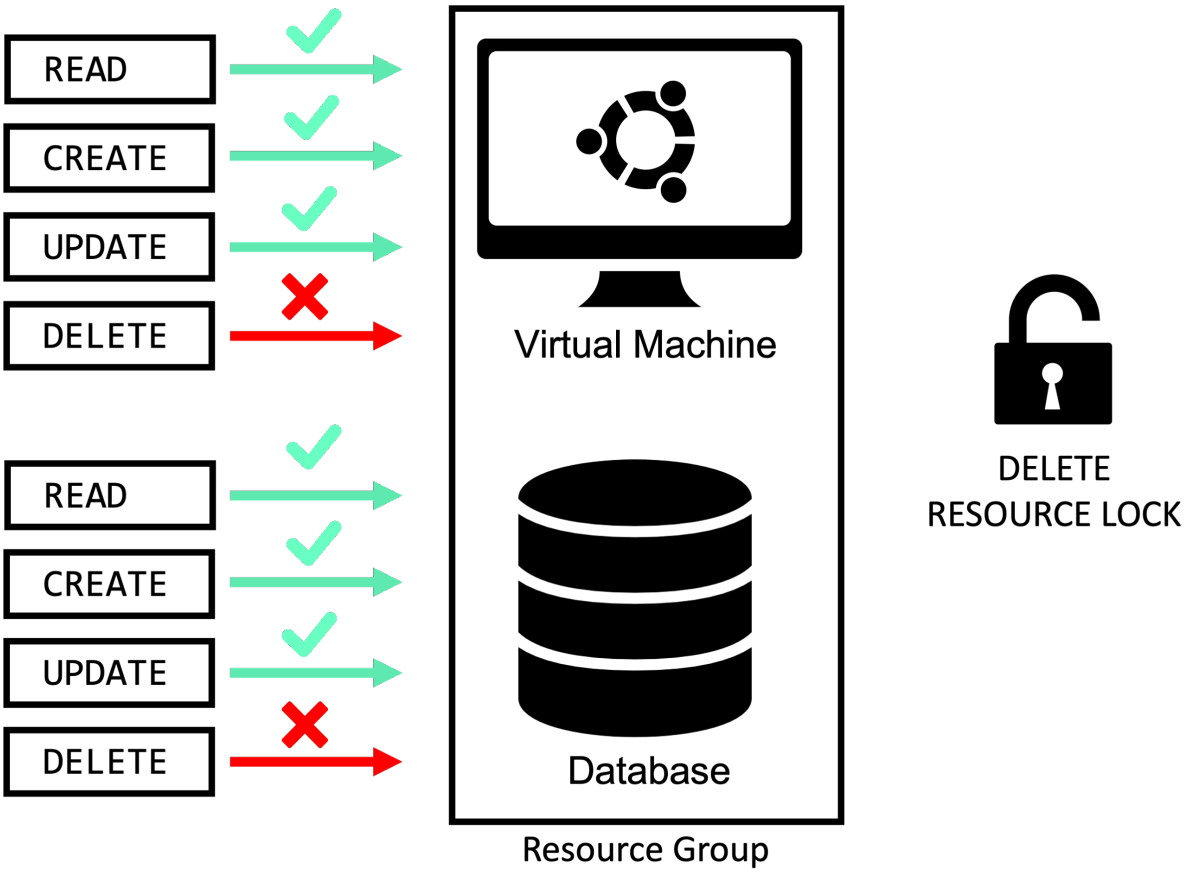
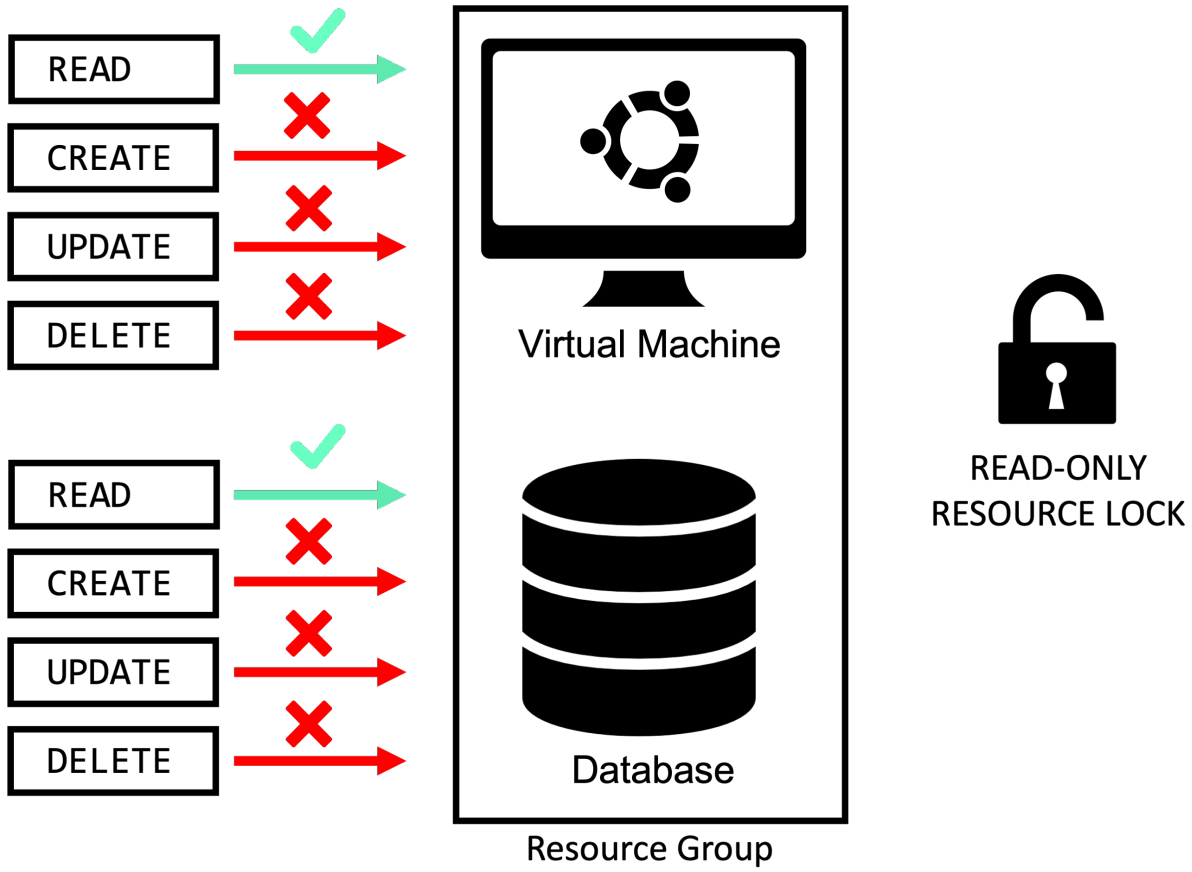
- Log in to the Azure CLI
- Get an overview of locks
- Create a resource group lock
- Create a resource lock

