


[< Previous](#)Unit 2 of 8 [Next >](#)✓ 100 XP 

# Define IT compliance with Azure Policy

8 minutes

Planning out a consistent cloud infrastructure starts with setting up policy. Your policies will enforce your rules for created resources, so your infrastructure stays compliant with your corporate standards, cost requirements, and any service-level agreements (SLAs) you have with your customers.



**Azure Policy** is an Azure service you use to create, assign and, manage policies. These policies enforce different rules and effects over your resources so that those resources stay compliant with your corporate standards and service level agreements. Azure Policy meets this need by evaluating your resources for noncompliance with assigned policies. For example, you might have a policy that allows virtual machines of only a certain size in your environment. After this policy is implemented, new and existing resources are evaluated for compliance. With the right type of policy, existing resources can be brought into compliance.

Imagine we allow anyone in our organization to create virtual machines (VMs). We want to

control costs, so the administrator of our Azure tenant defines a policy that prohibits the creation of any VM with more than 4 CPUs. Once the policy is implemented, Azure Policy will stop anyone from creating a new VM outside the list of allowed stock keeping units (SKUs). Also, if you try to *update* an existing VM, it will be checked against policy. Finally, Azure Policy will audit all the existing VMs in our organization to ensure our policy is enforced. It can audit non-compliant resources, alter the resource properties, or stop the resource from being created. You can even integrate Azure Policy with Azure DevOps, by applying any continuous integration and delivery pipeline policies that affect the pre-deployment and post-deployment of your applications.

### How are Azure Policy and RBAC different?

At first glance, it might seem like Azure Policy is a way to restrict access to specific resource types similar to role-based access control (RBAC). However, they solve different problems. RBAC focuses on *user actions at different scopes*. You might be added to the contributor role for a resource group, allowing you to make changes to anything in that resource group. Azure Policy focuses on *resource properties during deployment* and for already-existing resources. Azure Policy controls properties such as the types or locations of resources. Unlike RBAC, Azure Policy is a **default-allow-and-explicit-deny system**.

## Creating a policy

The process of creating and implementing an Azure Policy begins with creating a *policy definition*. Every policy definition has conditions under which it is enforced. And, it has an accompanying effect that takes place if the conditions are met. To apply a policy, you will:

1. Create a policy definition
2. Assign a definition to a scope of resources
3. View policy evaluation results

### What is a policy definition?

A *policy definition* expresses what to evaluate and what action to take. For example, you could


ensure all public websites are secured with HTTPS, prevent a particular storage type from being created, or force a specific version of SQL Server to be used.

Here are some of the most common policy definitions you can apply.

| Policy definition            | Description   |
|------------------------------|---|
| Allowed Storage Account SKUs | This policy definition has a set of conditions/rules that determine whether a storage account that is being deployed is within a set of SKU sizes. Its effect is to deny all storage accounts that do not adhere to the set of defined SKU sizes. |
| Allowed Resource Type        | This policy definition has a set of conditions/rules to specify the resource types that your organization can deploy. Its effect is to deny all resources that are not part of this defined list.   |
| Allowed Locations            | This policy enables you to restrict the locations that your organization can specify when deploying resources. Its effect is used to enforce your geographic compliance requirements.   |
| Allowed Virtual Machine SKUs | This policy enables you to specify a set of VM SKUs that your organization can deploy.  |
| Not allowed resource types   | Prevents a list of resource types from being deployed.  |

The policy definition itself is represented as a JSON file - you can use one of the pre-defined definitions in the portal or create your own (either modifying an existing one or starting from scratch). There are [hundreds of samples available on GitHub](#).

