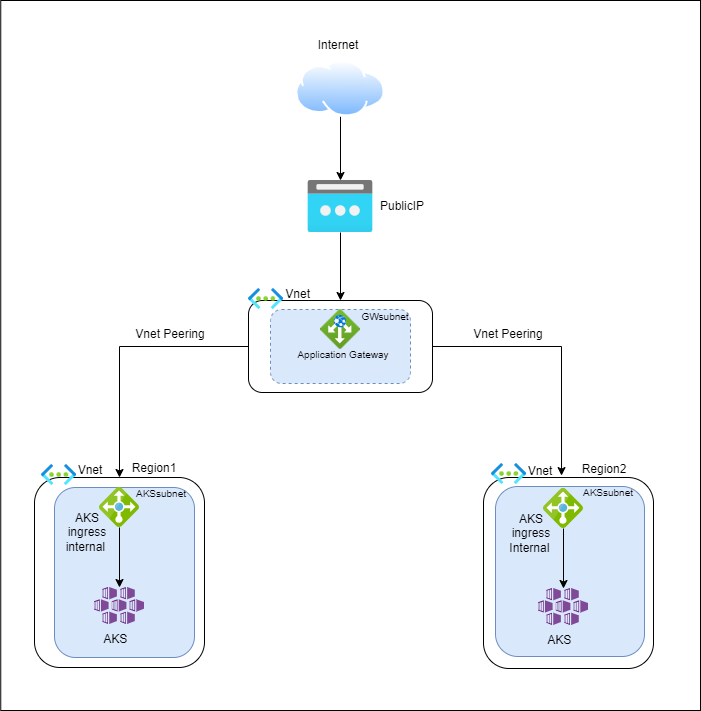
**AKS High Availability Architecture Documentation**

**Table of Contents**  
  
1. **Introduction**  
  - Purpose of the Documentation  
  - Overview of AKS High Availability Architecture  
  
2. **Architecture Components**  
  - Azure Application Gateway  
  - Two AKS Clusters in Different Regions  
  
3. **Configuration Steps**  
  - Setting up Ingress Controller in AKS Cluster  
  - Integrating Ingress Controller with Azure Application Gateway  
  - Adding AKS Clusters to the Backend Pool of Application Gateway  
  
4. **Benefits of the Architecture**  
  - Improved Reliability  
  - Efficient Traffic Distribution  
  - Enhanced Security Measures  
  
5. **Failover and Redundancy**  
  - Handling Failover Scenarios  
  - Redundancy Mechanisms  
  
6. **Performance Metrics**  
  - Metrics/Benchmarks to Evaluate Architecture Effectiveness

7. **Implementation Guidelines**  
  - Best Practices for AKS High Availability  
  - Considerations for Production Environments  
  
8. **Conclusion**  
   - Summary of Key Points  
   - Recommendations for Future Enhancements

1. **Introduction**  
     
    **1.1 Purpose of the Documentation**This documentation outlines the steps and considerations involved in creating a high-availability architecture using Azure Kubernetes Service (AKS), Azure Application Gateway, and multiple AKS clusters.  
     
    **1.2 Overview of AKS High Availability Architecture**  
     
   

**2. Architecture Components**

**2.1 Azure Application Gateway**Explanation of the role of Azure Application Gateway as a load balancer for routing traffic.

Application Gateway is used here for load balancing AKS cluster traffic which is present in two different geographical regions. **2.2 Two AKS Clusters in Different Regions**  
The rationale for using dual AKS clusters in different regions is to achieve geographical redundancy.

**3. Configuration Steps**

**3.1 Setting up Ingress Controller in AKS Cluster**Detailed steps for configuring Ingress Controller within each AKS cluster to manage external access to applications.

Give the role of network contributor to the AKS cluster on the Vnet to which you want to assign the ingress controller.

Steps: Go to the Azure portal and navigate to the>> Virtual networks >>Access role Assignments>>Add role assignments>> Select “Network contributor” as a role>> Select your AKS cluster as a member>> Review and assign the role.

**Install ingress on the AKS cluster**

Steps:

**Connect to the AKS cluster-**

az aks get-credentials --resource-group <ResourceGroupName> --name <AKSClusterName>

**Install ingress**

Add the NGINX Helm repository  
helm repo add ingress-nginx <https://kubernetes.github.io/ingress-nginx>  
# Install NGINX Ingress Controller  
helm install nginx-ingress ingress-nginx/ingress-nginx \  
 --namespace <YourNamespace> \  
 --set controller.service.externalTrafficPolicy=Local \  
 --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-internal"="true" \  
 --set controller.scope.enabled=true \  
 --set controller.scope.namespace=<YourNamespace>

**Expose service using Ingress**  
 Now, you can create Ingress resources to expose your services. For example, create a YAML file (e.g., my-ingress.yaml) with an Ingress

definition:  
apiVersion: [networking.k8s.io/v1](http://networking.k8s.io/v1)  
kind: Ingress  
metadata:  
 name: my-ingress  
 namespace: <YourNamespace>  
 annotations:  
[nginx.ingress.kubernetes.io/rewrite-target:](http://nginx.ingress.kubernetes.io/rewrite-target:) /  
spec:  
 rules:  
 - host: <your-domain>  
   http:  
     paths:  
     - path: /  
    pathType: Prefix  
       backend:  
         service:  
           name: <your-service>  
           port:  
             number: <your-service-port>  
  
Replace placeholders (<YourNamespace>, <your-domain>, <your-service>, <your-service-port>) with your actual values.  
  
  
**Apply the Ingress configuration**  
  
kubectl apply -f my-ingress.yaml

**3.2 Integrating Ingress Controller with Azure Application Gateway**  
 Step-by-step guide on integrating Ingress Controller with Azure Application Gateway.

To integrate ingress controller with azure application gateway you have to add your external IP of the ingress service to the backend pool of the Azure application gateway. Refer 3.3 point

**3.3 Adding AKS Clusters to the Backend Pool**  
Explanation of the process for adding AKS clusters to the backend pool of the Azure Application Gateway for efficient traffic distribution.

Navigate to the Azure portal>>search application gateway>>go to the backend pool option>>add backend pool>>provide required attributes Name, IP or Hostname (external IP of your ingress controller)>>review and create.  
  
 **4. Benefits of the Architecture**  
  
 4.1 Improved Reliability  
Two clusters / Two regions Two AKS clusters are deployed, each in a separate Azure region. During normal operations, network traffic is routed between all regions. If one region becomes unavailable, traffic is routed to a secondary region.  
  
  
This documentation provides a comprehensive guide to understanding, implementing, and maintaining the AKS high availability architecture using Azure Application Gateway and dual AKS clusters.

**5. Failover and Redundancy**

There are traffic failover rules to redirect traffic to a healthy AKS cluster in case of failure.

Utilized application gateway priority-based routing methods.

Leveraged Azure’s global data center presence to deploy AKS cluster in different geographical regions

6. **Performance Metrics**

* Azure Monitor Metrics:  
    - AKS Metrics: Leverage Azure Monitor to collect and analyze AKS-specific metrics like CPU usage, memory usage, and node health.
* Load Balancer Performance:  
    - Load Balancer Metrics: Monitor metrics related to the performance and health of any load balancers used in the architecture.
* Application Response Time:  
    - End-to-End Application Response Time: Measure the end-to-end response time of critical applications across regions to ensure acceptable user experience.