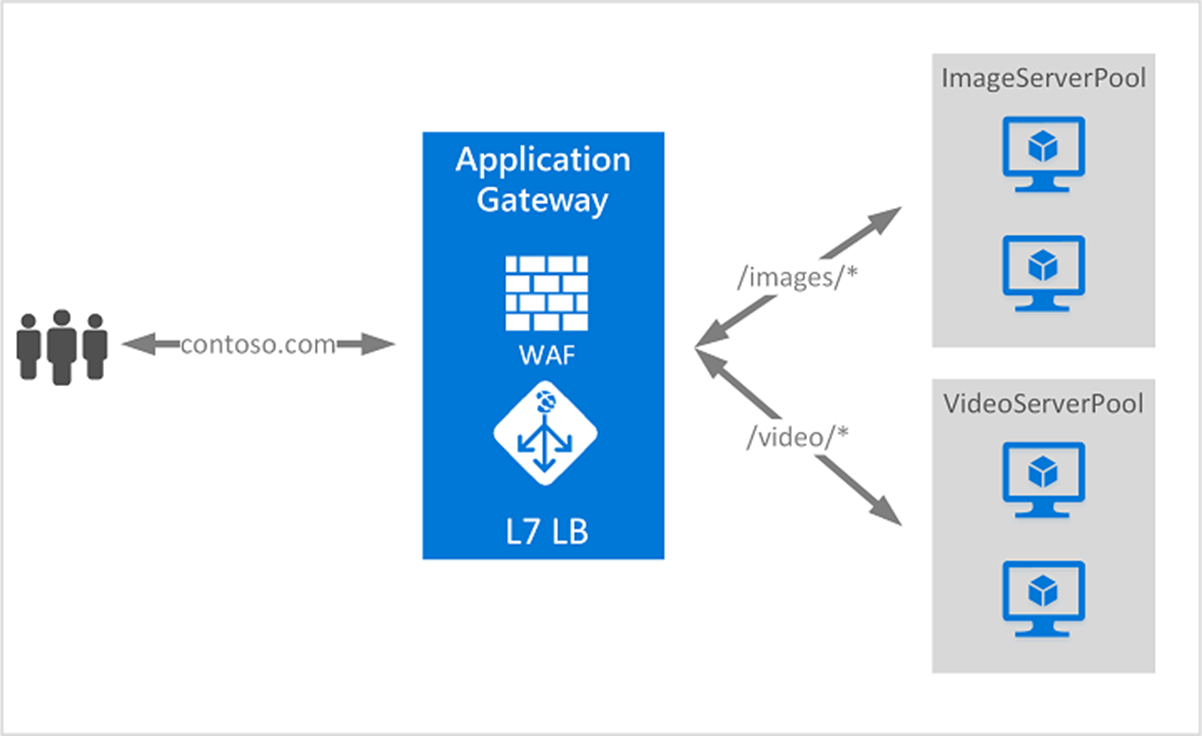
Module 5: Load balance HTTP(S) traffic in Azure

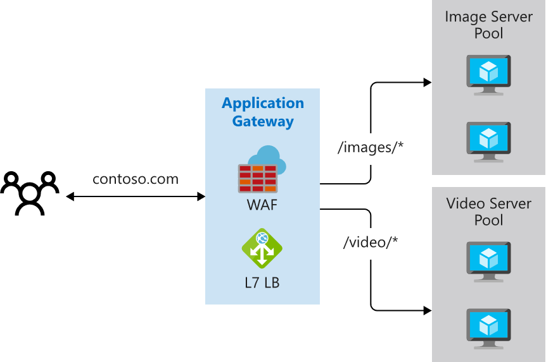
# Chapter 2: Design Azure Application Gateway

* Azure App Gateway (AG): web traffic LB that enables you to manage traffic to web apps.
  + Application Layer (OSI layer 7) load balancing:
    - Route based on attributes of an HTTP request, such as URI path or host headers.
      * **Ex.** route traffic based on incoming URL (/images) route traffic to specific set of servers pool for images.
      * 
  + **Features:**
    - Support for HTTP, HTTPS, HTTP/2 and WebSocket protocols.
    - Web application firewall (WAF) to protect against web app vulnerabilities.
    - End-to-end request encryption.
    - Auto-scaling to adjust capacity based on web traffic.
    - **Redirection**: to another site, or from HTTP to HTTPS.
    - Rewrite HTTP headers:
      * Headers pass parameter information with the request or the response.
    - Custom error pages

