Azure Cloud Billing Integration

Connecting your Azure account to Kubecost allows you to view Kubernetes metrics side-by-side with out-of-cluster (OOC) costs (e.g. Azure Database Services). Additionally, it allows Kubecost to reconcile measured Kubernetes spend with your actual Azure bill. This gives teams running Kubernetes a complete and accurate picture of costs. For more information, read Cloud Billing Integrations and this blog post.

To configure Kubecost's Azure Cloud Integration, you will need to set up daily exports of cost reports to Azure storage. Kubecost will then access your cost reports through the Azure Storage API to reconcile Kubernetes costs and display your cloud cost data.

A GitHub repository with sample files used in below instructions can be found here.

Step 1: Export Azure cost report

Azure Cost Management is not available for all offer types. Review the Azure documentation, Understand Cost Management data, to learn more.

Follow Azure's Create and Manage Exported Data tutorial to export cost reports and use the following configuration:

- For 'Type of data' select `Cost and usage details (amortized)`
- Provide a meaningful name for the 'Export name'
- For 'Dataset version' select `2021-10-01`
- For 'Frequency' select `Daily export of month-to-date costs`
- For 'Format' select `CSV`
- For 'Compression' select `None`

Take note of the Storage Account name and Container specified when choosing where to export the data to. Note that a successful cost export will require

`Microsoft.CostManagementExports` to be registered in your subscription.

It will take a few hours to generate the first report, after which Kubecost can use the Azure Storage API to pull that data.

Once the cost export has successfully executed, verify that a non-empty CSV file has been created at this path: `<STORAGE_ACCOUNT>/<CONTAINER_NAME>/<OPTIONAL_CONTAINER_PATH>/<COST_EXPORT_NAME>/<DATE_RANGE>/<CSV_FILE>`.

If you have sensitive data in an existing Azure Storage account, it is recommended to create a separate Azure Storage account to store your cost data export.

For more granular billing data it is possible to scope Azure cost exports to resource groups, management groups, departments, or enrollments. AKS clusters will create their own resource groups which can be used. This functionality can then be combined with Kubecost multi-cloud to ingest multiple scoped billing exports.

Step 2: Provide access to Azure Storage API

Obtain the following values from Azure to provide to Kubecost. These values can be located in the Azure Portal by selecting *Storage Accounts*, then selecting your specific Storage account for details.

- `azureSubscriptionID` is the "Subscription ID" belonging to the Storage account which stores your exported Azure cost report data.
- `azureStorageAccount` is the name of the Storage account where the exported Azure cost report data is being stored.
- `azureStorageAccessKey` can be found by selecting *Access keys* in your Storage account left navigation under "Security + networking". Using either of the two keys will work.

- `azureStorageContainer` is the name that you chose for the exported cost report when you set it up. This is the name of the container where the CSV cost reports are saved in your Storage account.
- `azureContainerPath` is an optional value which should be used if there is more than one billing report that is exported to the configured container. The path provided should have only one billing export because Kubecost will retrieve the most recent billing report for a given month found within the path.
- `azureCloud` is an optional value which denotes the cloud where the storage account exist, possible values are `public` and `gov`. The default is `public`.

Next, create a JSON file which **must** be named *cloud-integration.json* with the following format:

```
{
    "azure": [
        {
            "azureSubscriptionID": "AZ_cloud_integration_subscriptionId",
            "azureStorageAccount": "AZ_cloud_integration_azureStorageAcco
            "azureStorageAccessKey": "AZ_cloud_integration_azureStorageAc
            "azureStorageContainer": "AZ_cloud_integration_azureStorageCo
            "azureContainerPath": "",
            "azureCloud": "public"
        }
    ]
}
```

Additional details about the `cloud-integration.json` file can be found in our multi-cloud integration doc.