Overview

Resource locks are designed to prevent accidental deletion or edits. They are applied using role-based access control (RBAC). There are two types of locks as shown below: Delete and Read-Only.





As the diagrams show, Delete locks prevent removal of a resource. Read-Only locks prevent any edits to the resource, including creation, update, and deletion. Scopes can be applied on the subscription, resource group, or resource level. We will create these locks in this scenario. Note that only the Azure owner or a User Access Administrator can create locks.

First, we need a resource group. O'Reilly provides a resource group already created with a name saved to the variable `$resource`, which was created with this command below:

```
az group create --name $resource --location eastus
```

Then let's create a virtual machine:

```
az vm create --name 'MyVM' \
    --image UbuntuLTS \
    --location eastus \
    --resource-group $resource \
    --admin-username azureuser \
    --public-ip-sku Basic
```

```
$ az group create --name $resource --location eastus
{
  "id": "/subscriptions/8b5a4c25-6973-42af-99a5-926586bdc85d/resourceGr
  "location": "eastus",
  "managedBy": null,
  "name": "user-fkjtzsfchdzd",
  "properties": {
```

```
}
$ az vm create --name 'MyVM' \
>     --image UbuntuLTS \
>     --location eastus \
>     --resource-group $resource \
>     --admin-username azureuser \
>     --public-ip-sku Basic
Ignite (November) 2023 onwards "az vm/vms
 more about the default change and Truste
```

Resource Group Locks

Creating a lock is pretty straightforward. To apply a resource group lock, use the `az group lock create`. The `lock-type` can be either `CanNotDelete` or `ReadOnly` to apply a delete lock or read-only lock respectively.

