AZURE CONFIDENTIAL COMPUTING SECURITY SERVICES

New product: Azure Managed Confidential Consortium Framework

NDA Gated Preview | Participant Resources & Preview Scope v1.0

Est. Start: Est. Duration: 8 weeks Complexity: Medium (1-2 hours of effort)

Description

Azure Managed Confidential Consortium Framework streamlines the experience of building, deploying, and operating a secure Confidential Consortium Framework (CCF) application on a governed network of participants. By leveraging the power of trusted execution environments in Azure, CCF provides users enterprise-ready multiparty systems while keeping the operator (Microsoft) out of the trusted computing base (TCB). This reduces reliance on a single party to execute based on a trust relationship. Network participants or members must achieve a majority vote to perform actions like adding a new consortium member or deploying code changes. Transactions on the network will produce cryptographically verifiable receipts to prove the network has not been tampered with. In addition, a central ledger records the history of the network for traceability and auditability.

Customer Use Cases:

1. A consortium of banks wants to share reference data about securities to understand if they are within market consensus. To form an accurate market view, the individual data points must come from authenticated participants without revealing the contents of the submissions and must aggregate correctly.

2. A payment provider can run payment data through regulator-provided fraud-detection algorithms, which can flag a transaction without revealing personal data. Then, an independent authority can audit and confirm the accuracy of the fraud detection algorithm for flagged payments while protecting user privacy.

3. One or more software publishers can submit their Software Bill of Materials (SBOMs) to a service that provides receipts that guarantee their provenance, total ordering, immutability, and compliance with registration policies. The end users benefit from transparency over the published software by checking offline-verifiable receipts without needing access to the service.

Known Issues / Limitations

- During private preview, Managed CCF is available in the South-Central US region only.

Q&A

- *I don't know about Confidential Consortium Framework. Where can I learn about it?*
  The open-source Confidential Consortium Framework (CCF) website is a good place to learn about the technology, the benefits. In addition, check out their documentation to begin building your CCF-aware applications that you can run on Azure Managed Confidential Consortium Framework.

- *I want to use confidential computing but not sure what I can do with Azure Managed Confidential Consortium Framework.*

You can build applications with governance controls on data that can benefit from confidential centralized computing. Imagine a network where multiple parties agree on the code that will be run for an application, each owns their data, and only shares data related summaries or selectively reveals part of the data when conditions are met. Transactions on the network remains auditable through a ledger.

- *Is this similar to Distributed Ledger Technology?*
  Distributed Ledger Technology(DLTs) and CCF technologies support similar use cases & scenarios, have similar concepts like merkle tree, ledger, fault tolerance etc., though differ in how trust is established. Particularly, CCF based technologies utilize centralized computation via hardware-based Trusted Execution Environments to establish trust in the network, thereby enhancing the governance controls experience on the network. For more information, please read our Architecture documentation to learn about core concepts.

- *How do I begin?*
  You can use the instructions in this document to begin with Azure Managed CCF. We also have a quick start you can reference in CCF documentation.

Glossary

CCF	Confidential Consortium Framework
Consortium	A governance-based model where members can propose changes to a service. Other members can vote to accept or reject the change.

Preview Prerequisites

Aspect	Details
Required/Preferred Environmental Requirements	• An active Azure subscription. • Access to a Linux shell. • A machine with Azure CLI or access to the Azure cloud shell. Azure CLI can be installed by following the instructions at How to install the Azure CLI | Microsoft Learn
Required Roles & Permissions	Administrator
Clouds	Commercial Cloud

Feedback

What can be improved in the product experience?
What features would you like to see?
How might this technology support multi-party computing?
Would you be willing to pay a premium for this technology?

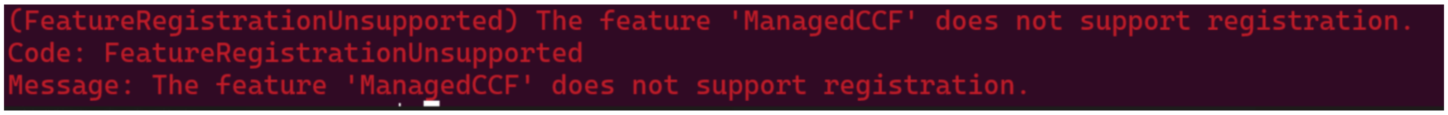
Onboarding to the Preview

Run the following commands in your subscription to enable the feature.

```
az feature registration create --namespace Microsoft.ConfidentialLedger --name ManagedCCF

az provider register --namespace Microsoft.ConfidentialLedger
```

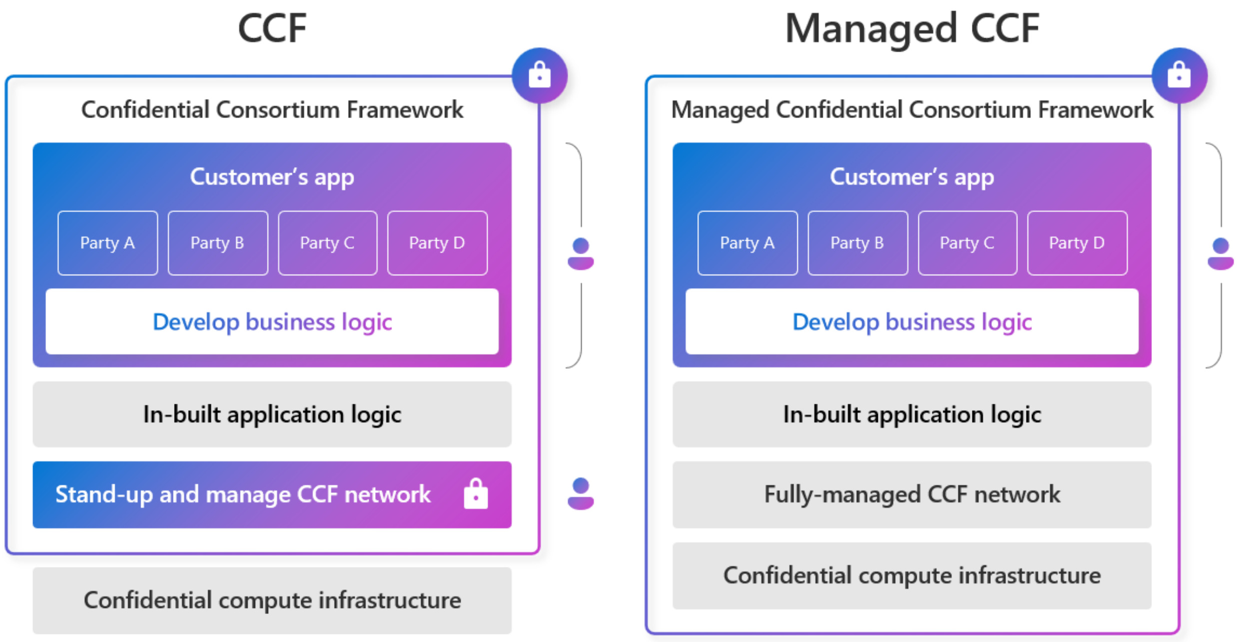
If you get either of the following error messages during the registration, your Tenant Id is not part of the private preview. Please contact us to fix it.

{"error":{"code":"InvalidResourceType","message":"The resource type could not be found in the namespace 'Microsoft.ConfidentialLedger' for api version '2022-09-08-preview'."}}



```
(FeatureRegistrationUnsupported) The feature 'ManagedCCF' does not support registration.
Code: FeatureRegistrationUnsupported
Message: The feature 'ManagedCCF' does not support registration.
```

How is Managed CCF different from CCF?


