An Azure Application Gateway is a web traffic load balancer that helps manage and route traffic to different backend services based on the URL path or host header. It is a layer 7 load balancer, which means that it can make routing decisions based on application-specific data, such as the contents of an HTTP request.

Some of the key features and benefits of an Azure Application Gateway include:

Routing rules: Application Gateway allows you to create routing rules to route traffic to different backend services based on the URL path or host header. You can also use routing rules to define HTTP headers and cookie-based session affinity.

SSL termination: Application Gateway can terminate SSL/TLS traffic, decrypting HTTPS traffic at the gateway and forwarding it to the backend servers as HTTP traffic.

Web Application Firewall (WAF): Application Gateway includes a WAF that provides protection against common web application vulnerabilities, such as SQL injection and cross-site scripting (XSS).

Autoscaling: Application Gateway can automatically scale up or down based on traffic demands. You can configure autoscaling rules based on metrics such as CPU usage and request rate.

Health probes: Application Gateway can periodically check the health of backend servers to ensure that they are available and responding to requests.

URL-based routing: Application Gateway allows you to route traffic based on URL paths, which can be useful for directing traffic to different backend services based on the URL.

Overall, Azure Application Gateway provides a scalable and highly available way to manage and route web traffic to backend services. It is a powerful tool for building modern, cloud-native applications in Azure.

The maximum number of scale units that can be configured for an Azure Application Gateway depends on the SKU (Standard or WAF) and the region in which it is deployed.

For the Standard SKU, the maximum number of scale units is 10, while for the WAF SKU, the maximum number of scale units is 5.

However, it is important to note that adding more scale units does not always translate to better performance or scalability. The number of scale units required for an Application Gateway depends on factors such as the size of the virtual machines, the amount of traffic, and the configuration settings of the Application Gateway.

It is recommended to start with a smaller number of scale units and monitor the performance of the Application Gateway. If additional capacity is needed, scale units can be added incrementally until the desired level of performance and scalability is achieved.

In Azure Application Gateway, a scale unit is a set of compute resources that are used to process incoming web traffic. A scale unit consists of a set of virtual machines that are responsible for processing traffic for a particular instance of an Application Gateway.

Each scale unit has its own configuration and set of backend servers, which allows you to scale and manage traffic for different applications independently. You can also configure multiple scale units within a single Application Gateway for high availability and increased capacity.

Here are some key points about scale units in Application Gateway:

Size: Each scale unit is comprised of one or more virtual machines, which can be scaled up or down based on traffic demands. The size and number of virtual machines in a scale unit can be adjusted based on the needs of your application.

Configuration: Each scale unit has its own configuration settings, including routing rules, SSL settings, and backend server pools. These settings can be configured independently for each scale unit to provide maximum flexibility and control.

Availability: Application Gateway uses active/standby deployment for scale units, which means that traffic is processed by one scale unit at a time while the other scale unit remains in standby mode. In the event of a failure, traffic is automatically redirected to the standby scale unit.

Capacity: By adding additional scale units, you can increase the capacity of your Application Gateway and improve its ability to handle traffic spikes and increased demand.

Overall, scale units provide a scalable and highly available way to process incoming web traffic in Azure Application Gateway. By configuring multiple scale units with independent configurations and backend server pools, you can achieve maximum flexibility and control over your application traffic.

In Azure Application Gateway, a frontend IP address is a public or private IP address that is used to receive incoming traffic from the internet or a virtual network.

A frontend IP address is associated with a listener, which is used to route traffic to a backend pool of resources based on the incoming URL, hostname, or path.

Here are some key points about frontend IP addresses in Application Gateway:

Types of frontend IP addresses: There are two types of frontend IP addresses that can be used with Application Gateway: public IP addresses and private IP addresses. Public IP addresses are used to receive traffic from the internet, while private IP addresses are used to receive traffic from a virtual network.

Multi-tenant support: Application Gateway supports multi-tenant frontends, which means that multiple listeners can use the same frontend IP address to receive traffic. This allows you to consolidate traffic for multiple applications onto a single IP address.

SSL termination: If SSL termination is enabled for a listener, the frontend IP address is used to terminate the SSL connection before the traffic is routed to the backend pool of resources. This provides an additional layer of security for your application.

Availability: You can associate multiple frontend IP addresses with an Application Gateway for high availability and increased capacity. If one frontend IP address becomes unavailable, traffic can be redirected to the other available frontend IP addresses.

Overall, frontend IP addresses are an important component of Azure Application Gateway that allow you to receive incoming traffic from the internet or a virtual network, and route it to the appropriate backend pool of resources based on the incoming URL, hostname, or path.