Let's create a `ReadOnly` lock on the entire resource group. Note that we are applying the `resource-group` argument to put this lock not necessarily *in* the resource group but rather *on* the resource group:

```
az group lock create \
    --lock-type ReadOnly \
```

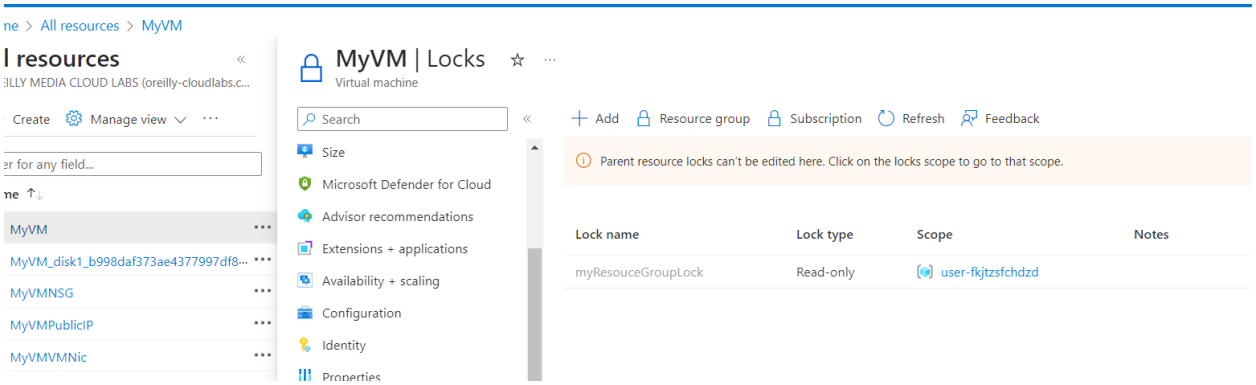
```
    --name myResouceGroupLock \
    --resource-group $resource
```

You can view your locks on a resource group by using the list command:

```
az group lock list --resource-group $resource --output table
```

```
$ az group lock create \
>     --lock-type ReadOnly \
>     --name myResouceGroupLock \
>     --resource-group $resource
{
  "id": "/subscriptions/8b5a4c25-6973-42af-99a5-926586b
.Authorization/locks/myResouceGroupLock",
  "level": "ReadOnly",
  "name": "myResouceGroupLock",
  "notes": null,
  "owners": null
```

```
$ az group lock list --resource-group $resource --output table
Level      Name                 ResourceGroup
--------   ------------------   ----------------
ReadOnly   myResouceGroupLock   user-fkjtzsfchdzd
$
```



To remove a resource group lock, call `az group lock delete` and provide the lock name and resource group:

```
az group lock delete \
    --name myResouceGroupLock \
    --resource-group $resource
```

Remember we can create and remove locks since we are the Azure account owner. Other users would not have this ability by default.

```
ReadOnly  myResouceGroupLock  user-fkjt
$ az group lock delete \
>     --name myResouceGroupLock \
>     --resource-group $resource
$
```

Resource Locks

To create a lock on an individual resource, use the `az lock create` command. Let's create a `CanNotDelete` lock on our virtual machine:

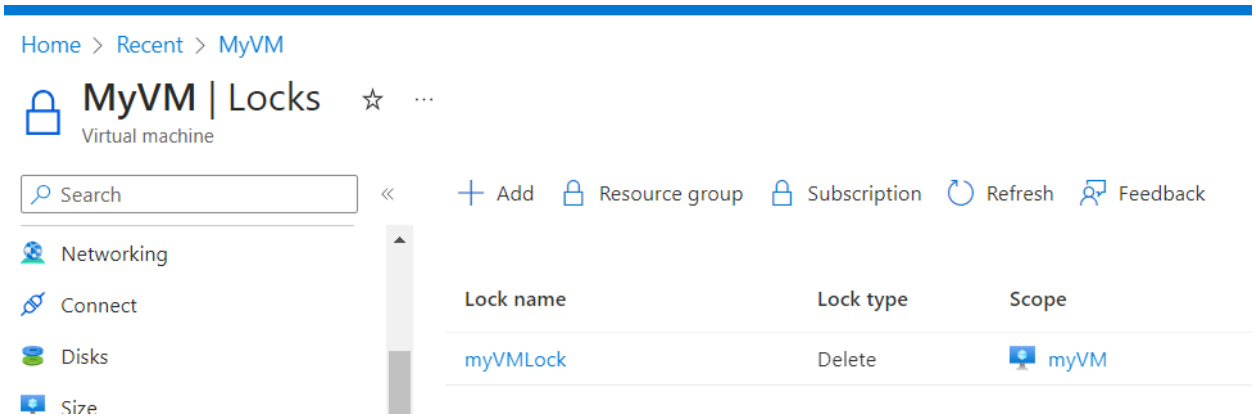
```
az lock create \
    --name myVMLock \
    --resource-group $resource \
    --lock-type CanNotDelete \
    --resource-type Microsoft.Compute/virtualMachines \
    --resource myVM
```

```
>     --resource-group $resource
$ az lock create \
>     --name myVMLock \
>     --resource-group $resource \
>     --lock-type CanNotDelete \
>     --resource-type Microsoft.Compute/virtualMachines \
>     --resource myVM
{
  "id": "/subscriptions/8b5a4c25-6973-42af-99a5-926586bdc85d
.Compute/virtualMachines/myVM/providers/Microsoft.Authorizat
  "level": "CanNotDelete",
  "name": "myVMLock"
```

To view a list of current locks, use the `az lock list` command:

```
az lock list --output table
```

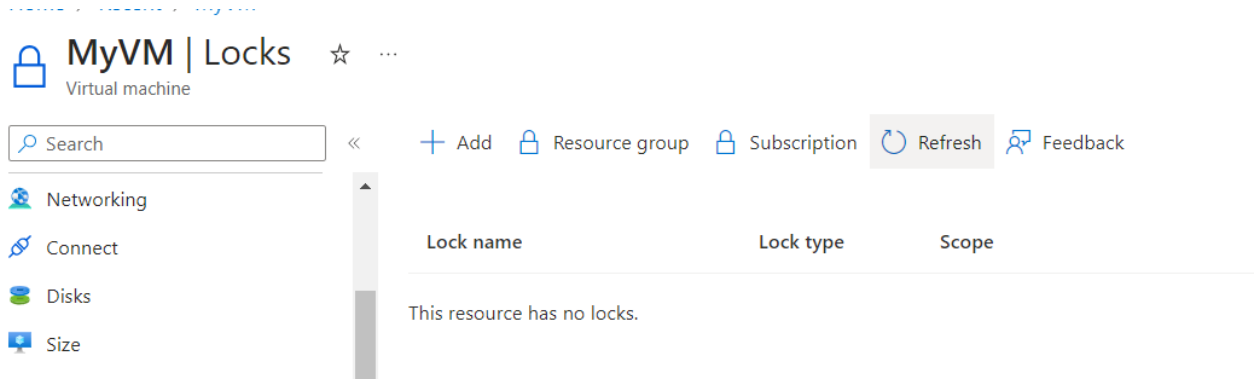
```
}
$ az lock list --output table
Level          Name       ResourceGroup
-------------  --------   -----------------
CanNotDelete   myVMLock   user-fkjtzsfchdzd
$
```



And if you want to delete a lock, use the `az lock delete` and specify the `name` of the lock, the `resource-name`, the `resource-group` and the `resource-type`:

```
az lock delete \
    --name myVMLock \
    --resource-name myVM \
    --resource-group $resource \
    --resource-type Microsoft.Compute/virtualMachines
```

```
CanNotDelete   myVMLock   user-fkjtzsfchdzd
$ az lock delete \
>     --name myVMLock \
>     --resource-name myVM \
>     --resource-group $resource \
>     --resource-type Microsoft.Compute/virtualMachines
$
```