Next, create the Secret:

```
$ kubectl create secret generic <SECRET_NAME> --from-file=cloud-
integration.json -n kubecost
```

Next, ensure the following are set in your Helm values:

```
kubecostProductConfigs:
  cloudIntegrationSecret: <SECRET_NAME>
```

Next, upgrade Kubecost via Helm:

```
$ helm upgrade kubecost kubecost/cost-analyzer -n kubecost -f values.yaml
```

You can verify a successful configuration by checking the following in the Kubecost UI:

- The Assets dashboard will be broken down by Kubernetes assets.
- The Assets dashboard will no longer show a banner that says "External cloud cost not configured".
- The Diagnostics page (via *Settings* > *View Full Diagnostics*) view will show a green checkmark under Cloud Integrations.

> If there are no in-cluster costs for a particular day, then there will not be out-of-cluster costs either

Troubleshooting and debugging

To troubleshoot a configuration that is not yet working:

- `$ kubectl get secrets -n kubecost` to verify you've properly configured `cloud-integration.json`.
- `$ helm get values kubecost` to verify you've properly set `.Values.kubecostProductConfigs.cloudIntegrationSecret`
- Verify that a non-empty CSV file has been created at this path in your Azure Portal Storage Account: `<STORAGE_ACCOUNT>/<CONTAINER_NAME>/<OPTIONAL_CONTAINER_PATH>/<COST_EXPORT_NAME>/<DATE_RANGE>/<CSV_FILE>`. Ensure new CSVs are being generated every day.
- When opening a cost report CSV, ensure that there are rows in the file that do not have a MeterCategory of “Virtual Machines” or “Storage” as these items are ignored because they are in cluster costs. Additionally, make sure that there are items with a UsageDateTime that matches the date you are interested in.

Azure Cloud Integration using Azure Workload Identity

Kubecost supports cloud integration via Azure Workload Identity. Refer to the Microsoft documentation to learn more about how to set up Azure Workload Identity in AKS.

For this tutorial, you will need the cluster name, resource group, federated identity credential name, and the Managed Identity Object ID.

Tutorial

1. Validate that OIDC is enabled on the Azure cluster.

```
$ export AKS_OIDC_ISSUER="$(az aks show -n $CLUSTER_NAME -g "${RESOURCE_G
https://westus.oic.<redacted>.azure.com/<redacted>
```

2. Assign the Storage Blob Data Contributor Role to the Managed Identity and scope it to the storage blob container resource that has the cost export. See this example:

```
az role assignment create --assignee "55555555-5555-5555-5555-55555555555
```

3. Create the federated credential between the Managed Identity and kubecost-cost-analyzer service account:

```
az identity federated-credential create --name ${FEDERATED_IDENTITY_CREDE
```

4. Create a JSON file which **must** be named *cloud-integration.json* with the following format:

```
{
    "azure": {
     "storage":[
      {
            "subscriptionID": "AZ_cloud_integration_subscriptionId",
            "account": "AZ_cloud_integration_azureStorageAccount",
            "container": "AZ_cloud_integration_azureStorageContainer",
            "path": "",
            "cloud": "public/gov",
            "authorizer":{
             "authorizerType": "AzureDefaultCredential"
            }
        }
     ]
   }
}
```

5. Create the secret.

```
$ kubectl create secret generic <SECRET_NAME> --from-file=cloud-integrati
```

6. Update the Helm *values.yaml* with the following and apply changes:

```
kubecostProductConfigs:
  cloudIntegrationSecret: <SECRET_NAME>
kubecostDeployment:
  labels:
    azure.workload.identity/use: "true"
serviceAccount:
  annotations:
    azure.workload.identity/client-id: $AZURE_CLIENT_ID
```

```
helm upgrade --install kubecost --repo https://kubecost.github.io/cost-an
```

Azure Rate Card Configuration

Kubecost needs access to the Microsoft Azure Billing Rate Card API to access accurate pricing data for your Kubernetes resources.

You can also get this functionality plus external costs by completing the full Azure billing integration.