CCF
Confidential Consortium Framework
Customer's app
Party A
Party B
Party C
Party D
Develop business logic
In-built application logic
Stand-up and manage CCF network
Confidential compute infrastructure
Managed CCF
Managed Confidential Consortium Framework
Customer's app
Party A
Party B
Party C
Party D
Develop business logic
In-built application logic
Fully-managed CCF network
Confidential compute infrastructure

Deploying a Managed CCF instance from the Azure Portal

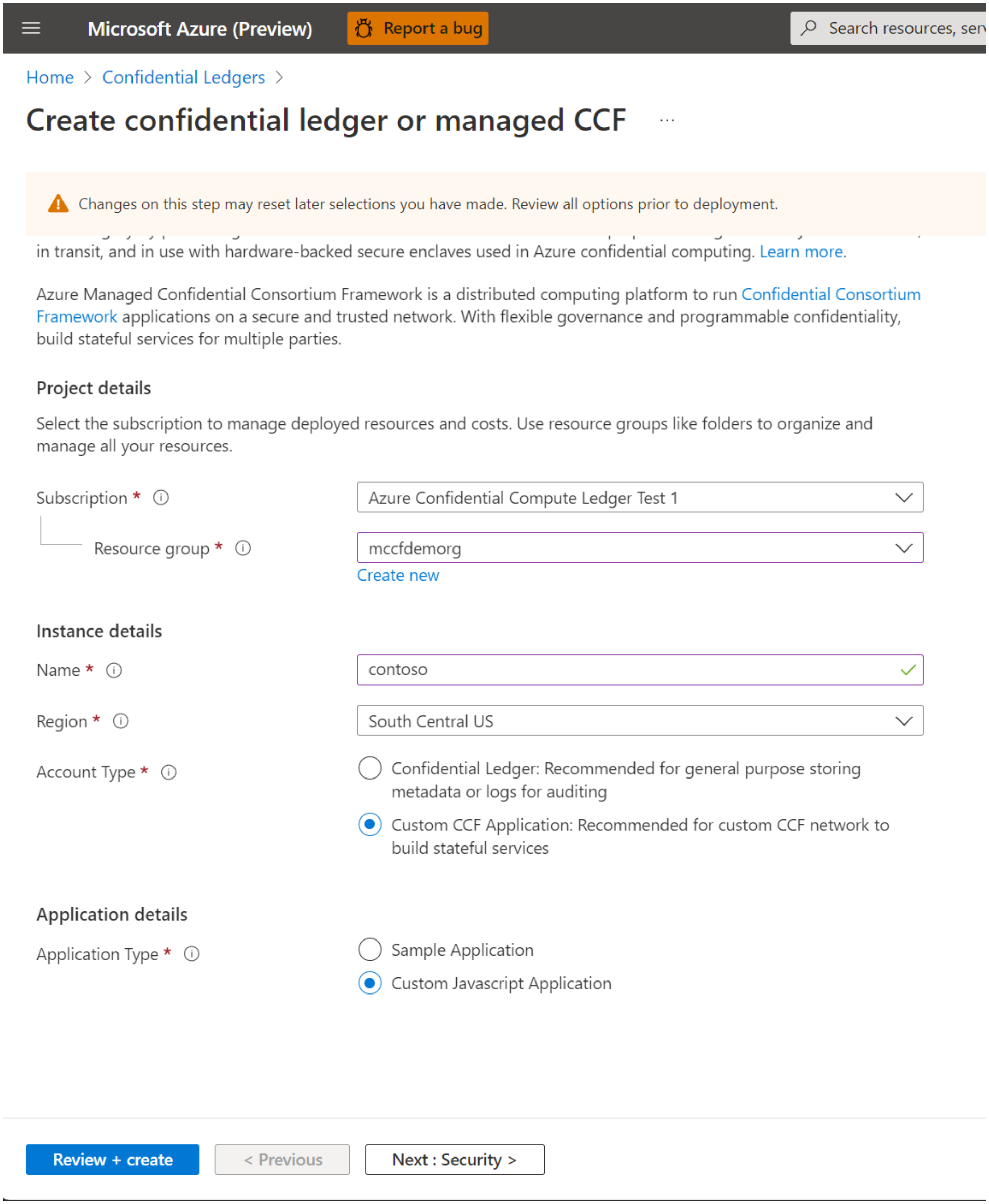
Customers can deploy a managed CCF instance from the Azure Portal or by using an ARM template. The following section will describe the steps to deploy a Managed CCF instance from the Azure Portal.

A managed CCF instance is governed by a consortium. The consortium members can propose changes to the service which can be accepted or rejected by other members of the consortium. Proposed changes include adding and removing members and users, deploying and updating an application, and updating the consortium. You need atleast one member identity certificate to create a Managed CCF instance and submit proposals. Before proceeding further, create member identity certificate(s) using the keygenerator.sh script available at microsoft/CCF: Confidential Consortium Framework (github.com). The instructions to use the script is available at Adding New Members - CCF documentation (microsoft.github.io).

1. Search for the Confidential Ledgers resource in the Azure Portal. **Note:** Please add these query strings to the portal URL to view the Managed CCF flight:

   &feature.Microsoft_Azure_ConfidentialLedger_managedccf=true

2. Select the correct subscription from the drop down.
3. In the Resource Group drop down, either select an existing Resource Group or create a new one.
4. Provide a unique name for the Managed CCF instance. It will be used to generate a host name.
5. Select **South Central US** from the Region drop down.
6. Select ‘Custom CCF Application’ for the Account Type.
7. Select the ‘Custom Javascript Application’ for the Application Type.



Home > Confidential Ledgers >

Create confidential ledger or managed CCF ...

Changes on this step may reset later selections you have made. Review all options prior to deployment.

in transit, and in use with hardware-backed secure enclaves used in Azure confidential computing. Learn more.

Azure Managed Confidential Consortium Framework is a distributed computing platform to run Confidential Consortium Framework applications on a secure and trusted network. With flexible governance and programmable confidentiality, build stateful services for multiple parties.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure Confidential Compute Ledger Test 1
Resource group *	mccfdemorg
	Create new