Subscription Locks

You can create a lock on an entire subscription by running `az lock create` and just providing the lock name and type. We will not execute this command but will put it here for reference:

```
az lock create \
    --name mySubscriptionLock \
    --lock-type ReadOnly
```

—----------

Resource Tags

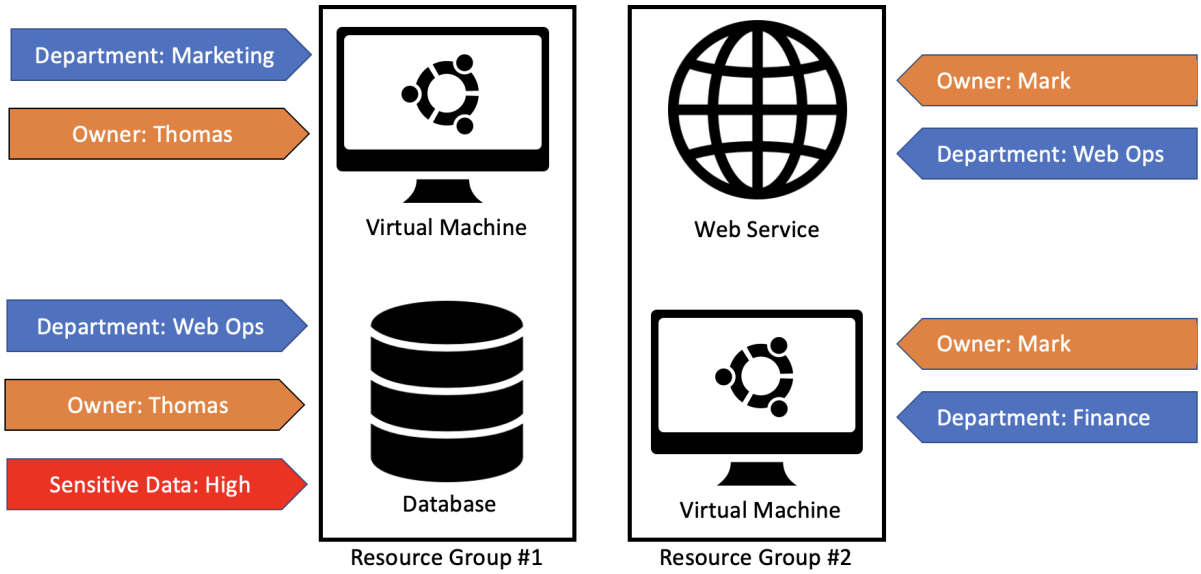
Log in to the Azure CLI
Create a resource tag
List resource tags
Update resource tags

Overview and Setup
Resource groups are effective and a simple way to force the organization of resources created in Azure. By bundling resources on a common initiative, application, or function, we create some form of organization. However, there may be moments where you want to organize resources beyond just a resource group, and arbitrarily assign tags that can be queried easily and can compose reports.

Resource tags are name-value pairs we can assign to our resources in order to organize them. By organizing beyond just the resource group, we can easily query resources and aggregate different reports on governance, operations, accounting, ownership, etc. We can apply resource tags to resources or entire resource groups. Just note that applying a tag to a resource group

does not put that tag on the individual resources inside it. You can also tag a subscription and the same rule applies, where its children resources and resource groups will not get that tag.

Here is an example diagram showing resources tagged across two security groups, tying different departments, owners, and data sensitivity labels to the resources.



Let's create a resource and a resource group, and practice tagging. To get started, let's first create a resource group. Let's also save its ID to a variable.

```
az group create --name $resource --location eastus
resourceGroupId=$(az group list --query "[?name=='$resource'].id" --output tsv)
```

