**1. What is an Azure Resource Group, and why is it important?**

When you create one resource multiple resources are getting created. How you manage them in a logical way is Resource group

A **Resource Group (RG)** is a logical container that holds Azure resources like VMs, databases, and storage accounts. It helps in managing and organizing resources for easier deployment, monitoring, access control, and cost management.

**✅ Importance:**

* **Lifecycle management**: Deleting the RG deletes all associated resources.
* **Access control**: Uses **RBAC** for permissions.
* **Cost tracking**: Can track costs per RG.
* **Policy enforcement**: Enables applying **Azure Policies** at the RG level.

**2. Can a resource exist in multiple resource groups?**

No, a resource can belong to only **one** Resource Group at a time. However, you can **move** resources between RGs or **link** resources across RGs via references.

**3. What happens if a Resource Group is deleted?**

If a Resource Group is deleted:

* **All resources inside it are also deleted** (except locked resources).
* Soft delete may apply to **Recovery Services Vault** and **Storage Accounts** if enabled.
* **Role assignments and policies are lost**.

**4. What are Azure Resource Locks, and how do they work?**

Azure **Resource Locks** prevent accidental deletion or modification of critical resources. There are two types:

* **ReadOnly**: Prevents changes but allows read access.
* **CanNotDelete**: Prevents deletion but allows modifications.

Locks override **RBAC permissions**, meaning even an **Owner** cannot delete a locked resource.

**5. How does resource group location affect resources inside it?**

The Resource Group's **location** determines:

* **Metadata storage** (e.g., deployment history).
* **Availability of Azure services** (if regional).
* Resources inside **can be in different regions** than the RG.

**Example:** A Resource Group in **UK South** can contain a VM in **West Europe**.

**6. Can you move resources between Resource Groups? What are the limitations?**

Yes, you can move resources between RGs **within the same subscription or across subscriptions**. However, there are limitations:

* Resources **must support move operations** (e.g., App Insights cannot be moved if it's linked to Log Analytics).
* **Locks and policies** are not transferred.
* Some resources may **require downtime** during the move.

**Command:**

Move-AzResource -DestinationResourceGroupName "NewRG" -ResourceId "/subscriptions/{sub-id}/resourceGroups/{OldRG}/providers/Microsoft.Compute/virtualMachines/{vm-name}"

**7. How do you enforce compliance in a Resource Group?**

Use **Azure Policies** and **Azure Blueprints** to enforce compliance.

**Example Policies:**

* Restrict VM SKU types
* Enforce **tagging** (e.g., environment = Production)
* Block public IP assignment

**8. What are the best practices for managing Azure Resource Groups?**

✅ **Best Practices:**

* Group resources **by application lifecycle** (Dev, Test, Prod).
* Apply **RBAC** at the Resource Group level.
* Use **naming conventions** (e.g., rg-prod-appname).
* Enable **Azure Policy** for security and compliance.
* Implement **tags** for cost tracking.
* Use **locks** for critical resources.
* Automate RG creation using **Terraform/Bicep**.

**9. How can you track costs for resources in a Resource Group?**

You can track costs using **Azure Cost Management** and **Budgets**:

1. **Azure Cost Analysis** → Shows costs by RG.
2. **Tags** → Assign business units or departments.
3. **Budgets** → Set spending alerts.

**Example CLI to check RG cost:**

az consumption usage list --query "[?resourceGroup=='myResourceGroup']"

**10. What is the difference between a Subscription and a Resource Group?**

| **Feature** | **Resource Group** | **Subscription** |
| --- | --- | --- |
| **Scope** | Logical grouping of resources | Billing unit for Azure services |
| **Billing Impact** | No separate billing | Subscription-level billing |
| **RBAC Control** | Applied at RG level | Applied at Subscription level |
| **Policies** | Can apply to RG | Can apply to entire Subscription |
| **Max Count** | No hard limit | Can have multiple |

**Azure Resource Locks**

Azure Resource Locks help prevent accidental deletion or modification of critical resources in an Azure subscription. They provide an extra layer of protection by enforcing restrictions on resources at the subscription, resource group, or individual resource level.

**Types of Resource Locks**

Azure provides two types of locks:

**Read-Only (CanNotDelete)**

* Prevents modification or deletion of a resource but allows read access.
* Equivalent to granting **Reader** role access.
* **Example:** You can read configurations but cannot update or delete the resource.

**Delete (ReadOnly)**

* Prevents deletion of a resource but allows modifications.
* **Example:** You can update the settings of a resource but cannot delete it.

