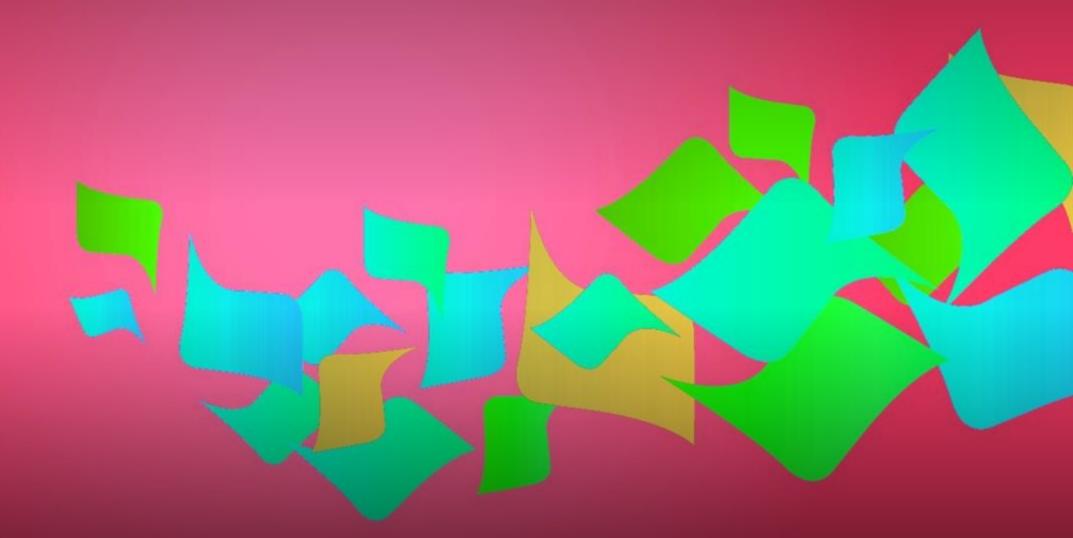
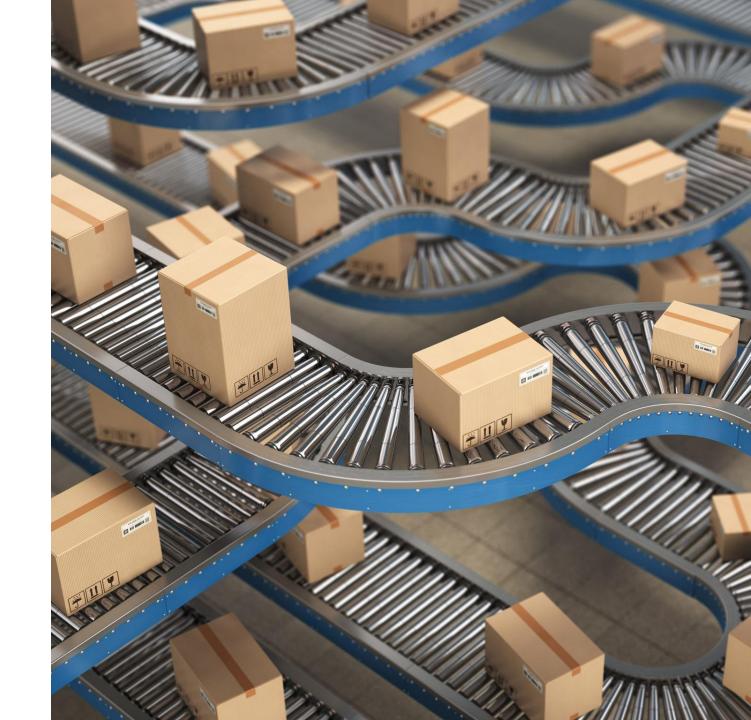
APPLICATION GATEWAY



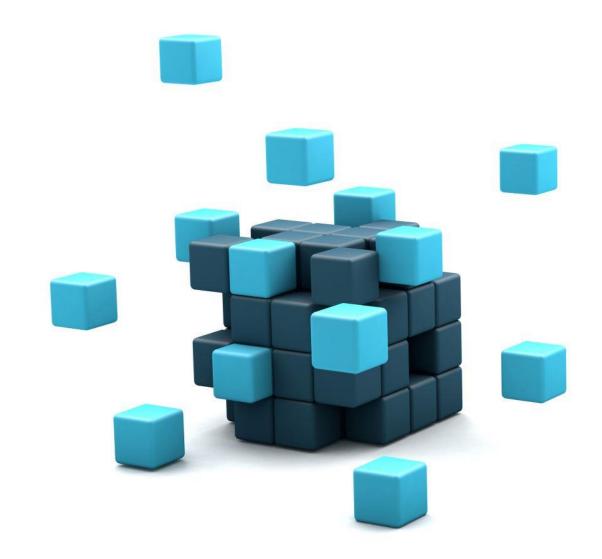
Introduction to Azure Load Balancer

- Azure Load Balancer distributes incoming network traffic across multiple VMs or services
- Provides high availability, fault tolerance, and better performance
- Supports both public and internal load balancing