Then let's create a virtual machine, and also save its ID to a variable.

```
az vm create --name 'MyVM' \
   --image UbuntuLTS \
   --location eastus \
   --resource-group $resource \
   --admin-username azureuser \
   --public-ip-sku Basic

resourceId=$(az vm list --query "[?name=='MyVM'].id" --output tsv)
```

```
EWBTLVEYGD56VEB0
$ az group create --name $resource --location eastus
{
  "id": "/subscriptions/6fff08fc-b4c5-4548-a434-33641c45686
  "location": "eastus",
  "managedBy": null,
  "name": "user-ayhixgkecxau",
```

```
$ resourceGroupId=$(az group list --query "[?name=='$resource'].id" --output tsv)
$ az vm create --name 'MyVM' \
>     --image UbuntuLTS \
>     --location eastus \
>     --resource-group $resource \
>     --admin-username azureuser \
>     --public-ip-sku Basic
Ignite (November) 2023 onwards "az vm/vmss create" command will deploy Gen2-Trusted
```

Tag a Resource

Let's apply two tags to the virtual machine resource, assigning a department=marketing key/value as well as a status=working key/value:

```
az tag create \
   --resource-id $resourceId \
   --tags department=marketing status=working
```

To view all the tags you have created so far, use the command az tag list.

```
$ az tag create \
>     --resource-id $resourceId \
>     --tags department=marketing status=working
{
  "id": "/subscriptions/6fff08fc-b4c5-4548-a434-33641c456864/resourceGroups/
.Compute/virtualMachines/MyVM/providers/Microsoft.Resources/tags/default",
  "name": "default",
  "properties": {
    "tags": {
      "department": "marketing",
      "status": "working"
    }
  },
```

az tag list

```
$ az tag list
[
  {
    "count": {
      "type": "Total",
      "value": 1
    },
    "id": "/subscriptions/6fff08fc-b4c5-4548-a434-336
    "tagName": "department",
    "values": [
      {
        "count": {
          "type": "Total",
```

You can also limit your tags to only certain resources. To look only at tags applied to the VM we just created, pass its ID to the --resource-id argument:

az tag list --resource-id $resourceId

```
$ az tag list --resource-id $resourceId
{
  "id": "/subscriptions/6fff08fc-b4c5-4548-a434-33
.Compute/virtualMachines/MyVM/providers/Microsoft.
  "name": "default",
  "properties": {
    "tags": {
      "department": "marketing",
      "status": "working"
```

When you search resources, you can use the tags in your JMESPath queries! If I want to find all VMs containing a department tag with the value marketing, we can put that condition in like this:

az vm list \
  --query "[?tags.department == 'marketing']" -o table

```
$ az vm list \
>     --query "[?tags.department == 'marketing']" -o table
Name    Location    ProvisioningState    VmId                                    TimeCreated
ResourceGroup
------  ----------  -------------------  --------------------------------------  ------------------------
-----------------
MyVM    eastus      Succeeded            aee3b800-c8ee-4be7-b9b6-72bd5647ef61  2023-06-21T06:54:41.8407
USER-AYHIXGKECXAU
```

Update a Tag
To apply a tag to a resource group, apply the resource group's ID to the --resource-id argument. Let's apply a tag needs_audit with a value of True to our resource group:

```
az tag create \
   --resource-id $resourceGroupId \
   --tags needs_audit=True
```

If we want to update this needs_audit tag to False, we can use the az tag update command. Note that the --operation argument accepts values of Merge, Replace, or Delete. Merge allows us to selectively update or insert new tags. Replace replaces all tags with a new set of tags we provide. Delete will allow us to remove tags with specified names.

```
USER-AYHIXGKECXAU
$ az tag create \
>     --resource-id $resourceGroupId \
>     --tags needs_audit=True
{
  "id": "/subscriptions/6fff08fc-b4c5-4548-a434-33641c45686
.Resources/tags/default",
  "name": "default",
  "properties": {
    "tags": {
      "needs_audit": "True"
    }
  },
  "resourceGroup": "user-ayhixgkecxau",
```

In this case, we just use Merge, which will update the needs_audit tag to False when we pass that to the tags argument.

```
az tag update \
   --resource-id $resourceGroupId \
   --operation merge \
   --tags needs_audit=False
```

```
}
$ az tag update \
>     --resource-id $resourceGroupId \
>     --operation merge \
>     --tags needs_audit=False
{
  "id": "/subscriptions/6fff08fc-b4c5-4548-a434-33641c
.Resources/tags/default",
  "name": "default",
  "properties": {
    "tags": {
      "needs_audit": "False"
    }
  },
  "resourceGroup": "user-ayhixgkecxau",
  "type": "Microsoft.Resources/tags"
}
```

MyVM

Virtual machine

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

Settings

- Networking
- Connect
- Disks
- Size
- Microsoft Defender for Cloud
- Advisor recommendations

Connect | Start | Restart | Stop | Capture

Essentials

Resource group (move)
user-ayhixgkecxau

Status
Running

Location
East US

Subscription (move)
cloudlabs43

Subscription ID
6fff08fc-b4c5-4548-a434-33641c456864

Tags (edit)
department : marketing | status : working

Resource groups

O'REILLY MEDIA CLOUD LABS (oreilly-cloudlabs.c...