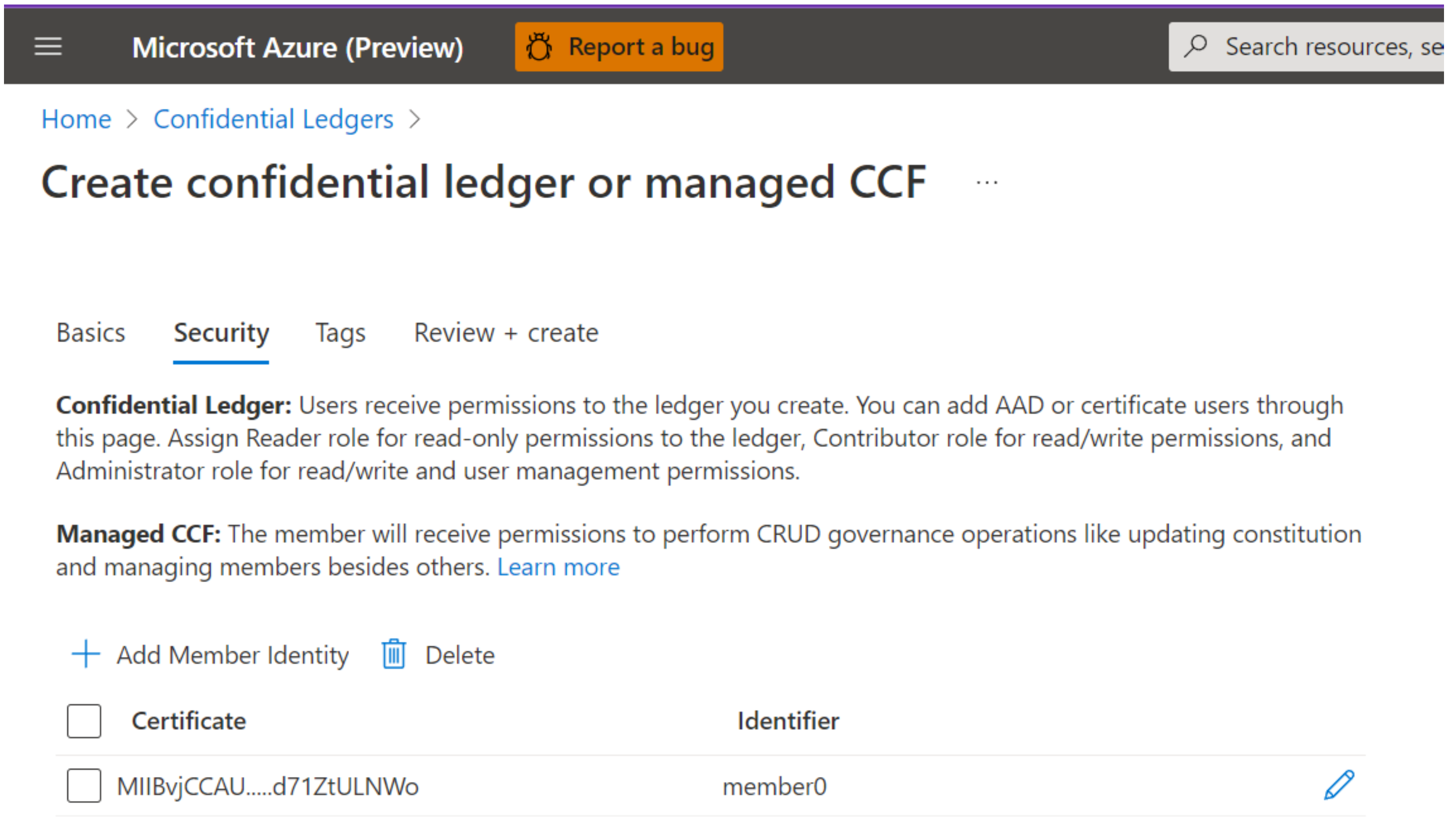
Instance details

Name *	contoso
Region *	South Central US
Account Type *	Confidential Ledger: Recommended for general purpose storing metadata or logs for auditing [✓ Custom CCF Application: Recommended for custom CCF network to build stateful services]

Application details

Application Type *	Sample Application [✓ Custom Javascript Application]

8. Click the Next button.
9. Click the ‘Add Member Identity’ button. A new window will open.
10. Paste the contents of the public certificate in PEM format and click Submit.
11. Repeat steps 9 and 10 to add more members. At least one member's identity is required.



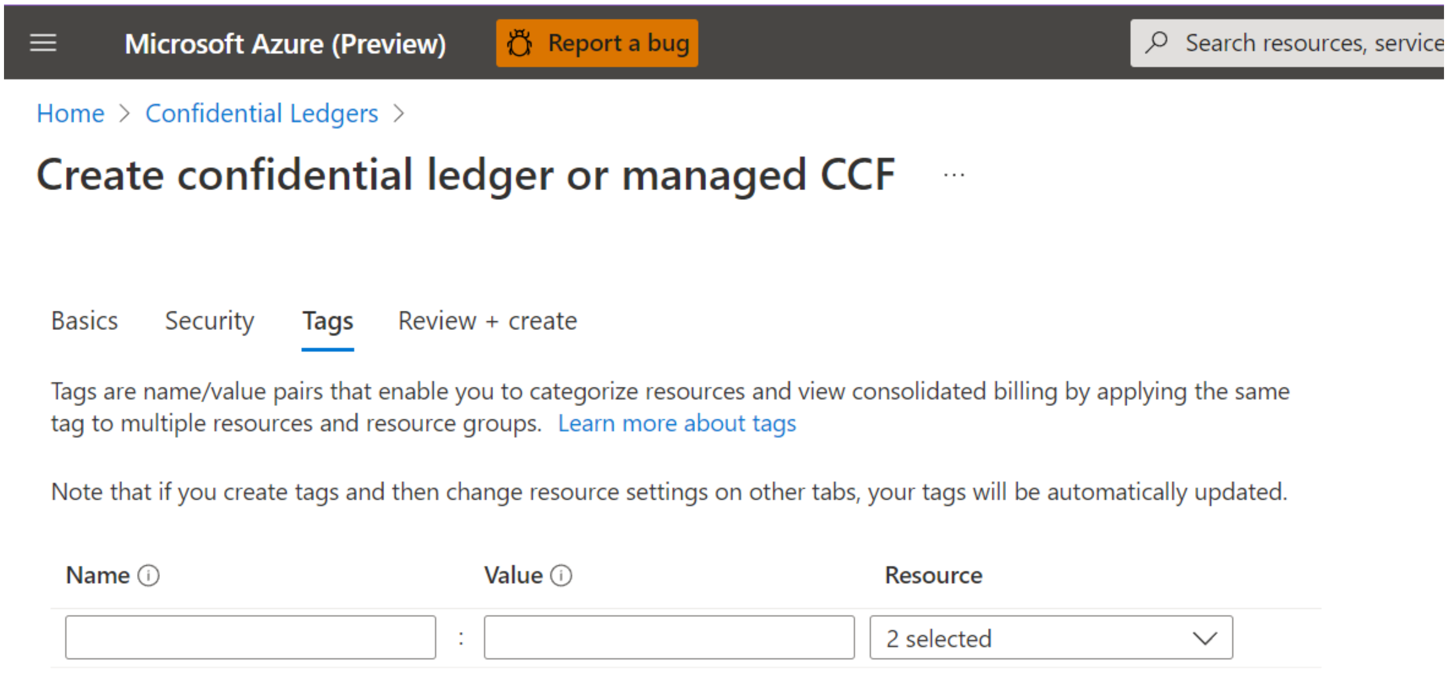
Create confidential ledger or managed CCF

Confidential Ledger: Users receive permissions to the ledger you create. You can add AAD or certificate users through this page. Assign Reader role for read-only permissions to the ledger, Contributor role for read/write permissions, and Administrator role for read/write and user management permissions.

Managed CCF: The member will receive permissions to perform CRUD governance operations like updating constitution and managing members besides others. Learn more

Certificate	Identifier
MIIBvjCCAU.....d71ZtULNWo	member0

12. Click the Next button.
13. Add Tags as needed and click the Next button.


Microsoft Azure (Preview)
Report a bug
Search resources, service
Home > Confidential Ledgers >