Creating a custom Azure role

Start by creating an Azure role definition. Below is an example definition, replace `YOUR_SUBSCRIPTION_ID` with the Subscription ID where your Kubernetes cluster lives:

```
{
    "Name": "KubecostRole",
    "IsCustom": true,
    "Description": "Rate Card query role",
    "Actions": [
        "Microsoft.Compute/virtualMachines/vmSizes/read",
        "Microsoft.Resources/subscriptions/locations/read",
        "Microsoft.Resources/providers/read",
        "Microsoft.ContainerService/containerServices/read",
        "Microsoft.Commerce/RateCard/read"
    ],
    "AssignableScopes": [
        "/subscriptions/YOUR_SUBSCRIPTION_ID"
    ]
}
```

Save this into a file called *myrole.json*.

Next, you'll want to register that role with Azure:

```
az role definition create --verbose --role-definition @myrole.json
```

Creating an Azure service principal

Next, create an Azure service principal.

```
az ad sp create-for-rbac --name "KubecostAccess" --role "KubecostRole" -
-scope "/subscriptions/YOUR_SUBSCRIPTION_ID" --output json
```

Keep this information which is used in the *service-key.json* below.

Supplying Azure service principal details to Kubecost

Option 1: Via a Kubernetes Secret (Recommended)

Create a file called *service-key.json* and update it with the Service Principal details from the above steps:

```
{
    "subscriptionId": "<Azure Subscription ID>",
    "serviceKey": {
        "appId": "<Entra ID App ID>",
        "displayName": "KubecostAccess",
        "password": "<Entra ID Client Secret>",
        "tenant": "<Entra Tenant ID>"
    }
}
```

Next, create a Secret for the Azure Service Principal

When managing the service account key as a Kubernetes Secret, the secret must reference the service account key JSON file, and that file must be named `service-key.json`.

```
kubectl create secret generic azure-service-key -n kubecost --from-
file=service-key.json
```

Finally, set the `kubecostProductConfigs.serviceKeySecretName` Helm value to the name of the Kubernetes secret you created. We use the value `azure-service-key` in our examples.

Option 2: Via Helm values

In the Helm values file:

```
kubecostProductConfigs:
  azureSubscriptionID: <Azure Subscription ID>
  azureClientID: <Entra ID App ID>
  azureTenantID: <Entra Tenant ID>
  azureClientPassword: <Entra ID Client Secret>
  azureOfferDurableID: MS-AZR-0003P
  azureBillingRegion: US
  currencyCode: USD
  createServiceKeySecret: true
```

Or at the command line:

```
helm upgrade --install kubecost kubecost/cost-analyzer -n kubecost \
  --set kubecostProductConfigs.azureSubscriptionID=<Azure Subscription ID
  --set kubecostProductConfigs.azureClientID=<Entra ID App ID> \
  --set kubecostProductConfigs.azureTenantID=<Entra Tenant ID> \
  --set kubecostProductConfigs.azureClientPassword=<Entra ID Client Secre
  --set kubecostProductConfigs.azureOfferDurableID=MS-AZR-0003P \
  --set kubecostProductConfigs.azureBillingRegion=US
  --set kubecostProductConfigs.currencyCode=USD
  --set kubecostProductConfigs.createServiceKeySecret=true
```

Azure billing region, offer durable ID, and currency

Kubecost supports querying the Azure APIs for cost data based on the region, offer durable ID, and currency defined in your Microsoft Azure offer.

Those properties are configured with the following Helm values:

- `kubecostProductConfigs.azureBillingRegion`
- `kubecostProductConfigs.azureOfferDurableID`
- `kubecostProductConfigs.currencyCode`

Be sure to verify your billing information with Microsoft and update the above Helm values to reflect your bill to country, subscription offer durable ID/number, and currency.

See also

The following Microsoft documents are a helpful reference:

- Microsoft Azure Offer Details
- Azure Pricing FAQ
- Geographic availability and currency support for the commercial marketplace
- Azure Portal > Cost Management + Billing > Billing Account Properties
- Understand Cost Management data