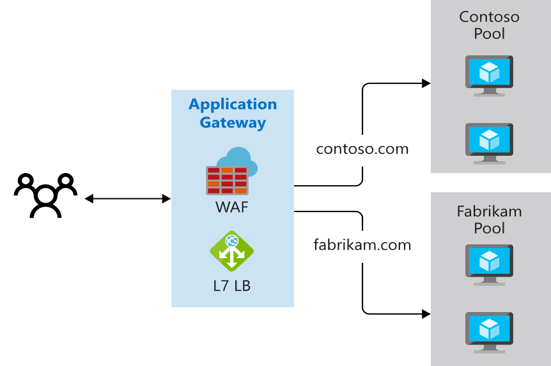
## Determine AG routing

* Client requests to web apps are sent to IP address/DNS name of the gateway.
  + AG then determines which pool to go to

### Method 1: Path-based routing

* Direct requests with different URL paths to diff back-end server pools.
  + 

### Method 2: Multiple site routing

* Configures more than one web app on the same application gateway instance.
  + Register multiple DNS names (CNAMEs) for IP address of AG and specify the name of each site (web app)
    - Requests will go to diff back-end pools dedicated to a site
    - 

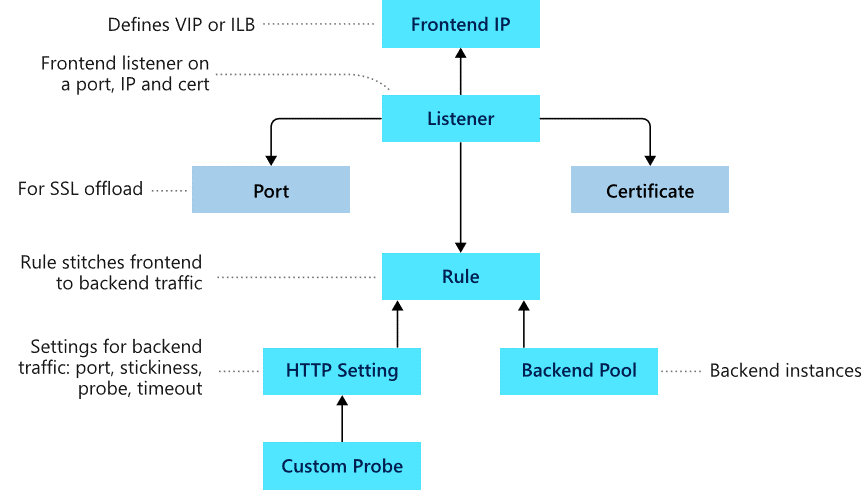
## Choosing between Azure AG v2 and WAF v2 SKUs

* Many factors but, scaling strategy is very important!

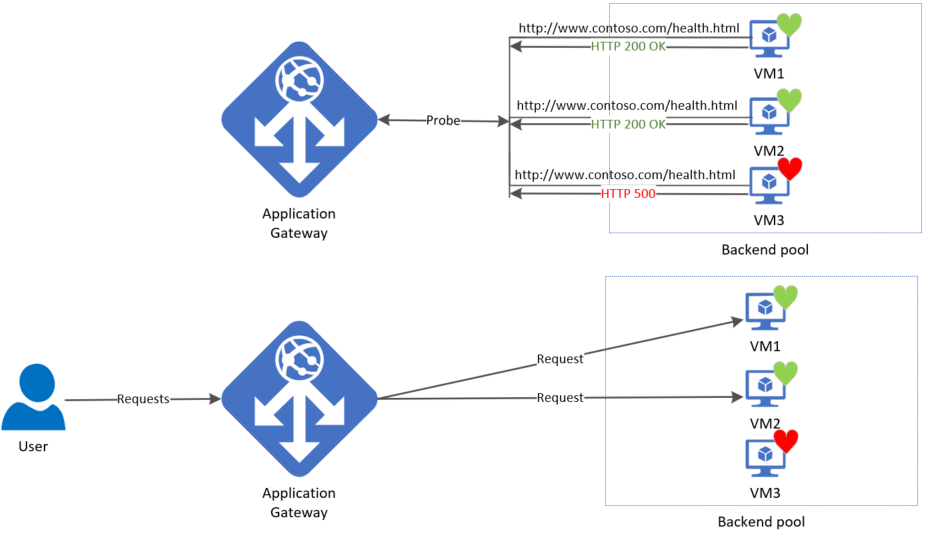
### Scaling Application Gateway and WAF v2

