The **Contributor** role in Azure is like giving someone the keys to manage everything in a room—except the ability to hand out more keys or change who has access. Here's a detailed breakdown of what it allows and what it restricts:

**✅ What a Contributor Can Do**

A user with the Contributor role can **create, delete, and manage** almost all Azure resources. This includes:

* **Virtual Machines**: Create, start, stop, delete, and configure VMs.
* **Storage Accounts**: Set up blob containers, file shares, and manage data access policies.
* **Databases**: Deploy and configure Azure SQL, Cosmos DB, etc.
* **Networking**: Create and manage VNets, NSGs, public IPs, load balancers.
* **Web Apps & Functions**: Deploy code, configure settings, scale resources.
* **Monitoring & Alerts**: Set up diagnostic settings, alerts, and metrics.
* **Resource Groups**: Create new groups and manage resources within them.
* **Tags & Locks**: Apply tags for organization and set resource locks to prevent accidental deletion.

Basically, they can **do everything needed to build and operate solutions** in Azure.

**🚫 What a Contributor Cannot Do**

Despite having broad access, there are key restrictions:

* ❌ **Cannot assign roles or change access permissions**  
  They can't use Azure RBAC to give others access or modify existing role assignments.
* ❌ **Cannot manage Azure Blueprints**  
  They can't assign or modify blueprint definitions.
* ❌ **Cannot share image galleries**  
  They can't publish or share VM images across subscriptions.
* ❌ **Cannot elevate access or delete authorization settings**  
  Actions like Microsoft.Authorization/\*/Delete and elevateAccess/Action are blocked.

**🔍 Behind the Scenes: Role Definition**

The Contributor role is defined with:

* Actions: ["\*"] → Full access to all management operations.
* NotActions: ["Microsoft.Authorization/\*", "Microsoft.Blueprint/\*", ...] → Explicitly blocks access control and blueprint-related actions.

This means the role is **powerful but scoped**, ideal for developers, engineers, or automation accounts that need to manage infrastructure but **shouldn’t control security or governance**.

Let’s compare the **Contributor**, **Owner**, and **Reader** roles in Azure so you can see how they stack up and where each fits best:

**🔍 Role Comparison Table**

| **Role** | **Permissions Scope** | **Can Assign Roles?** | **Can Manage Resources?** | **Can View Resources?** |
| --- | --- | --- | --- | --- |
| **Owner** | Full control over all resources | ✅ Yes | ✅ Yes | ✅ Yes |
| **Contributor** | Full management of resources (except access) | ❌ No | ✅ Yes | ✅ Yes |
| **Reader** | View-only access | ❌ No | ❌ No | ✅ Yes |

**🧠 Use Cases for Each Role**

* **Owner**:  
  Ideal for **subscription admins** or **team leads** who need to manage everything, including access control.
* **Contributor**:  
  Perfect for **developers**, **DevOps engineers**, or **automation accounts** that need to deploy and manage resources but **shouldn’t control access**.
* **Reader**:  
  Great for **auditors**, **finance teams**, or **stakeholders** who need visibility into resources without making changes.

**🛠️ Custom Roles?**

If you want to fine-tune access even further—like giving someone permission to restart VMs but not delete them—you can create **custom roles** using specific Actions and NotActions.