In this lab, you will learn how to list available policy definitions in your Azure subscription using the Azure CLI.
Azure Policy allows you to make sure your Azure resources comply to your organization standards, or are deployed using best practices.

- Log into the Azure CLI
- Understand Azure Policy
- List built-in Azure Policy definitions using Azure CLI
- Check built-in Azure policies in the Azure Portal

Use Azure Policy to enforce your company standards and best practices to your Azure subscriptions.

Let's understand what Azure Policy is in the next step.

Understand Azure Policy

Azure Policy allows you to make sure your Azure resources comply to your organization standards, or are deployed using best practices.

According to Microsoft, "Azure Policy helps to enforce organizational standards and to assess compliance at-scale."

Microsoft Azure offers many built-in policy definitions that you can use to enforce best practices, and organizational or government standards, to your cloud-based applications. Here are a few examples:

- Only allow resources to be deployed to certain Azure regions/locations (for example, only allow resources to be provisioned into the `East US` region)
- Only allow certain resource types to be deployed (for example, only allow `Azure Virtual Machines`, `App Services`, and `Storage Accounts` to be provisioned)

If none of the built-in policies satisfy your requirements, you can always create your own custom policy definitions.

After you choose the desired definition (either built-in or custom), you can *assign* it to a scope such as an Azure subscription or resource group, and all the resources within those scopes will be monitored for compliance.

In this lab, we will use Azure CLI to list the available built-in policy definitions.

List Built-in Policy Definitions Using Azure CLI

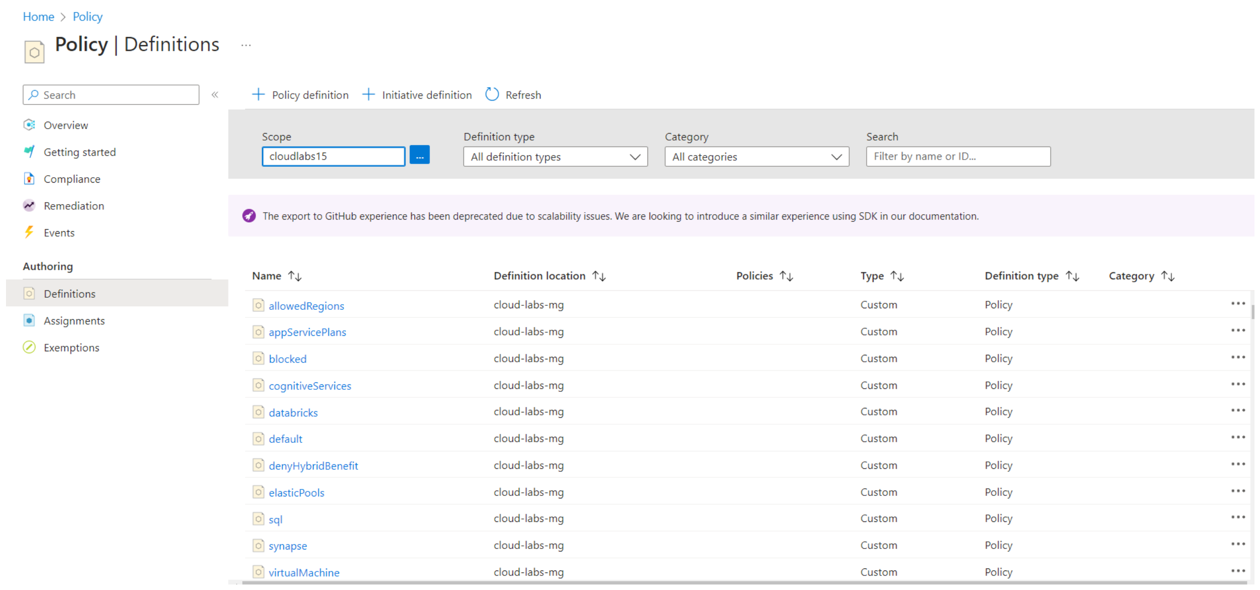
Use the following CLI command to list all the available built-in policies for your subscription:

```
az policy definition list --query "[].{PolicyType: policyType,
Description: description}"
```

Confirm that you can see the list of policy names and descriptions in the command output.

Note that these are available policy definitions and they are not necessarily assigned or in effect. In the next lab, we will talk about assigning policy definitions to a scope.

- Under `Authoring`, click on `Definitions`.
- Observe the list of available policies.


Home > Policy

Policy | Definitions

Scope: cloudlabs15

Definition type: All definition types

Category: All categories

Search: Filter by name or ID...

The export to GitHub experience has been deprecated due to scalability issues. We are looking to introduce a similar experience using SDK in our documentation.

Name	Definition location	Policies	Type	Definition type	Category
allowedRegions	cloud-labs-mg		Custom	Policy	
appServicePlans	cloud-labs-mg		Custom	Policy	
blocked	cloud-labs-mg		Custom	Policy	
cognitiveServices	cloud-labs-mg		Custom	Policy	
databricks	cloud-labs-mg		Custom	Policy	
default	cloud-labs-mg		Custom	Policy	
denyHybridBenefit	cloud-labs-mg		Custom	Policy	
elasticPools	cloud-labs-mg		Custom	Policy	
sql	cloud-labs-mg		Custom	Policy	
synapse	cloud-labs-mg		Custom	Policy	
virtualMachine	cloud-labs-mg		Custom	Policy	

In this lab, you will learn how to assign an Azure Policy definition to a scope using Azure CLI.

Azure Policy allows you to make sure your Azure resources comply to your organization standards, or are deployed using best practices.

According to Microsoft, "Azure Policy helps to enforce organizational standards and to assess compliance at-scale."