Create confidential ledger or managed CCF

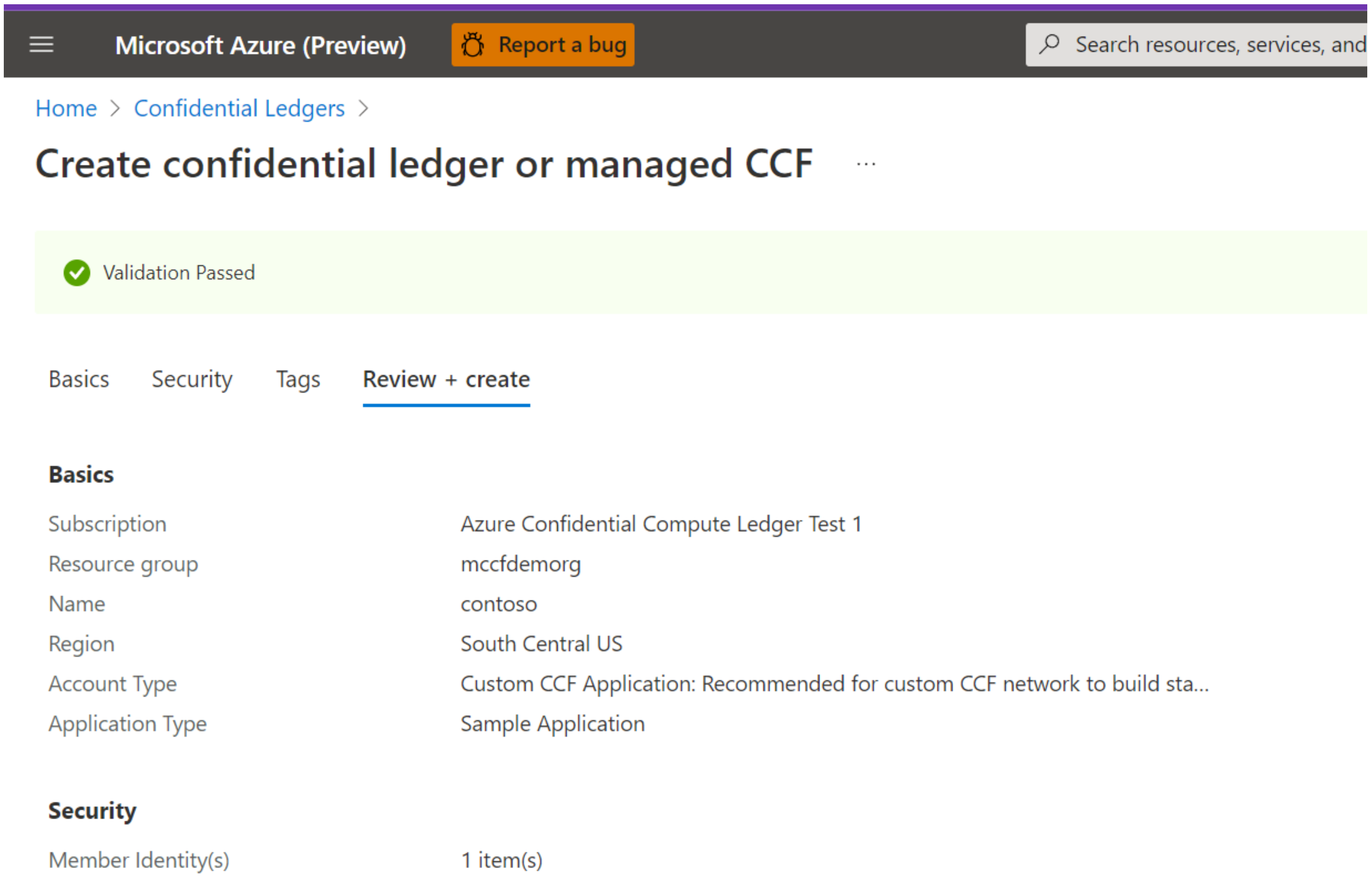
Basics
Security
Tags
Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. Learn more about tags

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name	Value	Resource
		2 selected

14. Review the information and click Create.


Microsoft Azure (Preview)
Report a bug
Search resources, services, and
Home > Confidential Ledgers >

Create confidential ledger or managed CCF

Validation Passed

Basics | Security | Tags | Review + create

Basics

Subscription	Azure Confidential Compute Ledger Test 1
Resource group	mccfdemorg
Name	contoso
Region	South Central US
Account Type	Custom CCF Application: Recommended for custom CCF network to build sta...
Application Type	Sample Application

Security

Member Identity(s)	1 item(s)

When the instance is ready, make a note of the Identity Service endpoint and the Managed CCF instance endpoint displayed on the screen. The next step is to deploy the JavaScript/TypeScript application.

Deploy a Custom Application

Prerequisites:

- *This section requires a Linux shell.*
- *The scripts used in this section are available at CCF/python/utils at main · microsoft/CCF (github.com)*
- *The managed CCF endpoint is https://contoso.confidential-ledger.azure.com*

This section will show the steps required to deploy a custom JavaScript banking application. The source code is available at https://github.com/microsoft/ccf-app-samples/tree/main/banking-app. The application supports certificate-based user authentication and ability to create accounts, deposit and transfer funds and view the balance.

Step 1: Download the Service Identity

Download the service identity certificate from the URI https://identity.confidential-ledger.core.azure.com/ledgerIdentity/contoso and extract the value in the ledgerTlsCertificate field. Remove the new line characters and save them to a file called service_cert.pem. The contents of the service_cert.pem should be as follows.

```
-----BEGIN CERTIFICATE-----
MIIBuDCCAT6gAwIBAgIRAOsSYTZ5AaxLnXzm/TnLTQgwCgYIKoZIzj0EAwMwFjEU
MBIGA1UEAwwLQ0NGIE5ldHdvcmswHhcNMjMwMjE2MTUyNzU2WhcNMjMwNTE3MTUy
NzU1WjAWMRQwEgYDVQQDDAtDQ0YgTmV0d29yazB2MBAGByqGSM49AgEGBSuBBAAi
….
AwMDaAAwZQIxAOEAmx5OFAt1aDfSsdlUoAsxcudtBYLmYI3oA4RuX+i/2XtxWFk7
…
uXRcBd2MA0uMRBsr
-----END CERTIFICATE-----
```

Step 2: Activate the Member(s)

The members (that were added from the Portal) will be in the ‘Accepted’ state. Use the commands below to activate them. The following commands assume that two members, namely member0 and member1 were added in the Portal.

```
### Activate Member0 ###
>curl https://contoso.confidential-ledger.azure.com/gov/ack/update_state_digest -X POST
--cacert service_cert.pem --key member0_privk.pem --cert member0_cert.pem --silent >
request.json

>./scurl.sh https://contoso.confidential-ledger.azure.com/gov/ack  --cacert service_cert.pem
--signing-key member0_privk.pem --signing-cert member0_cert.pem --header "Content-Type:
application/json" --data-binary @request.json

### Activate Member1 ###
>>curl https://contoso.confidential-ledger.azure.com/gov/ack/update_state_digest -X POST
--cacert service_cert.pem --key member1_privk.pem --cert member1_cert.pem --silent >
request.json

>./scurl.sh https://contoso.confidential-ledger.azure.com/gov/ack  --cacert service_cert.pem
--signing-key member1_privk.pem --signing-cert member1_cert.pem --header "Content-Type:
application/json" --data-binary @request.json
```

After the members are activated, the next step is to deploy the banking application.

Step 3: Deploy the Banking Application

Managed CCF requires applications to be converted into an application bundle for deployment. The instructions are available at JavaScript Application Bundle - CCF documentation (microsoft.github.io). The following commands will show how to deploy the application bundle.

```
### Submit the application bundle available in set_js_app.json ###

>proposal_id=`./scurl.sh --url https://contoso.confidential-ledger.azure.com/gov/proposals
--cacert service_cert.pem --signing-key member0_privk.pem --signing-cert member0_cert.pem -
-data-binary @set_js_app.json -H "content-type: application/json"|jq -r '.proposal_id'`

### Member0 accepts the proposal ###
>./scurl.sh https://contoso.confidential-
ledger.azure.com/gov/proposals/${proposal_id}/ballots --cacert service_cert.pem --signing-
key member0_privk.pem --signing-cert member0_cert.pem --data-binary @vote_accept.json -H
"content-type: application/json"

### Member1 accepts the proposal ###
>./scurl.sh https://contoso.confidential-
ledger.azure.com/gov/proposals/${proposal_id}/ballots --cacert service_cert.pem --signing-
key member1_privk.pem --signing-cert member1_cert.pem --data-binary @vote_accept.json -H
"content-type: application/json"
```

Step 4: Add User(s)