**Casestudy1**- I want to create an azure web app, and I want to apply Azure Resource Locks, so no one can delete it.

# Variables

RESOURCE\_GROUP="MyResourceGroup"

APP\_NAME="my-web-app-$RANDOM"

PLAN\_NAME="MyAppServicePlan"

LOCATION="eastus"

# Create a Resource Group

az group create --name $RESOURCE\_GROUP --location $LOCATION

# Create an App Service Plan

az appservice plan create --name $PLAN\_NAME --resource-group $RESOURCE\_GROUP --sku B1 --is-linux

# Create the Web App

az webapp create --name $APP\_NAME --resource-group $RESOURCE\_GROUP --plan $PLAN\_NAME

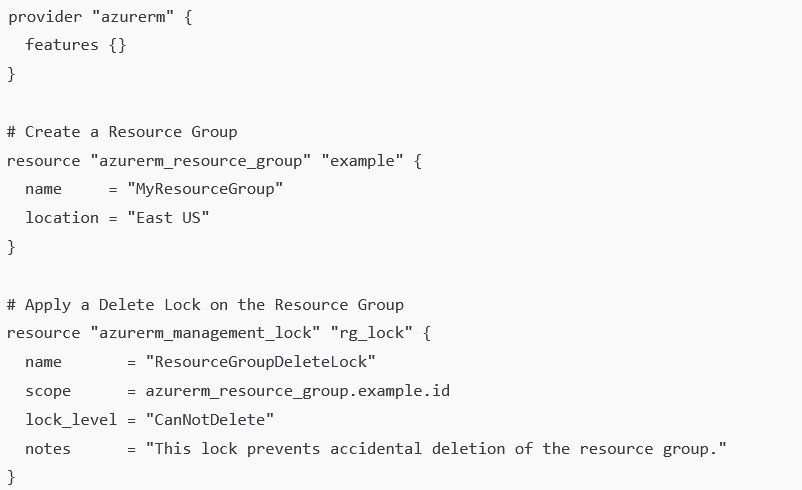
# Apply a Delete Lock to the Web App

az lock create --name "WebAppDeleteLock" --lock-type CanNotDelete --resource-group $RESOURCE\_GROUP --resource-name $APP\_NAME --resource-type "Microsoft.Web/sites"

Now, if you want to undo the operation of create then you would be going for delete-

Remove-AzResourceLock -LockName "WebAppDeleteLock" -ResourceGroupName $resourceGroup -ResourceName $appName -ResourceType "Microsoft.Web/sites"

Casestudy2- I want to create the resource group with accidental delete policy-



**What is Resource Lifecycle?**

The **Resource Lifecycle** refers to the different stages a cloud resource goes through from **creation** to **decommissioning**. In **Azure**, every resource (like VMs, storage accounts, databases, etc.) follows a lifecycle that includes provisioning, usage, updates, and eventual deletion.

**Phases of Resource Lifecycle**

1. **Provisioning (Creation)**
   * Resources are created based on configuration settings.
   * Example: Deploying a Virtual Machine, Storage Account, or Web App.
2. **Configuration & Initialization**
   * Applying necessary settings, security configurations, and access controls.
   * Example: Assigning IP addresses, setting up permissions, or configuring autoscaling.
3. **Operational (Active Use)**
   * Resource is actively used in production or development environments.
   * Example: Running workloads, storing data, or processing transactions.
4. **Monitoring & Optimization**
   * Performance and cost monitoring to ensure efficient resource usage.
   * Example: Using Azure Monitor and Application Insights for tracking performance.
5. **Modification & Scaling**
   * Adjusting resource configurations (scaling up/down, updating settings).
   * Example: Increasing VM size, adding more storage, or changing networking rules.
6. **Decommissioning (Retention & Deletion)**
   * Retiring unused resources to optimize cost and security.
   * Example: Deleting unused storage, removing old VM instances.

**1. Can a Resource Group Have an Alias Name?**

No, **Azure Resource Groups do not support alias names** natively. The resource group name must be **globally unique within a subscription** and cannot have an alias or alternate name.

**Workarounds:**

* **Use Azure Tags**: You can assign a friendly name as a tag to help identify the resource group.

az tag create --resource-id "/subscriptions/{subscription-id}/resourceGroups/MyResourceGroup" --tags AliasName=Prod-RG

**2. Can a Resource Group Have Multiple Locations?**

No, a **Resource Group itself is tied to a single region**, but **it can contain resources from multiple locations**.