In Azure Application Gateway, a backend is a collection of resources that receive incoming traffic from the frontend and process it. The backend can be composed of one or more virtual machines, virtual machine scale sets, IP addresses, or fully qualified domain names (FQDNs).

Here are some key points about backends in Application Gateway:

Backend pools: A backend pool is a collection of resources that can be used to process incoming traffic. The resources in a backend pool can be homogeneous or heterogeneous, and can include virtual machines, virtual machine scale sets, IP addresses, or FQDNs. You can create multiple backend pools for an Application Gateway to handle traffic for different applications.

Health probes: Application Gateway uses health probes to periodically check the availability and health of the resources in a backend pool. If a resource is detected to be unhealthy, Application Gateway will stop routing traffic to that resource until it becomes healthy again.

Load balancing: Application Gateway uses load balancing algorithms to distribute incoming traffic across the resources in a backend pool. There are different load balancing algorithms to choose from, such as round-robin, least connections, and IP hash.

SSL termination: If SSL termination is enabled for a listener, Application Gateway can terminate the SSL connection and decrypt the traffic before routing it to the backend pool.

Affinity: Application Gateway supports session affinity, which means that all requests from a particular client are routed to the same resource in the backend pool. This can be useful for applications that require stateful connections.

Overall, backends are a critical component of Azure Application Gateway that enable you to process incoming traffic from the frontend and route it to the appropriate resources in the backend pool based on your application's requirements. By configuring multiple backend pools with health probes and load balancing algorithms, you can achieve high availability, scalability, and performance for your applications.

In Azure Application Gateway, a backend pool is a collection of resources that are used to process incoming traffic from the frontend. The resources in a backend pool can be homogeneous or heterogeneous, and can include virtual machines, virtual machine scale sets, IP addresses, or fully qualified domain names (FQDNs).

Here are some key points about backend pools in Application Gateway:

Resource types: A backend pool can include resources of different types, such as virtual machines, virtual machine scale sets, IP addresses, or FQDNs. This allows you to create a heterogeneous backend pool that can handle different types of traffic.

Health probes: Application Gateway uses health probes to check the availability and health of the resources in a backend pool. You can configure the interval, timeout, and threshold for the health probe. If a resource is detected to be unhealthy, Application Gateway will stop routing traffic to that resource until it becomes healthy again.

Load balancing algorithms: Application Gateway uses load balancing algorithms to distribute incoming traffic across the resources in a backend pool. There are different load balancing algorithms to choose from, such as round-robin, least connections, and IP hash.

Affinity: Application Gateway supports session affinity, which means that all requests from a particular client are routed to the same resource in the backend pool. This can be useful for applications that require stateful connections.

Multi-tenant support: Application Gateway supports multi-tenant backend pools, which means that you can use the same backend pool for multiple listeners. This allows you to consolidate traffic for multiple applications onto a single backend pool.

SSL termination: If SSL termination is enabled for a listener, Application Gateway can terminate the SSL connection and decrypt the traffic before routing it to the backend pool.

Overall, backend pools are a critical component of Azure Application Gateway that allow you to process incoming traffic from the frontend and route it to the appropriate resources based on your application's requirements. By configuring health probes, load balancing algorithms, and session affinity, you can achieve high availability, scalability, and performance for your applications.

In Azure Application Gateway, the term "target type" refers to the type of resource that is included in a backend pool. A backend pool is a collection of resources that are used to process incoming traffic from the frontend, and can include various types of resources such as virtual machines, virtual machine scale sets, IP addresses, or fully qualified domain names (FQDNs).

When you add a resource to a backend pool, you need to specify its target type, which tells Application Gateway how to interact with that resource. Here are the different target types that are supported in Application Gateway:

IP address: This is the simplest target type, and refers to a specific IP address that is associated with a backend resource. You can specify a single IP address, or a range of IP addresses.

FQDN: This target type allows you to specify a fully qualified domain name (FQDN) that is associated with a backend resource. This is useful if your backend resources are hosted on a cloud service that uses dynamic IP addresses.

Virtual machine: This target type refers to a specific virtual machine that is associated with a backend resource. You need to specify the virtual machine's IP address or FQDN, and also provide credentials for Application Gateway to authenticate with the virtual machine.

Virtual machine scale set: This target type refers to a set of virtual machines that are managed as a group. You need to specify the virtual machine scale set's name and resource group, and also provide credentials for Application Gateway to authenticate with the virtual machines.

By supporting different target types, Application Gateway allows you to create a flexible and scalable backend infrastructure that can handle different types of traffic. By configuring health probes, load balancing algorithms, and session affinity for each backend pool, you can achieve high availability, scalability, and performance for your applications.