Adding a user is a two-step process. The first step is to propose a new user followed by a voting on the proposal. We will add two users, namely user0 and user1.

```
### Create a public and private key pair for user0 ###
>user0_id=$(openssl x509 -in "user0_cert.pem" -noout -fingerprint -sha256 | cut -d "=" -f 2 |
sed 's/://g' | awk '{print tolower($0)}')

### Create set_user0.json file with the public certificate of the user ###
{
  "actions": [
    {
      "name": "set_user",
      "args": {
        "cert": "-----BEGIN CERTIFICATE-----\... n7c3AgKUrAxzq\n-----END CERTIFICATE-----\n"
      }
    }
  ]
}

### create vote_accept.json ###
{
  "ballot": "export function vote (proposal, proposerId) { return true }"
}

### Member0 proposes the addition of user0 ###
>proposal_id=`./scurl.sh --url https://contoso.confidential-ledger.azure.com/gov/proposals
--cacert service_cert.pem --signing-key member0_privk.pem --signing-cert member0_cert.pem --
data-binary @set_user0.json -H "content-type: application/json"|jq -r '.proposal_id'`

### Member0 accepts the proposal ###
>./scurl.sh https://contoso.confidential-
ledger.azure.com/gov/proposals/${proposal_id}/ballots --cacert service_cert.pem --signing-key
member0_privk.pem --signing-cert member0_cert.pem --data-binary @vote_accept.json -H
"content-type: application/json"

### Member1 accepts the proposal ###
>./scurl.sh https://contoso.confidential-
ledger.azure.com/gov/proposals/${proposal_id}/ballots --cacert service_cert.pem --signing-key
member1_privk.pem --signing-cert member1_cert.pem --data-binary @vote_accept.json -H
"content-type: application/json"
```

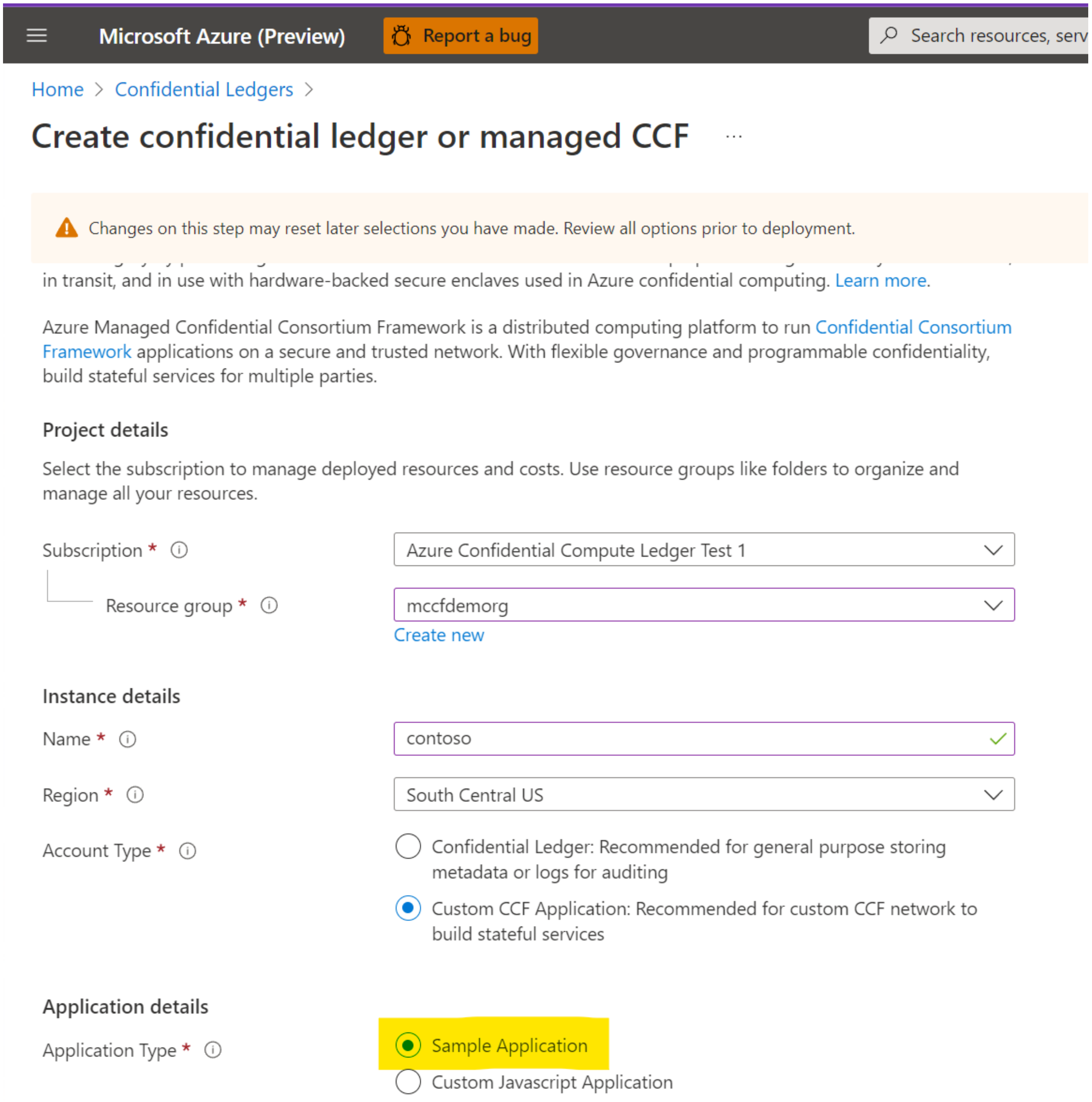
We have submitted the application and added users to it. The next step is to submit transactions.

Step 4: Submit Transactions

The following commands will create two accounts, namely *current_account* and *savings_account* for user0 and user1 respectively. Next, we will deposit $100 into the current_account and transfer $40 from the current_account to the savings_account.

```
### Create current_acccount for user 0 ###

>curl https://contoso.confidential-ledger.azure.com/app/account/${user0_id}/current_account -
X PUT --cacert service_cert.pem --cert member0_cert.pem --key member0_privk.pem

### Create savings_account for user 1 ###

>curl https://contoso.confidential-ledger.azure.com/app/account/${user1_id}/savings_account -
X PUT --cacert service_cert.pem --cert member0_cert.pem --key member0_privk.pem

# Deposit $100 to current_account ###

>curl https://contoso.confidential-ledger.azure.com/app/deposit/${user0_id}/current_account -
X POST --cacert service_cert.pem --cert member0_cert.pem --key member0_privk.pem -H "Content-
Type: application/json" --data-binary '{ "value": 100 }'

### Transfer $40 from user0 to user1 ###

>curl https://contoso.confidential-ledger.azure.com/app/transfer/current_account -X POST
--cacert service_cert.pem --cert user0_cert.pem --key user0_key.key -H "Content-Type:
application/json" --data-binary "{ \"value\": 40, \"user_id_to\": \"${user1_id}\",
\"account_name_to\": \"savings_account\" }"

### Check the balance ###

>curl https://contoso.confidential-ledger.azure.com/app/balance/current_account -X GET
--cacert service_cert.pem --cert user0_cert.pem --key user0_key.key

>curl https://contoso.confidential-ledger.azure.com/app/balance/savings_account -X GET
--cacert service_cert.pem --cert user1_cert.pem --key user1_key.key
```

As a convenience, the banking application can be deployed from the Azure Portal by choosing the '*Sample Application*' option. When chosen, step # 3 (*Deploy the Banking Application*) is redundant. The rest of the steps remain unchanged.



Microsoft Azure (Preview)

Report a bug

Search resources, serv

Home > Confidential Ledgers >

Create confidential ledger or managed CCF ...

Changes on this step may reset later selections you have made. Review all options prior to deployment.

in transit, and in use with hardware-backed secure enclaves used in Azure confidential computing. Learn more.