Here is an example of a Compute policy that only allows specific virtual machine sizes:

|                            |  |
|----------------------------|--|
| JSON                       |  Copy |
| <pre>{<br/>  "if": {</pre> |  |

```
"allOf": [
  {
    "field": "type",
    "equals": "Microsoft.Compute/virtualMachines"
  },
  {
    "not": {
      "field": "Microsoft.Compute/virtualMachines/sku.name",
      "in": "[parameters('listOfAllowedSKUs')]"
    }
  }
],
"then": {
  "effect": "Deny"
}
```

Notice the `[parameters('listOfAllowedSKUs')]` value; this value is a *replacement token* that will be filled in when the policy definition is applied to a scope. When a parameter is defined, it's given a name and optionally given a value.

## Applying Azure policy

To apply a policy, we can use the Azure portal, or one of the command-line tools such as Azure PowerShell by adding the `Microsoft.PolicyInsights` extension.

PowerShell

 Copy

```
# Register the resource provider if it's not already registered
Register-AzResourceProvider -ProviderNamespace 'Microsoft.PolicyInsights'
```

Once we have registered the provider, we can create a policy assignment. For example, here's a policy definition that identifies virtual machines not using managed disks.

PowerShell

 Copy

```
# Get a reference to the resource group that will be the scope of the as-
signment
```

```
$rg = Get-AzResourceGroup -Name '<resourceGroupName>'

# Get a reference to the built-in policy definition that will be assigned
$definition = Get-AzPolicyDefinition | Where-Object { $_.Properties.DisplayName -eq 'Audit VMs that do not use managed disks' }

# Create the policy assignment with the built-in definition against your resource group
New-AzPolicyAssignment -Name 'audit-vm-manageddisks' -DisplayName 'Audit VMs without managed disks Assignment' -Scope $rg.ResourceId -PolicyDefinition $definition
```

The preceding commands use the following information:

| Parameter          | Description   |
|--------------------|---|
| <b>Name</b>        | The actual name of the assignment. For this example, <code>audit-vm-manageddisks</code> was used.   |
| <b>DisplayName</b> | Display name for the policy assignment. In this case, you're using <code>Audit VMs without managed disks Assignment</code> .  |
| <b>Definition</b>  | The policy definition, based on which you're using to create the assignment. In this case, it's the ID of policy definition <code>Audit VMs that do not use managed disks</code> .  |
| <b>Scope</b>       | A scope determines what resources or grouping of resources the policy assignment gets enforced on. It could range from a subscription to resource groups. Be sure to replace <code>&lt;scope&gt;</code> with the name of your resource group. |

## Identifying non-compliant resources

We can use the applied policy definition to identify resources that aren't compliant with the policy assignment through the Azure portal

The results match what you see in the Resource compliance tab of a policy assignment in the Azure portal:

Home > Policy - Compliance

Policy - Compliance

Search (Ctrl+/)

Assign policy Assign initiative Refresh

Scope: Contoso Type: All definition types Compliance state: All compliance states Search: Filter by name or id...

Overall resource compliance: 100% Non-compliant initiatives: 0 out of 1 Non-compliant policies: 0 out of 39 Non-compliant resources: 0 out of 4

| NAME                              | SCOPE                | COMPLIANCE STATE | COMPLIANCE | NON-COMPLIANT RESOURCES | NON-COMPLIANT POLICIES |
|-----------------------------------|----------------------|------------------|------------|-------------------------|------------------------|
| Audit VMs that do not use ma...   | Contoso/PolicyTarget | Compliant        | 100%       | 0                       | 0                      |
| [Preview]: Enable Monitoring i... | Contoso/PolicyTarget | Compliant        | 100%       | 0                       | 0                      |

Or we can again use the command-line tools to identify the resources in your resource group that are non-compliant to the policy assignment

PowerShell

Copy

```
Get-AzPolicyState -ResourceGroupName $rg.ResourceGroupName -PolicyAssignmentName 'audit-vm-manageddisks' -Filter 'IsCompliant eq false'
```

Here's an example of the output we might get:

Output

Copy

```
Timestamp           : 3/9/19 9:21:29 PM
ResourceId           : /subscriptions/{subscriptionId}/resource-groups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmId}
PolicyAssignmentId   : /subscriptions/{subscriptionId}/providers/microsoft.authorization/policyassignments/audit-vm-manageddisks
PolicyDefinitionId   : /providers/Microsoft.Authorization/policyDefinitions/06a78e20-9358-41c9-923c-fb736d382a4d
IsCompliant          : False
SubscriptionId       : {subscriptionId}
ResourceType         : /Microsoft.Compute/virtualMachines
ResourceTags         : tbd
PolicyAssignmentName  : audit-vm-manageddisks
PolicyAssignmentOwner : tbd
PolicyAssignmentScope : /subscriptions/{subscriptionId}
PolicyDefinitionName  : 06a78e20-9358-41c9-923c-fb736d382a4d
PolicyDefinitionAction : audit
PolicyDefinitionCategory : Compute
```

```
ManagementGroupIds      : {managementGroupId}
```

## Assign a definition to a scope of resources

Once you've defined one or more policy definitions, you'll need to assign them. A *policy assignment* is a policy definition that has been assigned to take place within a specific scope.

This scope could range from a full subscription down to a resource group. Policy assignments are inherited by all child resources. This inheritance means that if a policy is applied to a resource group, it is applied to all the resources within that resource group. However, you can exclude a subscope from the policy assignment. For example, we could enforce a policy for an entire subscription and then exclude a few select resource groups.

You can assign any of these policies through the Azure portal, PowerShell, or Azure CLI. When you assign a policy definition, you will need to supply any parameters that are defined.

[Home](#) > [Policy - Definitions](#) > [Allowed virtual machine SKUs](#) > Allowed virtual machine SKUs

### Allowed virtual machine SKUs

Assign policy

Description

Stop all VMs except Standard\_A2 series.

PARAMETERS

\* Allowed SKUs ⓘ

3 selected

## Policy effects

Requests to create or update a resource through Azure Resource Manager are evaluated by