* 2 Models for Scaling:
  + **Autoscaling**:
    - AG & WAF v2 SKUs scale up/down based on traffic needs (elasticity)
      * Allows cost saving & elasticity, but you need to set min/max instance count.
      * Each instance has approx. 10 additional reserved Capacity Units.
        + (If set to Zero = no reserved capacity and is pure auto-scaling).
      * By specifying max instance count, ensures AG doesn't scale beyond the specified number of instances. Default value max = 20.
  + **Manual**:
    - No auto-scale so mandatory to specify instance count.
      * Instance count can vary from 1 to 125 instances.

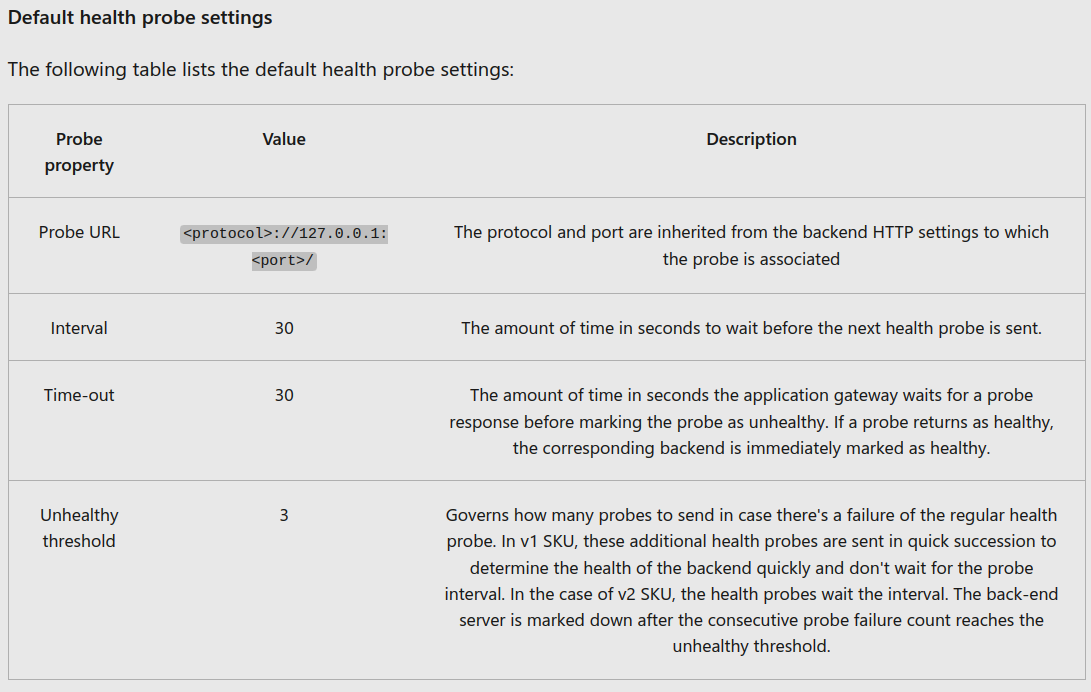
# Chapter 3: Configure Azure Application Gateway

* **Components**
  + 
    - **Front-end Config**
      * You can set a public IP address, a private IP address, or both on an AG.
        + Public IP address required to host a back-end that clients access over the Internet via an Internet-facing virtual IP.
    - **Back-end Config**
      * Used to route requests to the back-end servers that serve the request.
        + **Ex**. NICs, VMSS, Public IPs, internal IPs, FQDNs, and multi-tenant back-ends like Azure App Service.