Azure Managed Confidential Consortium Framework is a distributed computing platform to run Confidential Consortium Framework applications on a secure and trusted network. With flexible governance and programmable confidentiality, build stateful services for multiple parties.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure Confidential Compute Ledger Test 1
Resource group *	mccfdemorg
	Create new

Instance details

Name *	contoso
Region *	South Central US
Account Type *	Confidential Ledger: Recommended for general purpose storing metadata or logs for auditing [✓ Custom CCF Application: Recommended for custom CCF network to build stateful services]

Application details

Application Type *	[✓ Sample Application] Custom Javascript Application

Deploy a Managed CCF instance Using an ARM Template

An ARM template and the associated parameters file is available in the Appendix section of the document. It can be used to deploy a Managed CCF instance using the following commands.

```
### Login to Azure and set the active subscription ###
>az login
>az account set -s <<subscription id>>

### Initiate a Managed CCF resource deployment ###
>az deployment group create -g <<resource group name>> --template-file template.json --
parameters parameters.json
```

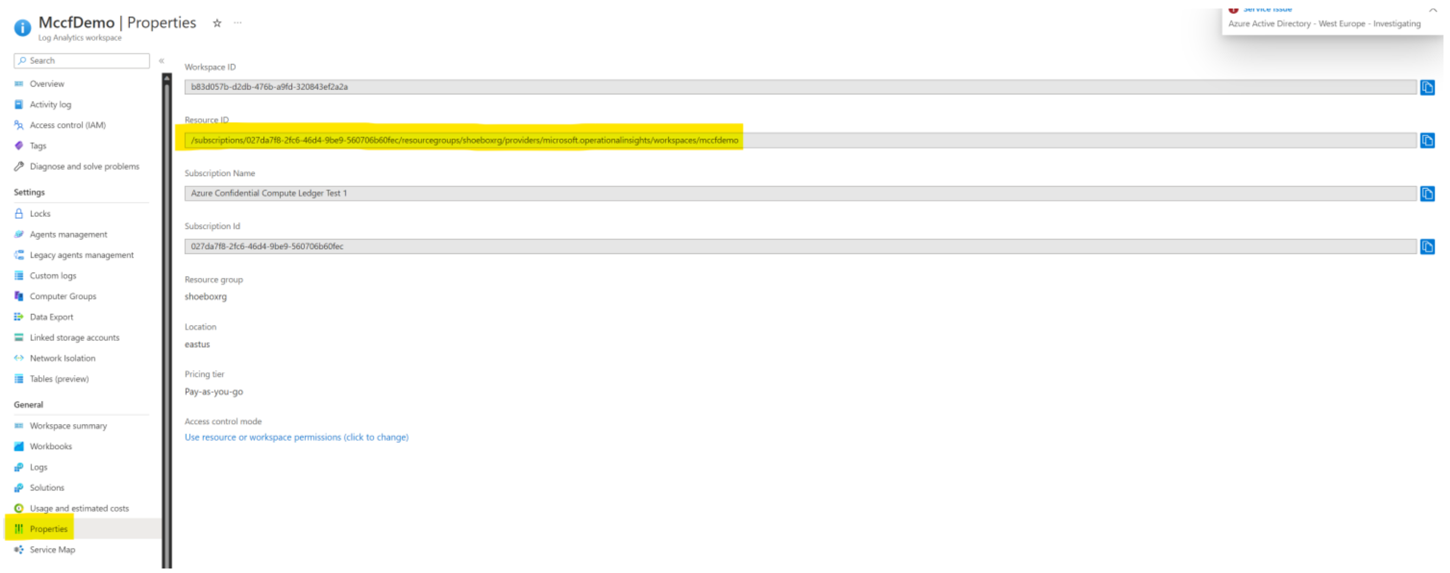
When it is finished, you will find the Managed CCF resource inside the resource group that was supplied in the command.

View the Application Logs

The logs generated from the application can be viewed in the subscription where the Managed CCF instance is deployed. There is cost associated with enabling diagnostics on monitored resources. To determine the pricing for different regions, refer to this resource [Pricing - Azure Monitor | Microsoft Azure](#).

1. Create a Log Analytics workspace by following the instructions available at [Log Analytics workspace overview - Azure Monitor | Microsoft Learn](#)

2. After a workspace is created, make a note of the Resource ID from the Properties page.


MccfDemo | Properties
Log Analytics workspace

Azure Active Directory - West Europe - Investigating

Workspace ID
b83d057b-d2db-476b-a9fd-320843ef2a2a

Resource ID
/subscriptions/027da7f8-2fc6-46d4-9be9-560706b60fec/resourcegroups/shoeboxrg/providers/microsoft.operationalinsights/workspaces/mccfdemo

Subscription Name
Azure Confidential Compute Ledger Test 1

Subscription Id
027da7f8-2fc6-46d4-9be9-560706b60fec

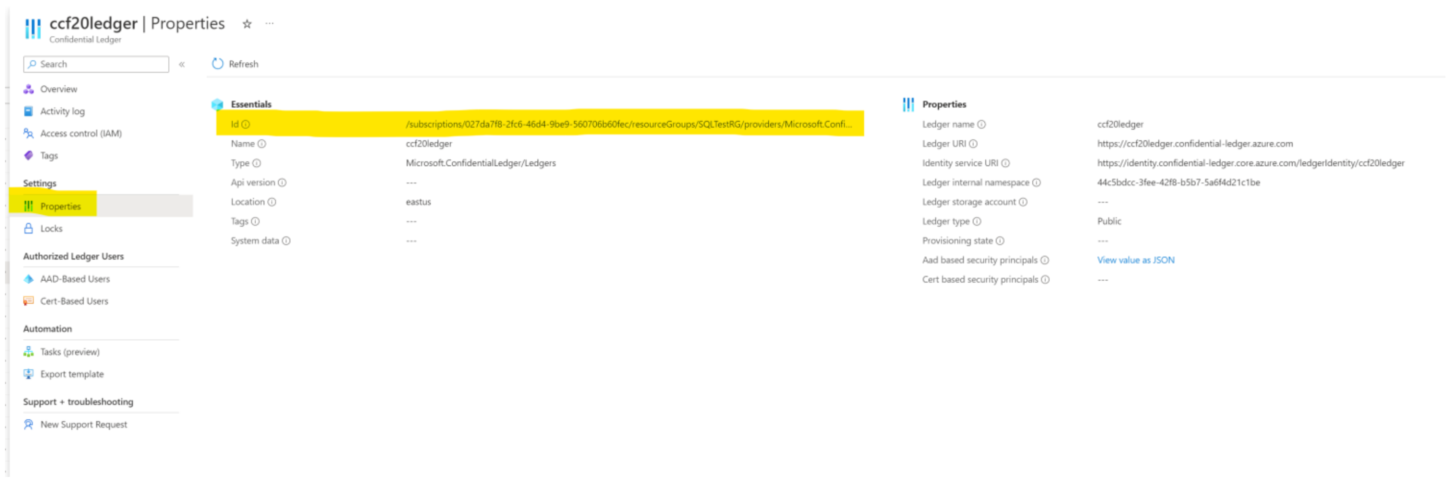
Resource group
shoeboxrg

Location
eastus

Pricing tier
Pay-as-you-go

Access control mode
Use resource or workspace permissions (click to change)

3. Open the Managed CCF instance. Make a note of the Resource ID from the Properties page.



Essentials

Id	/subscriptions/027da7f8-2fc6-46d4-9be9-560706b60fec/resourceGroups/SQLTestRG/providers/Microsoft.Confi...
Name	ccf20ledger
Type	Microsoft.ConfidentialLedger/Ledgers
Api version	---
Location	eastus
Tags	---
System data	---