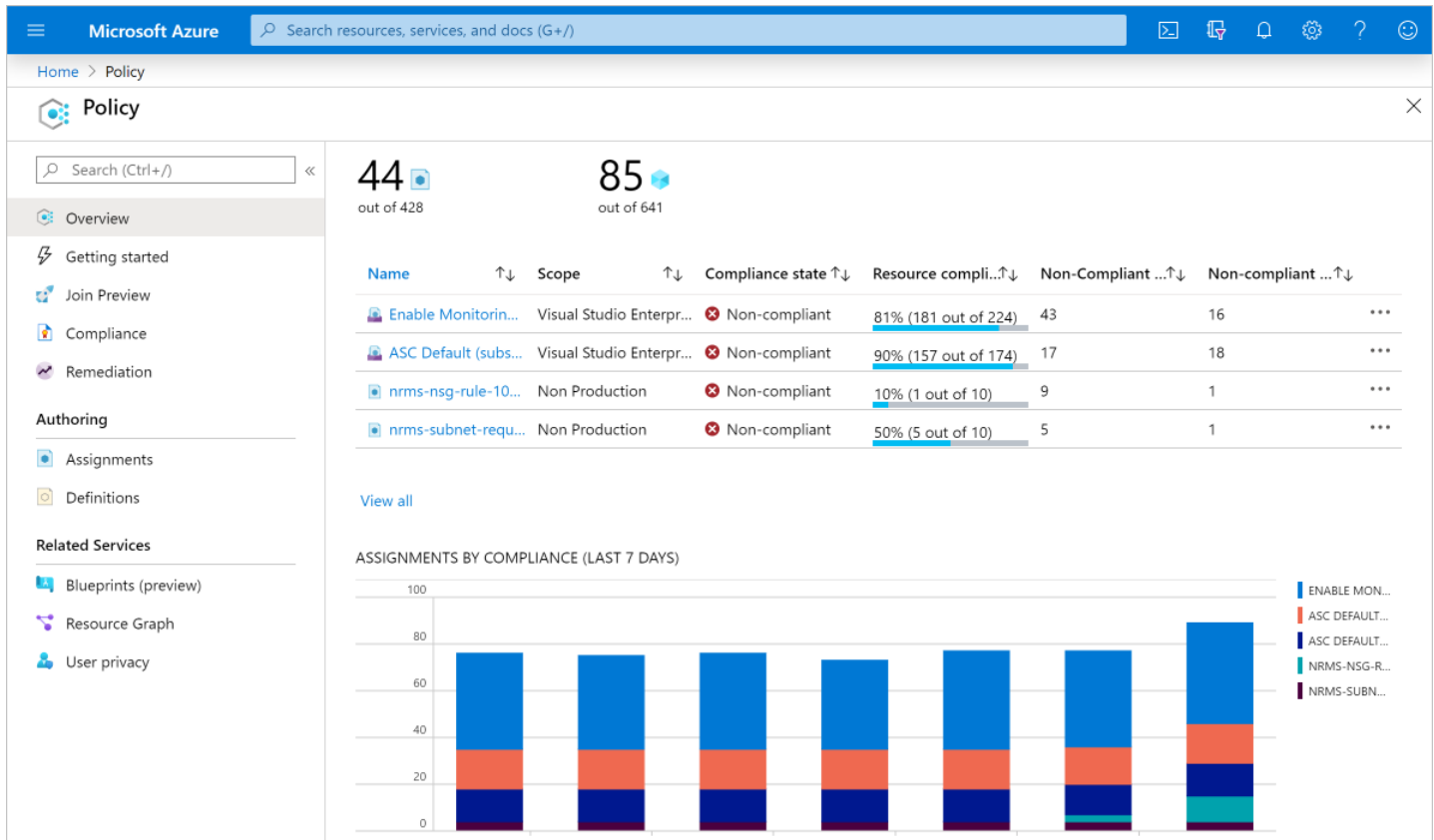
Azure Policy first. Policy creates a list of all assignments that apply to the resource and then evaluates the resource against each definition. Policy processes several of the effects before handing the request to the appropriate Resource Provider to avoid any unnecessary processing if the resource violates policy.

Each policy definition in Azure Policy has a single effect. That effect determines what happens when the associated policy rule is matched. When that happens, Azure Policy will take a specific action based on the assigned effect.

| Policy Effect              | What happens?   |
|----------------------------|---|
| Deny                       | The resource creation/update fails due to policy.   |
| Disabled                   | The policy rule is ignored (disabled). Often used for testing.  |
| Append                     | Adds additional parameters/fields to the requested resource during creation or update. A common example is adding tags on resources such as Cost Center or specifying allowed IPs for a storage resource. |
| Audit,<br>AuditIfNotExists | Creates a warning event in the activity log when evaluating a non-compliant resource, but it doesn't stop the request.  |
| DeployIfNotExists          | Executes a template deployment when a specific condition is met. For example, if SQL encryption is enabled on a database, then it can run a template after the DB is created to set it up a specific way. |

## View policy evaluation results

Azure Policy can allow a resource to be created even if it doesn't pass validation. In these cases, you can have it trigger an audit event that can be viewed in the Azure Policy portal, or through command-line tools. The easiest approach is in the portal as it provides a nice graphical overview that you can explore. You can find the Azure Policy section through the search field or *All Services*.



From this screen, you can spot resources that are not compliant and take action to correct them.

## Removing a policy definition

Finally, you can delete policy requirements through the portal, or through the PowerShell command `Remove-AzPolicyAssignment` as shown below.

PowerShell

Copy

```
Remove-AzPolicyAssignment -Name 'audit-vm-manageddisks' -Scope '/subscriptions/<subscriptionID>/resourceGroups/<resourceGroupName>'
```

## Next unit: Organize policy with initiatives

Continue >