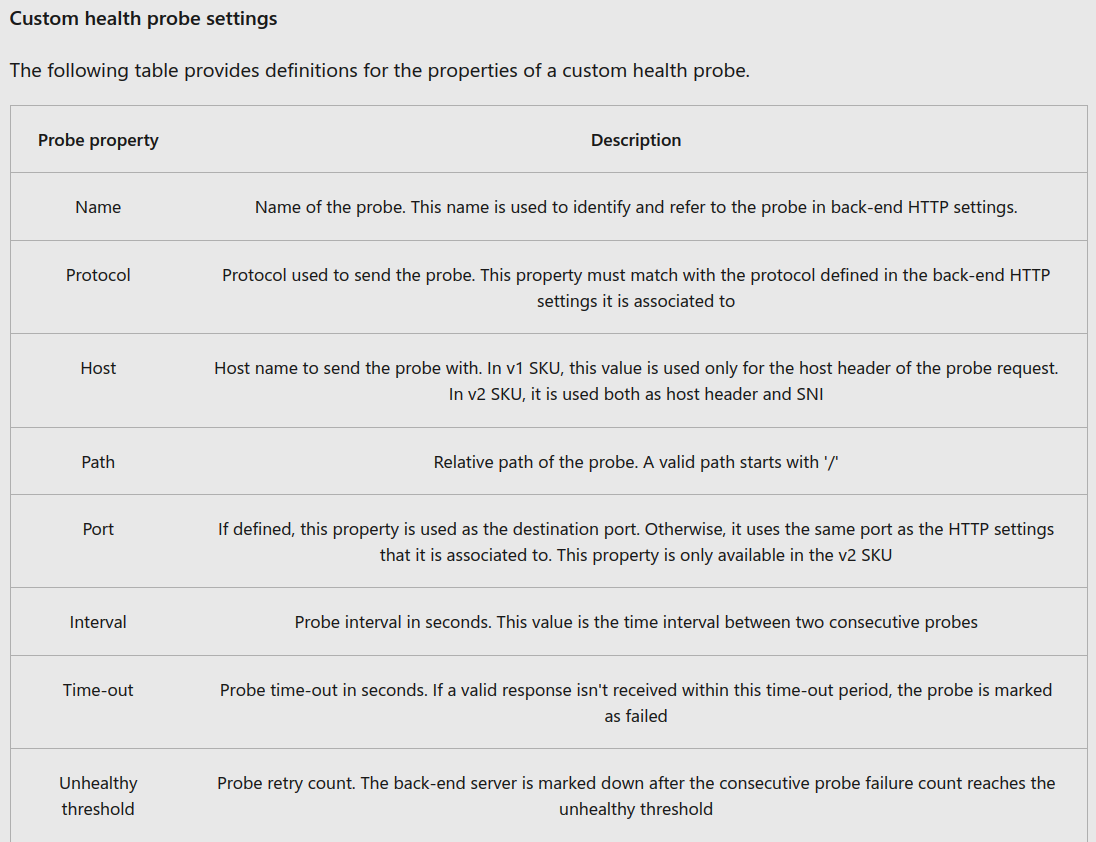
## Configure health probes

* By default, AGs send the health probes with the same port that is defined in the back-end HTTP settings. But you can use a custom port!
  + This monitors the health & removes unhealthy resources from the pool
  + Source IP address that AG uses for health probes depends on the back-end pool:
    - If server address is public, source address is AG’s front-end public IP.
    - If server address is private, source IP is from AG subnet's private IP address space.
    - 

## Default health probe

* Default health probe is auto-configured if no custom probe are configured.
  + Monitoring works by making an HTTP(s) GET request to the IP/FQDN of the back-end pool.
    - Healthy HTTP response = status code 200-399 which is checked every 30 seconds. Unhealthy servers still get checked but AG won’t forward requests there till it is healthy
  + 

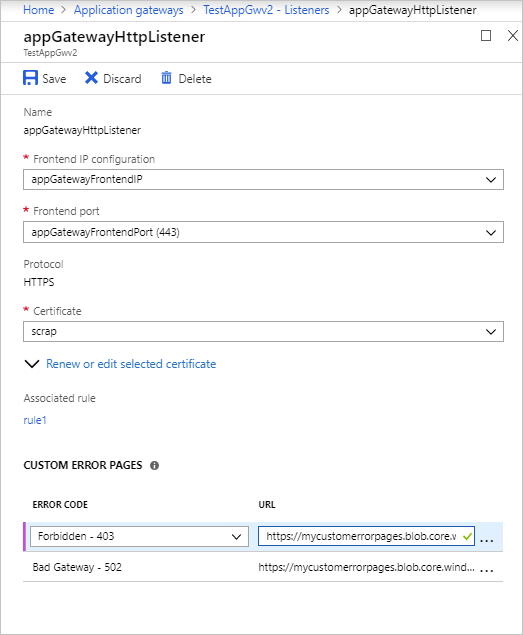
## Custom health probe

* 

## Probe matching

* **Default**: an HTTP(S) response w/ status code 200-399 is considered **healthy**.
* Custom probes have 2 other criteria:
  + HTTP response *status code* match – Match user specified HTTP response code/ranges.
  + HTTP response *body* match – Match based on HTTP response body and matches with a user specified string.
    - Looks for presence of user specified string in response body NOT full RegEx match.