- Log into the Azure CLI
- Understand Azure policy assignments
- Assigning built-in Azure policy definitions to a subscription using Azure CLI
- Confirm the creation of the policy assignment
- Delete a policy assignment

Use Azure Policy to enforce your company standards and best practices to your Azure subscriptions.

Understand Azure Policy Assignments

Azure Policy allows you to make sure your Azure resources comply to your organization standards, or are deployed using best practices.

According to Microsoft, "Azure Policy helps to enforce organizational standards and to assess compliance at-scale."

Microsoft Azure offers many built-in policy definitions that you can use to enforce best practices, and organizational or government standards, to your cloud-based applications. If none of the built-in policies satisfy your requirements, you can always create your own custom policy definitions.

After you choose the desired definition (either built-in or custom), you need to *assign* the policy to a scope such as an Azure subscription or resource group, and all the resources within those scopes will be monitored for compliance. For example, you can assign the `Allowed Resource Types` policy to a subscription, and from that moment only allowed resources can be provisioned in the said subscription.

In this lab, you will learn the Azure CLI command to assign built-in policy definitions to your desired scope, such as your temporary subscription.

Assign a Built-in Policy Definition to a Scope Using Azure CLI

First we need to choose a policy definition to assign. This can be either a built-in or custom definition. We will use a built-in policy named `Storage account keys should not be expired`. When assigned, the storage accounts deployed to the scope should have a never-expiring key. Read more about this policy here.

Note: Imagine this policy is assigned to a sample subscription such as `SUB01`. Both existing and future storage accounts in `SUB01` will be monitored for compliance. This means you will get a compliance report in the policy overview page and new resources should be compliant; otherwise, their deployment will fail!

First, let's find this policy definition ID/name and store it into a variable:

```
POLICY_ID=$(az policy definition list --query "[?displayName ==
'Storage account keys should not be expired'].name" -o tsv)
```

Use the following command to see the policy ID/name if needed:

```
echo $POLICY_ID
```

We also need the `scope`. It can be a management group, subscription, or even a resource group. Let's choose your temporary resource group. Store the resource group ID in a variable using this command:

```
RG_ID=$(az group show --name $resource --query id -o tsv)
```

Use the following command to see the policy ID/name if needed:

```
echo $RG_ID
```

Now you have all the information needed to create a policy assignment. **Our environment does not allow the assignment of policies due to security limitations**, but we will show the CLI command for your reference:

```
az policy assignment create --name 'Storage account keys should
not be expired' --enforcement-mode Default --policy $POLICY_ID
--scope $RG_ID
```

Important: A policy definition can be assigned to multiple scopes using multiple policy assignments!

Confirm the Azure Policy Assignment Creation

You can use the following command to list all the policy assignments for a scope:

```
az policy assignment list --scope $RG_ID
```

Delete the Azure Policy Assignment

Finally, use this command to delete a policy assignment. This command will *unassign* the policy from the scope:

```
az policy assignment delete --name "Storage account keys should
not be expired"
```

Important: Deleting a policy assignment will *not* delete the policy definition, so it can be assigned later if needed.

In this lab, you will learn how to create a custom Azure Policy definition using Azure CLI. Check this document for details on a policy file structure.

Understand the Need for Custom Azure Policies

Azure Policy allows you to make sure your Azure resources comply to your organization standards, or are deployed using best practices.

Microsoft Azure offers many built-in policy definitions that you can use to enforce best practices, and organizational or government standards, to your cloud-based applications. If none of the built-in policies satisfy your requirements, you can always create your own custom policy definitions.

To create a new custom policy definition, you need to create a *policy definition JSON file*. Then you pass this file to the `az policy definition create` command, as we will see in the next step. The content of this JSON file can also be passed inline as a command string parameter.

Here are the fields for the policy structure file:

- `display name`: The name of the new policy.
- `description`: Description for the policy explaining what it does.
- `mode`: Can be `all` (evaluates resource groups, subscriptions, and all resource types), or `indexed` (only evaluates resource types that support tags and location). You will choose `all` for the most cases.
- `metadata`: Stores information about the policy definition.
- `parameters`: List of the parameters a policy accepts. For example, to allow deployment to the `East US` region, simply pass `East US` as a parameter value to the `allowed locations` policy. For details, see this document.
- `policy rule`: Consists of `If` and `Then` blocks to determine the conditions for the policy to check/enforce. See this document for more details.
  - `logical evaluation`
  - `effect`

In this lab, you will learn the Azure CLI command to create a custom policy definition. Similar to built-in policies, custom policy definitions can be assigned to different scopes, such as your temporary subscription.

Create a Custom Policy Definition Using Azure CLI

The following simple policy definition ensures that the created Azure storage accounts are read-only. Any existing storage account will show as `incompliant` in the Policy Overview page.

```
{
    "if":
    {
        "field": "type",
        "equals": "Microsoft.Storage/storageAccounts/write"
    },
    "then":
    {
        "effect": "deny"
    }
}
```

Note: You can save this in a JSON file and pass it to the CLI command, or simply pass the JSON value as the command value. Note that we are skipping the *double quotes* `"` character using a *backslash* such as `\"`:

```
az policy definition create --name readOnlyStorageAccount
--rules "{ \"if\": { \"field\": \"type\", \"equals\":
\"Microsoft.Storage/ storageAccounts/write\" }, \"then\": {
\"effect\": \"deny\" } }"
```

List the Policy Definitions Using Azure CLI

You can use the following command to list all the custom policy definitions:

```
az policy definition list --query
"[?policyType=='Custom'].{DisplayName: displayName, Description:
description}"
```