Properties

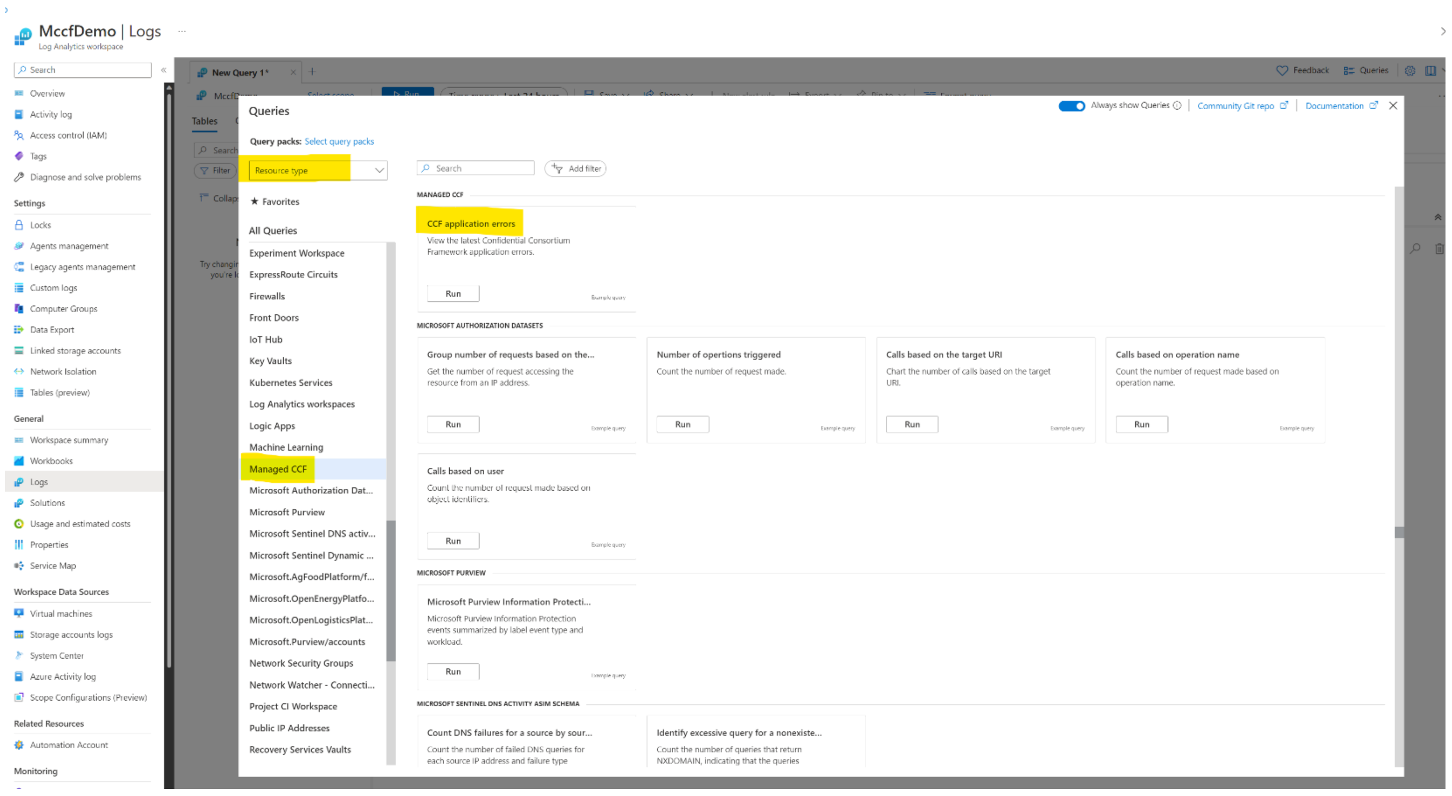
Ledger name	ccf20ledger
Ledger URI	https://ccf20ledger.confidential-ledger.azure.com
Identity service URI	https://identity.confidential-ledger.core.azure.com/ledgerIdentity/ccf20ledger
Ledger internal namespace	44c5bdcc-3fee-42f8-b5b7-5a6f4d21c1be
Ledger storage account	---
Ledger type	Public
Provisioning state	---
Aad based security principals	View value as JSON
Cert based security principals	---

4. Enable Diagnostics on the Managed CCF instance using the following command.

```
az monitor diagnostic-settings create --name <<a unique name>> --resource <<resource id of
the Managed CCF resource>> --workspace <<resource id of the log analytics workspace>> --logs
'[
     {
       "category": "applicationlogs",
       "enabled": true,
       "retentionPolicy": {
         "enabled": false,
         "days": 0
       }
     }
   ]'
```

5. Wait for a few minutes for the application logs to appear in the Log Analytics workspace.

6. Open the Logs page and group the queries by Resource type from the drop down. Navigate to the 'Azure Managed CCF' resource.



MANAGED CCF

CCF application errors
View the latest Confidential Consortium Framework application errors.

MICROSOFT AUTHORIZATION DATASETS

Group number of requests based on the...
Get the number of request accessing the resource from an IP address.

Number of opertions triggered
Count the number of request made.

Calls based on the target URI
Chart the number of calls based on the target URI.

Calls based on operation name
Count the number of request made based on operation name.

Calls based on user
Count the number of request made based on object identifiers.

MICROSOFT PURVIEW

Microsoft Purview Information Protecti...
Microsoft Purview Information Protection events summarized by label event type and workload.

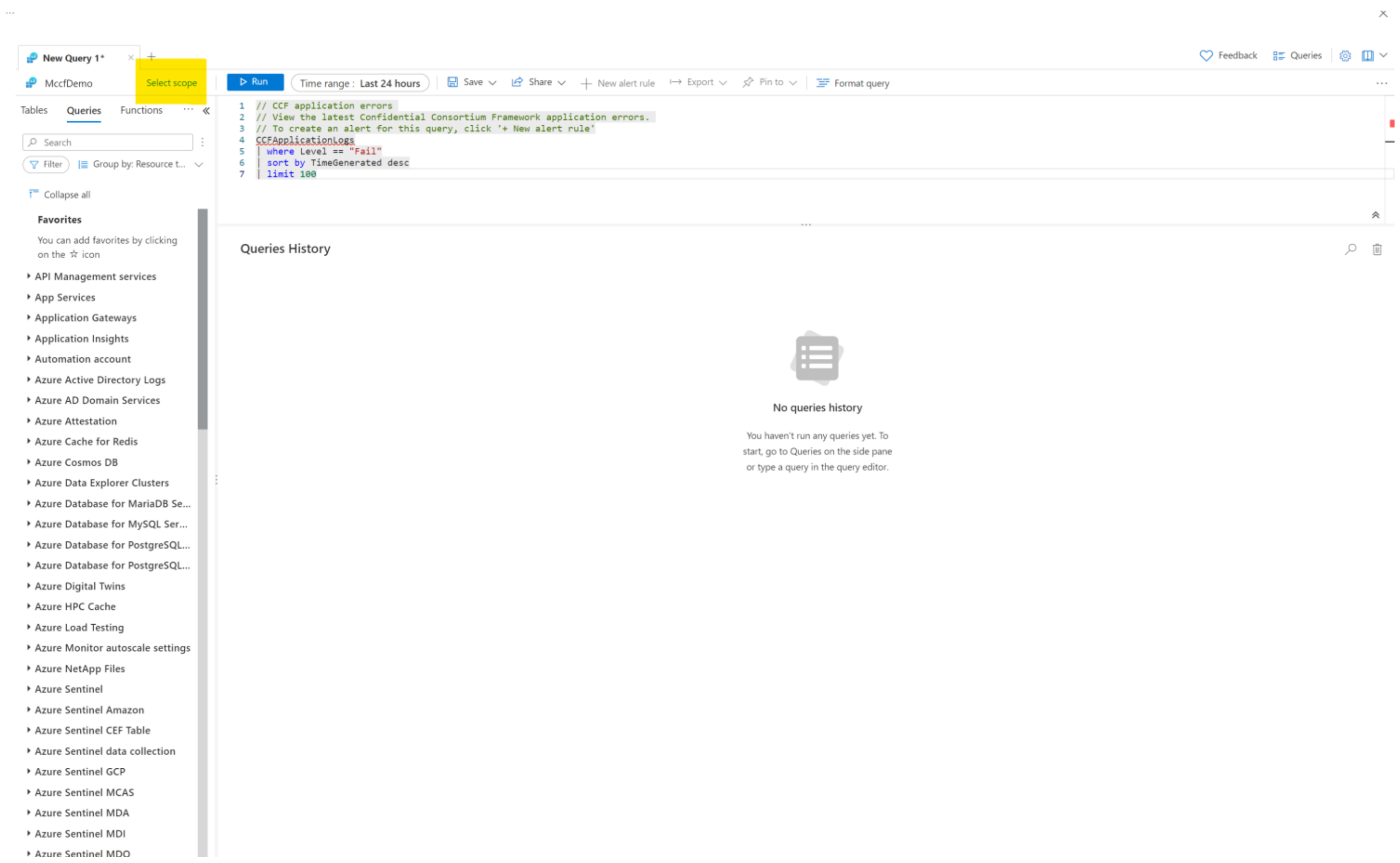
MICROSOFT SENTINEL DNS ACTIVITY ASIM SCHEMA