## Configure listeners

* Listener checks for incoming connection requests by port #, protocol, host, and IP
  + **Example**:
    - 

### Listener types

* **Basic**: All requests for any domain will be accepted and forwarded to back-end pools.
* **Multi-site**: Forward requests to different back-end pools based on the host header or host names.

### Order of processing listeners

* **v1 SKU**: requests are matched according to the order of the rules and the type of listener.
* **v2 SKU:** multi-site listeners are processed before basic listeners.

### Front-end IP, Port, Protocol

* Choose front-end IP to associate with this listener (incoming requests on this IP).
* Choose front-end port (existing port or new).
  + Port used for public-facing or private listeners.
* Choose HTTP/HTTPS:
  + **HTTP**: unencrypted traffic from client to AG.
  + **HTTPS**: encrypted traffic from client to AG
    - Enables TLS termination or end-to-end TLS encryption.

## Redirection overview

* AG has generic redirection mechanism: *Redirect traffic from one listener to another listener/external site*.
  + Simplifies app config, optimizes resource usage, supports new redirection scenarios (global and path-based).
* Common redirection scenario for web apps is automatic HTTP to HTTPS redirection to ensure encrypted traffic occurs
  + Add a new redirect config to routing rule and specifying another listener with HTTPS protocol as the target listener.
* **Redirection Types**:
  + 301 (Permanent Redirect), 302 (Found), 303 (See Other), 307 (Temporary Redirect)
* **Capabilities of AG redirection**:
  + *Global redirection –* Redirect from one listener to another listener on AG. (HTTP to HTTPS redirection)
  + *Path-based redirection* – Enables HTTP to HTTPS redirection only on a specific site area (**ex**. URL /cart/\*).
  + *Redirect to external site*: Require new redirect configuration object (**ex**. target listener or external site to which redirection is desired).

## Application Gateway request routing rules

* Created by default, this rule binds default **listener** (appGatewayHttpListener) w/ default **back-end pool** (appGatewayBackendPool) and default **back-end HTTP settings** (appGatewayBackendHttpSettings)
  + 2 Types:
    - Basic – all req. to associated listener goes to single back-end pool
    - Path-based – routes requests to specific pool based on URL path
* **Order of processing rules:** *Processed in the order that paths are listed*
* **Associated Listener**:
  + Request-routing rule associated w/ the listener evaluates to determine the back-end pool to route the request to.
* **Associated back-end pool**:
  + Back-end pool that the rule determines contains should serve requests that the listener receives.
    - Basic or Path-based
* **Associated back-end HTTP setting**:
  + Back-end HTTP setting for each rule (Basic or Path-based)
    - Port number, protocol, and other information that's specified in this setting is used to determine how to route requests from AG to the back-end back-end pool.

## Configure URL-based routing

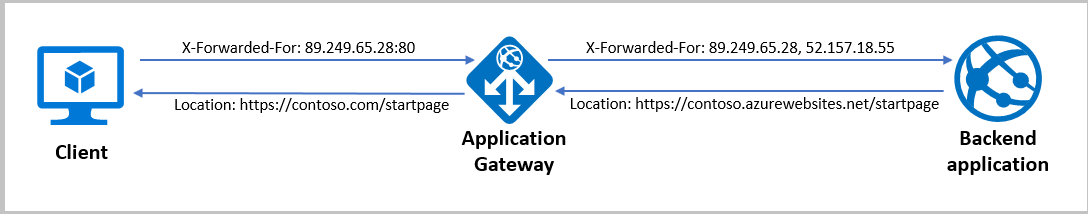
* Allows routing of traffic to back-end server pools based on URL Paths of the request
  + v1 SKU, rules processed in order they are listed in portal.
  + v2 SKU, exact matches have higher precedence.

