL\_NAME}\" \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""

Step 5: Peer Org

After the orderer has added the peer org to the channel, peer org should join the peer nodes to the channel by downloading the orderer org TLS root certificate

# command to add peer node in the channel, this command add all the nodes of the peer

kubectl exec -it test-setup -- bash -c "/byn.sh joinNodesInChannel \"${CHANNEL\_NAME}\" \"${ORDERER\_END\_POINT}\" \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""

This completes the setup of HLF blockchain network with an ordering service and a peer organization with a test channel setup between them. You can add multiple peer organizations to the consortium following the same steps.

### Chaincode commands

Once you have the consortium setup, you can test your setup by running a chaincode on the network. Below is the sequence of steps that you will have to run on the peer AKS cluster. You will have to run these commands on each of the peer nodes where you want to install/invoke/query the chaincode.

# Install chaincode on a peer node. This command is to be executed on peer AKS cluster

**kubectl exec -it test-setup -- bash -c "/byn.sh installDemoChaincode \"${PEER\_NODE\_NAME}\""**

# Instantiate chaincode on a peer node. This command is to be executed on peer AKS cluster

**kubectl exec -it test-setup -- bash -c "/byn.sh instantiateDemoChaincode \"${PEER\_NODE\_NAME}\" \"${CHANNEL\_NAME}\" \"${ORDERER\_END\_POINT}\" \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""**

# Invoke chaincode on a peer node. This command is to be executed on peer AKS cluster

**kubectl exec -it test-setup -- bash -c "/byn.sh invokeDemoChaincode \"${PEER\_NODE\_NAME}\" \"${CHANNEL\_NAME}\" \"${ORDERER\_END\_POINT}\" \"${AZURE\_FILE\_STORAGE\_URI}?${SAS\_TOKEN}\""**

# Query chaincode on a peer node. This command is to be executed on peer AKS cluster

**kubectl exec -it test-setup -- bash -c "/byn.sh queryDemoChaincode \"${PEER\_NODE\_NAME}\" \"${CHANNEL\_NAME}\""**