In Azure Application Gateway, a routing rule is a set of conditions that determine how incoming traffic from the frontend is routed to the appropriate backend pool. A routing rule is associated with a listener, which is responsible for listening for incoming traffic on a specific port and protocol.

Here are some key points about routing rules in Application Gateway:

Conditions: A routing rule consists of one or more conditions that must be met in order for the rule to be applied. Conditions can be based on the URL path, host header, query string, HTTP method, or a custom header. You can define multiple conditions for a single routing rule, and they can be combined using logical operators (AND, OR, NOT).

Backend pool: Once a routing rule is applied, incoming traffic is routed to the appropriate backend pool. You can specify a single backend pool for a routing rule, or use multiple backend pools in a weighted round-robin fashion.

HTTP settings: Each routing rule can have its own set of HTTP settings, which control the behavior of HTTP requests and responses. You can configure settings such as request timeout, response timeout, maximum request size, and whether to preserve the original host header.

URL rewrite: Application Gateway supports URL rewrite, which allows you to modify the URL path or query string of incoming requests before they are forwarded to the backend pool. This can be useful for scenarios such as redirecting requests to a different URL, or removing query string parameters.

SSL termination: If SSL termination is enabled for a listener, Application Gateway can terminate the SSL connection and decrypt the traffic before applying the routing rule. This allows you to route traffic based on the contents of the decrypted traffic.

Overall, routing rules are a critical component of Azure Application Gateway that allow you to route incoming traffic from the frontend to the appropriate backend pool based on your application's requirements. By defining conditions, specifying backend pools, and configuring HTTP settings and URL rewrite, you can achieve high availability, scalability, and performance for your applications.

In Azure App Gateway, a listener is a component that listens for incoming traffic on a particular protocol and port. When a request is received, the listener processes the request and forwards it to the backend pool or backend server associated with it.

To create a listener in Azure App Gateway, you need to perform the following steps:

Open the Azure portal and go to your App Gateway resource.
Click on "Listeners" in the left-hand menu.
Click on the "Add" button to create a new listener.
Select the protocol and port that you want the listener to use (e.g., HTTP on port 80).
Choose the SSL certificate if you want to use HTTPS.
Select the backend pool or backend server to which the listener should forward traffic.
Save the listener configuration.
You can create multiple listeners on an Azure App Gateway to support different protocols, ports, and SSL certificates. Additionally, you can configure the listeners to support end-to-end encryption, WAF policies, and custom error pages, among other things.

In Azure App Gateway, there are several types of listeners that you can create based on the protocol and port used to listen for incoming traffic. The following are the most common listener types in Azure App Gateway:

HTTP listener: This type of listener listens for incoming HTTP traffic on port 80 or any other port that you specify. HTTP listeners can be used to redirect HTTP traffic to HTTPS, apply SSL offload, and enable cookie-based session affinity.

HTTPS listener: This type of listener listens for incoming HTTPS traffic on port 443 or any other port that you specify. HTTPS listeners require a valid SSL certificate to secure the traffic.

Multi-Site listener: This type of listener allows you to listen for traffic on multiple domains or subdomains using a single IP address and port combination. Multi-Site listeners are useful when you want to host multiple websites on a single App Gateway.

Web Application Firewall (WAF) listener: This type of listener is used to protect your web applications from common web attacks, such as SQL injection and cross-site scripting (XSS). WAF listeners can be used in conjunction with HTTP or HTTPS listeners.

TCP listener: This type of listener allows you to listen for traffic on a specific TCP port, such as SMTP or FTP. TCP listeners can be used for non-HTTP/HTTPS traffic, such as email or file transfer protocols.

UDP listener: This type of listener allows you to listen for traffic on a specific UDP port, such as DNS or DHCP. UDP listeners can be used for real-time applications, such as voice over IP (VoIP) and streaming media.

You can create multiple listeners of different types on an Azure App Gateway to support various use cases and protocols.

Sure, here are the most basic listener types in Azure App Gateway:

HTTP listener: This type of listener listens for incoming HTTP traffic on port 80 or any other port that you specify. It's the most basic listener type and is commonly used to redirect HTTP traffic to HTTPS or to apply SSL offload.

HTTPS listener: This type of listener listens for incoming HTTPS traffic on port 443 or any other port that you specify. It requires a valid SSL certificate to secure the traffic, and it's the most basic way to enable HTTPS on your web application.

You can use these basic listener types to get started with Azure App Gateway and secure your web application traffic. As your requirements evolve, you can explore other listener types, such as multi-site listeners, WAF listeners, TCP listeners, and UDP listeners, to support more advanced scenarios.

In Azure App Gateway, you can create custom error pages for HTTP status codes 4xx and 5xx. When a user requests a page that returns an error status code, the App Gateway can redirect the user to a custom error page that you define.