## Configure rewrite policies in AG

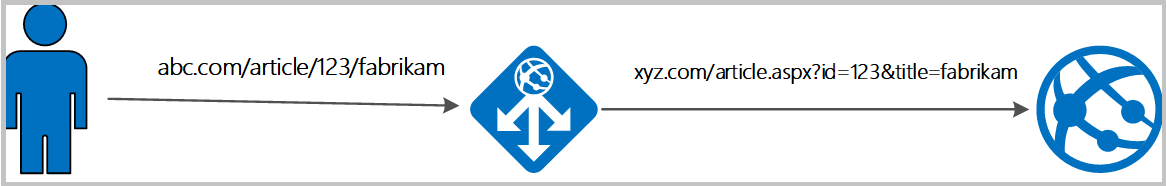
* Rewrite selected content of requests and responses based on specific conditions (**ex**. req/response info)

### Supported rewrite types

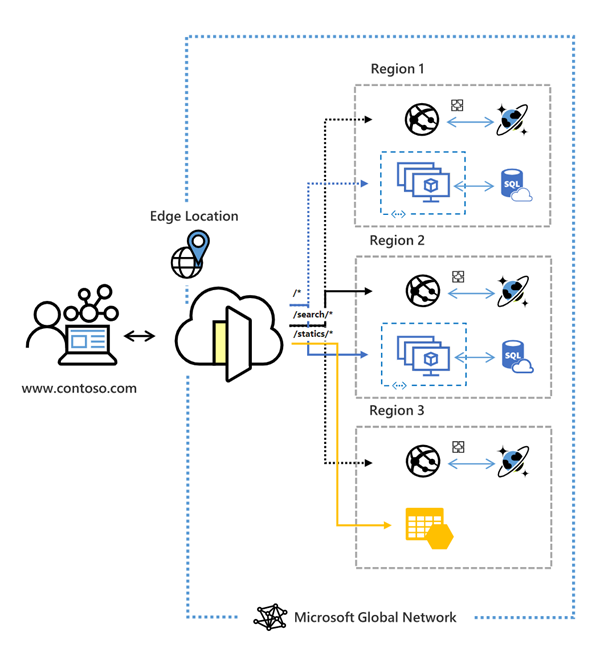
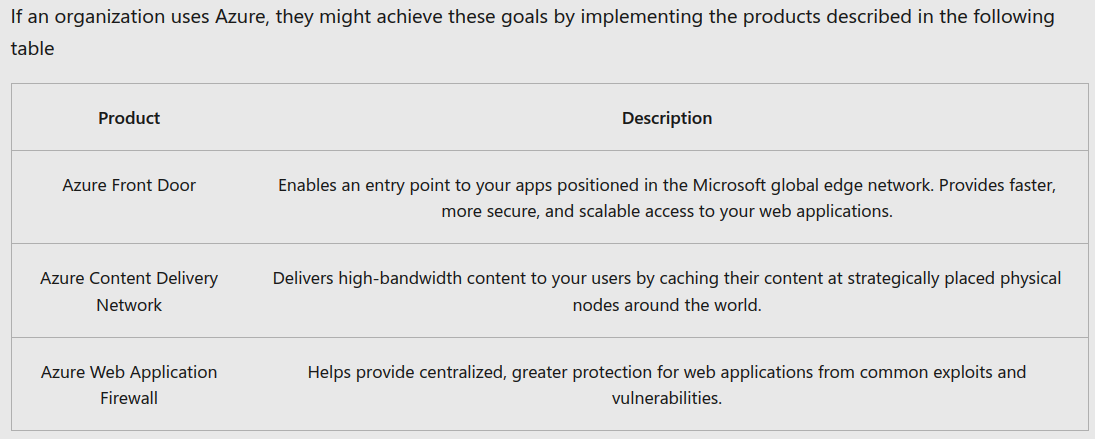
#### Request and response headers

* Headers allow client & server to pass additional info w/ a request or response
* Used to:
  + Add security-related header fields (HSTS/ X-XSS-Protection)
  + Remove response header fields that reveal sensitive info
  + Remove port information from X-Forwarded-For headers.
* 