Count DNS failures for a source by sour...
Count the number of failed DNS queries for each source IP address and failure type

Identify excessive query for a nonexiste...
Count the number of queries that return NXDOMAIN, indicating that the queries

7. A sample KQL query called ‘CCF application errors’ is provided to view the errors emitted from the application. Click the Load to editor button.

8. Before executing the query, adjust the scope using the Select scope button.



```
// CCF application errors
// View the latest Confidential Consortium Framework application errors.
// To create an alert for this query, click '+ New alert rule'
CCFApplicationLogs
| where Level == "Fail"
| sort by TimeGenerated desc
| limit 100
```

For a complete reference on the Keyword Query Language syntax (KQL), refer to the documentation at Keyword Query Language (KQL) syntax reference | Microsoft Learn.

Key Contacts

Product Lead: Shubhra Sinha | shubhra.sinha@microsoft.com
Engineering Lead: Bhuvaneshwari Krishnamurthi | bhkrishn@microsoft.com
Private Preview PM: Kristina Quick | kquick@microsoft.com Sudan Zhuang | Sudan.Zhuang@microsoft.com

Thank you! Your participation is a vital part of our Cloud + AI Security product development process.

Microsoft

Appendix

ARM Template file (template.json)

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "resourceName": {
            "type": "String",
            "metadata": {
                "description": "Resource Name"
            }
        },
        "resourceType": {
            "defaultValue": "",
            "allowedValues": [
                "Public",
                "Private",
                "App"
            ],
            "type": "String",
            "metadata": {
                "description": "Values are App, Public or Private"
            }
        },
        "location": {
            "type": "String",
            "metadata": {
                "description": "Azure Region"
            }
        },
        "aclAadBasedSecurityPrincipals": {
            "defaultValue": [],
            "type": "Array",
            "metadata": {
                "description": "AAD-based users"
            }
        },
        "aclCertBasedSecurityPrincipals": {
            "defaultValue": [],
            "type": "Array",
            "metadata": {
                "description": "Cert-based users"
            }
        },
        "mccfAppSourceUri": {
            "defaultValue": "",
            "type": "String",
            "metadata": {
                "description": "MCCF App type"
            }
        },
        "mccfAppLanguage": {
            "defaultValue": "JS",
            "allowedValues": [
                "JS",
                "CPP"
            ],
            "type": "String",
            "metadata": {
                "description": "MCCF Language Runtime"
            }
        },
        "mccfMemberBasedSecurityPrincipals": {
            "defaultValue": [],
            "type": "Array",
            "metadata": {
                "description": "Member identities"
            }
        },
        "tags": {
            "type": "Object",
            "metadata": {
                "description": "Additional Properties"
            }
        }
    },
    "variables": {
        "copy": [
            {
                "name": "certBasedSecurityPrincipals",
                "count": "[length(parameters('aclCertBasedSecurityPrincipals'))]",
                "input": {
                    "cert":
"[last(take(parameters('aclCertBasedSecurityPrincipals'),copyIndex('certBasedSecurityPrincipa
ls',1))).cert]",
                    "ledgerRoleName":
"[last(take(parameters('aclCertBasedSecurityPrincipals'),copyIndex('certBasedSecurityPrincipa
ls',1))).ledgerRoleName]"
                }
            },
            {
                "name": "memberBasedSecurityPrincipals",
                "count": "[length(parameters('mccfMemberBasedSecurityPrincipals'))]",
                "input": {
                    "certificate":
"[last(take(parameters('mccfMemberBasedSecurityPrincipals'),copyIndex('memberBasedSecurityPri
ncipals',1))).cert]",
                    "encryptionkey":
"[last(take(parameters('mccfMemberBasedSecurityPrincipals'),copyIndex('memberBasedSecurityPri
ncipals',1))).encryptionKey]",
                    "tags":
"[last(take(parameters('mccfMemberBasedSecurityPrincipals'),copyIndex('memberBasedSecurityPri
ncipals',1))).tags]"
                }
            }
        ]
    },
    "functions": [],
    "resources": [
        {
            "type": "Microsoft.ConfidentialLedger/ManagedCCFs",
            "apiVersion": "2022-09-08-preview",
            "name": "[parameters('resourceName')]",
            "location": "[parameters('location')]",
            "tags":
"[if(contains(parameters('tags'),'Microsoft.ConfidentialLedger/ManagedCCFs'),parameters('tags
')['Microsoft.ConfidentialLedger/ManagedCCFs'],json('{}'))]",
            "properties": {
                "deploymentType": {
                    "languageRuntime": "[parameters('mccfAppLanguage')]",
                    "appSourceUri": "[parameters('mccfAppSourceUri')]"
                },
                "memberIdentityCertificates": "[variables('memberBasedSecurityPrincipals')]"
            },
            "condition": "[equals(parameters('resourceType'),'App')]"
        },
        {
            "type": "Microsoft.ConfidentialLedger/ledgers",
            "apiVersion": "2020-12-01-preview",
            "name": "[parameters('resourceName')]",
            "location": "[parameters('location')]",
            "tags":
"[if(contains(parameters('tags'),'Microsoft.ConfidentialLedger/ledgers'),parameters('tags')['
Microsoft.ConfidentialLedger/ledgers'],json('{}'))]",
            "properties": {
                "ledgerType": "[parameters('resourceType')]",
                "aadBasedSecurityPrincipals":
"[parameters('aclAadBasedSecurityPrincipals')]",
                "certBasedSecurityPrincipals": "[variables('certBasedSecurityPrincipals')]"
            },
            "condition": "[or(equals(parameters('resourceType'), 'Public'),
equals(parameters('resourceType'), 'Private'))]"
        }
    ],
    "outputs": {}
}
```

Parameters file (parameters.json)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
      "resourceName": {
          "value": "contoso"
      },
      "resourceType": {
          "value": "App"
      },
      "location": {
          "value": "southcentralus"
      },
      "aclAadBasedSecurityPrincipals": {
          "value": []
      },
      "aclCertBasedSecurityPrincipals": {
          "value": []
      },
      "mccfAppSourceUri": {
          "value": "sample"
      },
      "mccfAppLanguage": {
          "value": "JS"
      },
      "mccfMemberBasedSecurityPrincipals": {
        "value": [
            {
                "cert": "-----BEGIN CERTIFICATE-----\nMIIBvj...d71ZtULNWo\n-----END
CERTIFICATE-----",
                "shortCert": "MIIBvjCCAU.....d71ZtULNWo",
                "encryptionKey": "",
                "tags": {
                    "identifier": "member1",
                    "shortCert": "MIIBvjCCAU.....d71ZtULNWo"
                }
            }
        ]
      },
      "tags": {
          "value": {}
      }
  }
}
```