To create a custom error page in Azure App Gateway, you need to perform the following steps:

Create an HTML file that contains the content of your custom error page.
Upload the HTML file to a storage account or a web server that's accessible from the App Gateway.
Open the Azure portal and go to your App Gateway resource.
Click on "HTTP settings" in the left-hand menu.
Click on the HTTP setting that you want to configure.
Scroll down to the "Custom error configuration" section and click on "Add".
Enter the HTTP status code that you want to configure (e.g., 404 for "Not Found").
Enter the path to the custom error page file that you uploaded in step 2 (e.g., /errorpages/404.html).
Save the configuration.
Repeat these steps for each HTTP status code that you want to configure a custom error page for.

Note that you can also configure a default error page that's displayed for all HTTP status codes that don't have a custom error page configured. To configure a default error page, simply create an HTML file named "default.html" and upload it to the storage account or web server. Then, in the App Gateway HTTP settings, set the "Default backend path" to the path of the default error page file (e.g., /errorpages/default.html).

In Azure App Gateway, a backend target is a resource or endpoint that the gateway routes traffic to. The backend target can be an Azure Virtual Machine, a container instance, an Azure Web App, an Azure Function, or any other HTTP or HTTPS endpoint that's accessible over the internet.

To configure a backend target in Azure App Gateway, you need to perform the following steps:

Open the Azure portal and go to your App Gateway resource.
Click on "Backend pools" in the left-hand menu.
Click on "Add" to create a new backend pool.
Enter a name for the backend pool.
Select the backend pool's protocol (HTTP or HTTPS).
Select the backend pool's load-balancing algorithm (Round Robin, Least Connections, or IP Hash).
Add one or more backend targets to the pool by specifying their IP addresses, FQDNs, or resource IDs.
Save the backend pool configuration.

Once you've configured a backend pool, you can associate it with a listener to route traffic to the backend targets. When a user makes a request to the App Gateway, the gateway inspects the request and forwards it to one of the backend targets in the associated backend pool based on the load-balancing algorithm.

Note that you can configure multiple backend pools and associate them with different listeners to support different scenarios. For example, you can create a backend pool for your web app and another backend pool for your API, and associate each pool with a separate listener.

In Azure App Gateway, you can configure backend targets of different types depending on the requirements of your application. Here are some of the most common target types:

Virtual machine: You can configure a backend target as an Azure virtual machine (VM) that's running a web server, such as IIS or Apache. This allows you to route traffic to the VM's web server and load balance traffic across multiple VMs if needed.

Azure Web App: You can configure a backend target as an Azure Web App, which is a fully managed platform for building, deploying, and scaling web applications. This allows you to easily scale your web app and take advantage of features such as automatic OS patching and deployment slots.

Container instance: You can configure a backend target as an Azure Container Instance, which is a lightweight, standalone container that can run a single service or application. This allows you to deploy and scale containerized applications without having to manage the underlying infrastructure.

Azure Function: You can configure a backend target as an Azure Function, which is a serverless compute service that allows you to run code in response to events. This allows you to build serverless APIs and microservices that can scale automatically based on demand.

External endpoint: You can configure a backend target as an external HTTP or HTTPS endpoint that's accessible over the internet. This allows you to route traffic to an endpoint outside of your Azure environment, such as a third-party API or a partner service.

These are just a few examples of the target types that you can configure in Azure App Gateway. When choosing a target type, consider factors such as scalability, manageability, and security to ensure that the target meets the requirements of your application.

Backend health refers to the status of a backend target in Azure App Gateway. The gateway periodically checks the health of each backend target to ensure that it's available and responsive. If a backend target is not responding or is responding with errors, the gateway marks it as unhealthy and stops routing traffic to it.

To monitor the health of backend targets in Azure App Gateway, you can use the following features:

Health probes: A health probe is a request that the gateway sends to a backend target to check its availability and responsiveness. You can configure a health probe with a specific HTTP or HTTPS path and a timeout value. The gateway sends probes to each backend target at a regular interval, and if a target doesn't respond within the specified timeout period or returns an error status code, the gateway marks it as unhealthy.

Backend health monitoring: Azure App Gateway provides a built-in backend health monitoring feature that allows you to view the health status of each backend target in real-time. You can see the number of healthy and unhealthy targets, as well as the reason for any target that's marked as unhealthy.

Custom health probes: If the built-in health probes don't meet your requirements, you can create custom probes using PowerShell or the Azure portal. Custom probes allow you to test specific functionality of your backend targets and can help you detect issues that the built-in probes might not catch.

By monitoring the health of backend targets in Azure App Gateway, you can ensure that traffic is routed only to healthy targets, which helps improve the availability and reliability of your application.