#### URL path/query string

* Rewrite host name, path, and query string of the request URL
  + 

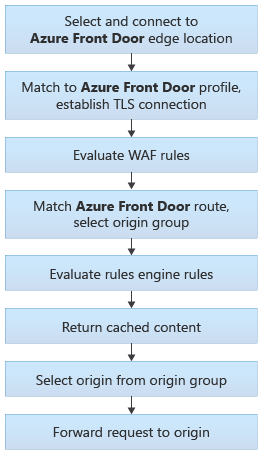
# Chapter 5: Design and configure Azure Front Door (FD)

* Azure FD allows access between users and your app’s static and dynamic web content across the globe.
  + It delivers your content using the MS global edge network:
  + 
  + A secure, modern cloud CDN provides a distributed platform of servers
  + 

## Azure Front Door tier comparison

* Standard vs. Premium
  + FD combines Azure FD (classic), Azure CDN Standard (classic), Azure WAF into 1 secure cloud CDN platform
  + Users connect to your app through MS global network & Azure FD routes user requests to the fastest and most available application back-end.

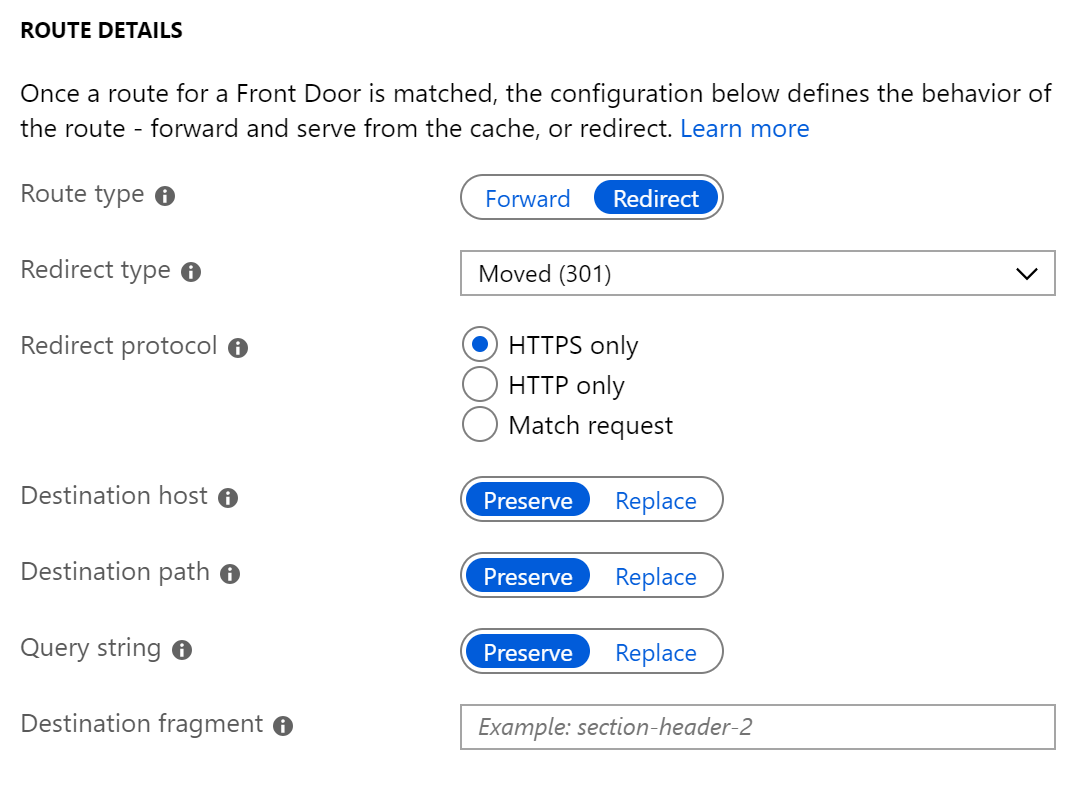
## Routing architecture overview

* 

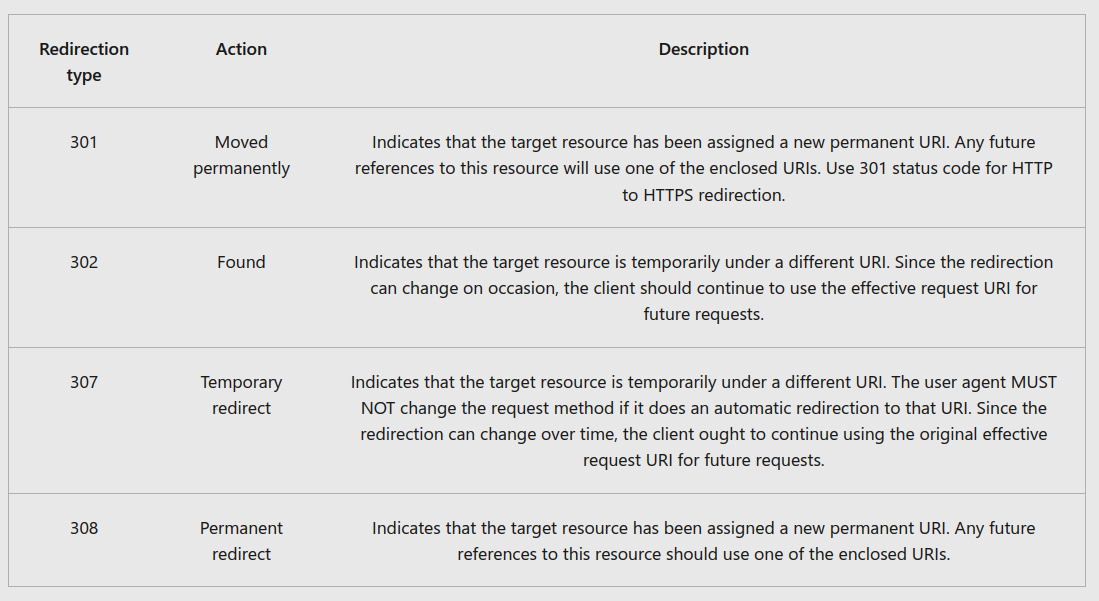
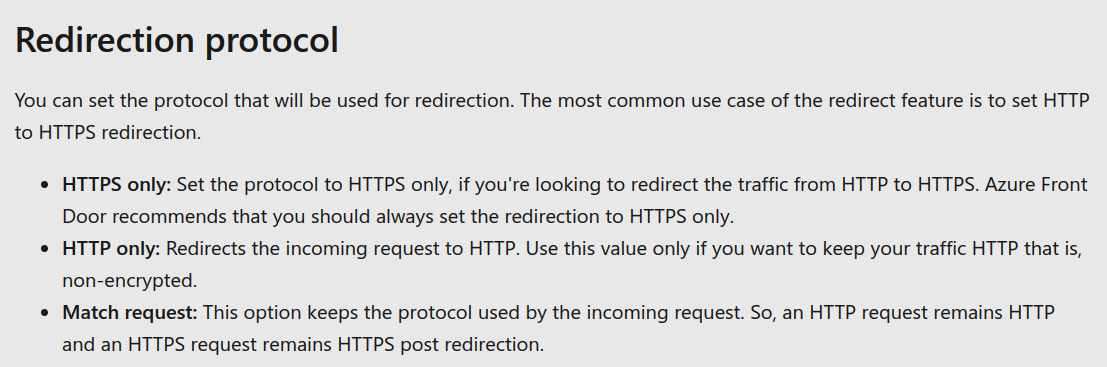
## Configure redirection rules in Front Door

* After establishing a connection & successful TLS handshake, FD will determine which routing rule to match the request

### Front Door route rules configuration structure

* 2 major parts:
  + "*left-hand side*" of route is matched by FD to the incoming request.
    - HTTP(s) Protocols
    - Hosts (**ex**. www.foo.com, \*.bar.com)
    - Paths (**ex**. /, /users/, /file.gif)
  + "*right-hand side*" of route defines how FD processes the request.
* **Route data**
  + Cached data for a specific route (gets cached response)
* **Route matching**
  + FD matches req to *most-specific match first* looking only at the left-hand side of the route.
  + Criteria: *HTTP* protocol, then *Front-end* host, then the *Path*.
* **Azure FD redirects traffic at each of the following levels: protocol, host-name, path, query string.**
  + 

### Redirection Types:

* 
* 

## Configure health probes, including customization of HTTP response codes

* Probes are used to determine health/proximity of each back-end pool for 1 Front Door environment
  + FD environment periodically sends HTTP/HTTPS request to configured back-ends and the response determines the ‘best’ back-end resource for request to go to.
* **Supported HTTP methods for health probes**
  + GET method means retrieve info.
  + HEAD method is identical to GET except that the server MUST NOT return a message-body
* **Responses**:
  + 