Create | Manage view

Filter for any field...

Name

user-ayhixgkecxau

user-ayhixgkecxau | Tags

Resource group

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Resource visualizer
- Events

Settings

- Deployments
- Security
- Policies

Delete all

Tags are name/value pairs that enable yo
groups. Tag names are case insensitive, b

Do not enter names or values that could
globally.

Name

needs_audit

user-ayhixgkecxau (Resource gro

needs_audit : False

No changes

If we want to delete the tags on a resource or resource group altogether, use the az tag delete command.

```
$ az tag delete --resource-id $resourceGroupId --yes
$
```

Subscription Tags
Finally, we can create a tag for the entire subscription. We just need to provide a name on top of any tags we provide. Since we do not have permissions to create tags at the subscription level in this environment, here is the the command for reference:

az tag create --name MySubscriptionTag --tags sunset=True

ARM Templates

Log in to the Azure CLI
Understand Azure ARM templates
Import an ARM template
Export an ARM template

Overview
When developing a project where application code and database schemas are being iterated, it can be helpful to track the definitions for your Azure resources too. Azure Resource Management (ARM) templates provide a means to import and export one or more resource definitions in the JSON format. By declaring your resources as flat files, you can transport their definitions easily as well as put them under version control. We will learn how to import and export ARM templates in this scenario.

First, we need a resource group.
az group create --name $resource --location eastus
Then let's create a virtual machine. Let's also save its ID to a variable:

```
$ az group create --name $resource --location eastus
{
  "id": "/subscriptions/7075f968-4ed0-4eda-a971-4afd22b73b63/reso
  "location": "eastus",
  "managedBy": null,
  "name": "user-zsakmiwlprfy",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
```

az vm create --name MyVM \
  --image UbuntuLTS \

```
  --location eastus \
  --resource-group $resource \
  --admin-username azureuser \
  --public-ip-sku Basic
```

resourceId=$(az vm list --query "[?name=='MyVM'].id" --output tsv)

```
$ az vm create --name MyVM \
>     --image UbuntuLTS \
>     --location eastus \
>     --resource-group $resource \
>     --admin-username azureuser \
>     --public-ip-sku Basic
Ignite (November) 2023 onwards "az vm/vmss cr
 more about the default change and Trusted La
Consider using the "Ubuntu2204" alias. On Apr
```

Importing a Template

Open up the my_template.json file in the editor. It is an ARM template to create a storage account. Note that it already declares the SKU, its location, and other properties. There is also a parameter for the storageName on line 15, declared as "name": "[parameters('storageName')]",. When we see this parameters(...) placeholder, that means we need to provide that parameter when we build off the template.

The storageName has to be unique, so we will generate a randomId to concatenate to our storage name of mystorageXXXXXXXXXX, where XXXXXXXXXX is the 10-character generated ID. When we call the az deployment group create command, we provide the resource group and template file location. We will also pass the need parameters not specified in the template, which again in this case is the storageName:

randomId=$(cat /dev/urandom | env LC_ALL=C tr -dc 'a-z0-9' | fold -w 10 | head -n 1)

```
az deployment group create \
  --name 'MyImportedStorage' \
  --resource-group $resource \
  --template-file my_template.json \
  --parameters storageName=mystorage${randomId}
```