Prerequisites

- An Azure subscription
- Virtual machines or services to load balance
- Availability Sets



Step 1: Sign in to the Azure Portal

- Go to portal.azure.com in your web browser
- Sign in with your Azure account credentials



Step 2: Create a Load Balancer

- Click on "+ Create a resource"
 in the upper-left corner
- Search for "Load Balancer"
 and select it from the results
- Click "Create" to start the creation process



Step 3: Basics Configuration



SELECT SUBSCRIPTION, RESOURCE GROUP, AND REGION



CHOOSE "PUBLIC" OR "INTERNAL" LOAD BALANCER



DEFINE A NAME FOR THE LOAD BALANCER

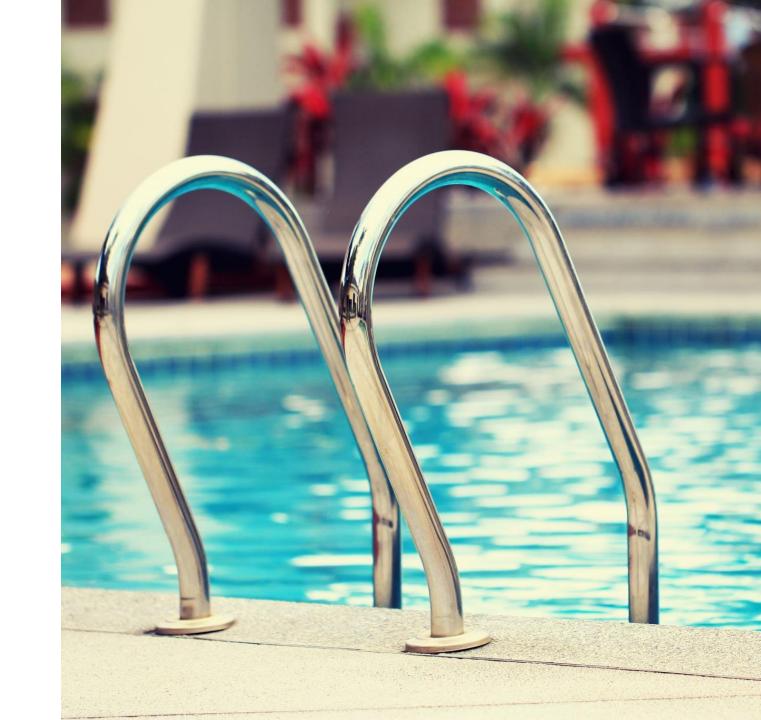
Step 4: Frontend IP Addresses

- Choose an existing public IP or create a new one
- Select an existing subnet



Step 5: Backend Pools

- Click "Add a backend pool."
- Give it a name
- Select backend VMs or services
- Click "Add" to add them to the pool



Step 6: Health Probes





CLICK "ADD A PROBE."

PROVIDE A NAME



CONFIGURE PROTOCOL, PORT, AND PROBING INTERVALS



DEFINE UNHEALTHY AND HEALTHY THRESHOLDS

Step 7: Load Balancing Rules

- Click "Add a load balancing rule."
- Give it a name
- Define the frontend IP and port
- Choose the backend pool and health probe
- Configure the protocol and backend port



Step 8: Review + Create

- Review all the configurations
- Click "Create" to deploy the Load Balancer



Step 9: Access and Test

- Obtain the public IP address from the overview page
- Update DNS settings or application configurations
- Test your services using the Load Balancer's IP address

```
modifier_ob.
  mirror object to mirror
mirror_mod.mirror_object
 peration == "MIRROR_X":
elror_mod.use_x = True
"Irror_mod.use_y = False
 lrror_mod.use_z = False
 operation == "MIRROR_Y"
irror_mod.use_x = False
 lrror_mod.use_y = True
 !rror_mod.use_z = False
  _operation == "MIRROR_Z"
  rror_mod.use_x = False
  rror_mod.use_y = False
  rror_mod.use_z = True
 melection at the end -add
   ob.select= 1
   er ob.select=1
   ntext.scene.objects.action
  "Selected" + str(modified
   irror ob.select = 0
  bpy.context.selected obj
   lata.objects[one.name].sel
  int("please select exactle
  OPERATOR CLASSES ----
    pes.Operator):
     X mirror to the selected
    ject.mirror_mirror_x"
 ontext):
ext.active_object is not
```