After that finishes, let's query for the resource to make sure it was created. We should see it in the output of this command below:

az resource list --query "[?name=='mystorage${randomId}']" --output table

```
$
$ resourceId=$(az vm list --query "[?name=='MyVM'].id" --output tsv)
$ randomId=$(cat /dev/urandom | env LC_ALL=C tr -dc 'a-z0-9' | fold -w 10 | head -n 1)
$
$ az deployment group create \
>   --name 'MyImportedStorage' \
>   --resource-group $resource \
>   --template-file my_template.json \
>   --parameters storageName=mystorage${randomId}
{
  "id": "/subscriptions/7075f968-4ed0-4eda-a971-4afd22b73b63/resourceGroups/user-zsakmiwlprfy/
.Resources/deployments/MyImportedStorage",
```

```
$ az resource list --query "[?name=='mystorage${randomId}']" --output table
Name                 Location    Kind       CreatedTime                       ChangedTime
rovisioningState    ResourceGroup
-------------------  ----------  ---------  --------------------------------  ----------------
------------------  ----------------
mystorage570z0iswre  eastus      StorageV2  2023-06-21T07:21:23.978811+00:00  2023-06-21T07:21
ucceeded            user-zsakmiwlprfy
```

Exporting a Template
We can also export an ARM template for a given resource. Below we export the VM definition for the virtual machine we created and put it in a myVm.json file:

```
az group export \
   --resource-group $resource \
   --resource-ids $resourceId  > myVm.json
```

```
az group export \
   --resource-group $resource  > myVm.json
```

Cleanup
We won't run this command; instead, we'll show the template. You can export the resources for an entire deployment or subscription using az deployment sub export. This will export the resources in the entire Azure account in JSON format.

```
az deployment sub export --name
                    [--subscription]
```

This covers the essentials of importing and exporting ARM templates. If you find yourself working on projects that go through iterative development and need to implement version control and track definitions, be sure to spend time on resource definitions and how they are structured as ARM templates.

Azure Monitor

Log in to the Azure CLI
Explore the Azure Monitor

View activity logs
Perform autoscaling on VMs

Step 2 - Overview
The Azure Monitor allows you to collect performance data on your Azure environments. You can get insight into the availability and performance of your resources and determine if you need to apply remedies, such as changing autoscaling policies. In this lab, we will create a single resource and browse the monitoring tools. We will also also explore scaling actions.

```
az group create --name $resource --location eastus
```

Let's create a virtual machine:

```
az vm create --name 'MyVM' \
   --image UbuntuLTS \
   --location eastus \
   --resource-group $resource \
   --admin-username azureuser \
   --public-ip-sku Basic
```

To view the five most recent events that have occurred on Microsoft Azure, run the following command:

```
az monitor activity-log list --query "[5]"
```

If you want to limit to activity in only our resource group, specify the --resource-group argument:

```
az monitor activity-log list --resource-group $resource
```

You can also specify a type of resource as well as a range of time for activities. If we want to list only virtual machine activity, we would provide the --namespace with an argument of Microsoft.Compute:

```
az monitor activity-log list \
   --namespace Microsoft.Compute
```

You can also list alerts that have occurred. You can configure different conditions and be notified when these conditions are breached:

```
az monitor activity-log alert list --resource-group $resource
```

Step 4 - Autoscaling
Autoscaling allows you to match resource capacity against workloads at a given time. For example, you can increase or decrease the number of VMs based on thresholds of traffic conditions. It can only scale horizontally in this manner, meaning it copes with workload changes by adding or removing VMs.

Resources that support autoscaling include:

VM scale sets
Azure App Service
Azure Logic Apps
Additional services listed in the Microsoft Docs

you could change the number of virtual machines at any time using this command template, where 3 specifies the number of instances and $resourceId is the ID of your scale set:

```
az monitor autoscale create \
   --name myVMAutoScaler \
   --resource-group $resource \
   --count 3 \
   --resource $resourceId
```

Note too that we can specify a --max-count and --min-count to dynamically create a range based on how much traffic is being received. It may be helpful to set --email-administrator and --email-coadministrators to true so notifications about scaling changes are sent out.

You can then view autoscale settings for items in a given resource group with this command. Note this will be empty since we did not apply any autoscaling:

```
az monitor autoscale list --resource-group $resource
```

We can also view an autoscale setting by name (e.g., if we did create that scale set) as shown:

```
az monitor autoscale show --name myVMAutoScaler --resource-group $resource
```