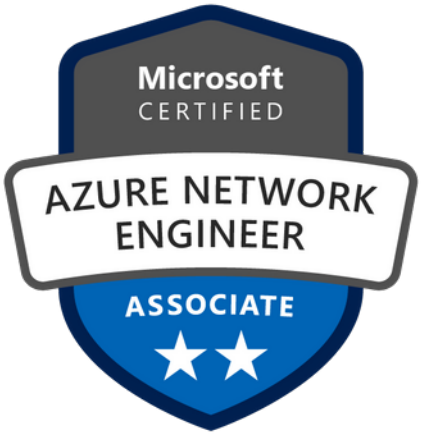

Azure Application Gateway and Azure WAF

Azure Application Gateway is essentially a load balancer for web traffic, but it also provides us with better traffic control. Traditional load balancers operate on the transport layer and allow us to route traffic based on protocol (TCP or UDP) and IP address, mapping IP addresses, and protocols in the frontend to IP addresses and protocols in the back end. This "classic" operation mode is often referred to as layer 4. Application gateway expands on that and allows us to use hostnames and paths to determine where traffic should go, making it a layer 7 load balancer. For example, we can have multiple servers that are optimized for different things. If one of our servers is optimized for video, then all video requests should be routed to that specific server based on the incoming URL request.

We will cover the following recipes in this chapter:

- Creating a new application gateway
- Configuring the backend pools
- Configuring HTTP settings
- Configuring listeners
- Configuring rules
- Configuring probes
- Configuring a **Web Application Firewall** (**WAF**)
- Customizing WAF rules
- Creating a WAF policy

Technical requirements

For this chapter, an Azure subscription is required.

Creating a new application gateway

Azure Application Gateway can be used as a simple load balancer to perform traffic distribution from the frontend to the backend based on protocols and ports. But it can also expand on that and perform additional routing based on hostnames and paths. This allows us to have resource pools based on rules and also allows us to optimize performance. Using these options and performing routing based on context will increase application performance, along with providing high availability. Of course, in this case, we need to have multiple resources for each performance type in each backend pool (each performance type requests a separate backend pool).

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to create a new application gateway, we must do the following:

1. In the Azure portal, select **Create a resource** and choose **Application Gateway** under **Networking** (or search for `application gateway` in the search bar).
2. In the new pane, we must provide information for **Subscription**, **Resource group**, **Name**, **Region**, **Tier**, **Autoscaling**, **Instance count**, **Availability zone**, and **HTTP2**. We must also select the **Virtual network** and **Subnet** that will be associated with our application gateway. You will be limited to virtual networks that are located in the region that is selected for the application gateway:

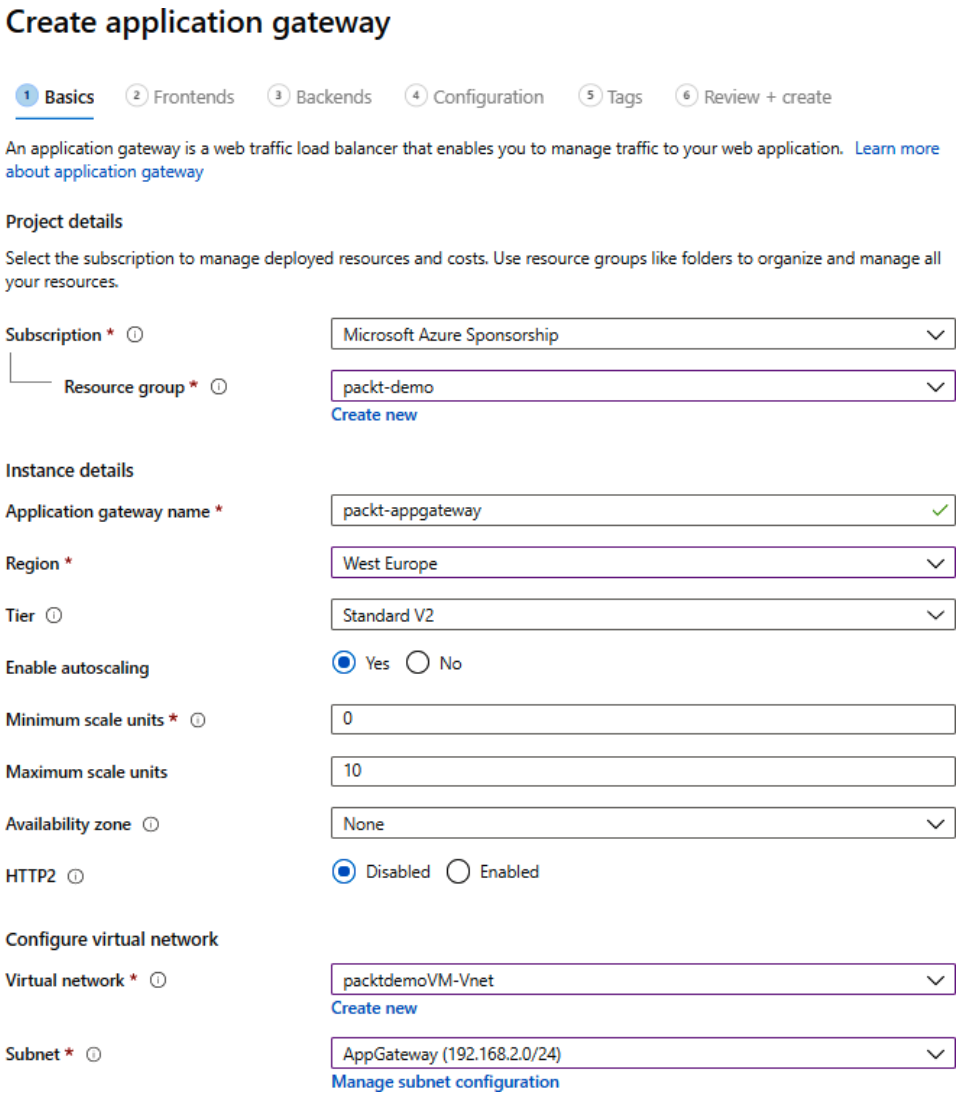

Figure 12.1: Configuring project details for application gateway

3. Now, we fill in the **Frontends** tab. Here, we need to select the type of IP address that the frontend will use (**Public**, **Private**, or **Both**) and provide an IP (select an existing one or create a new one):

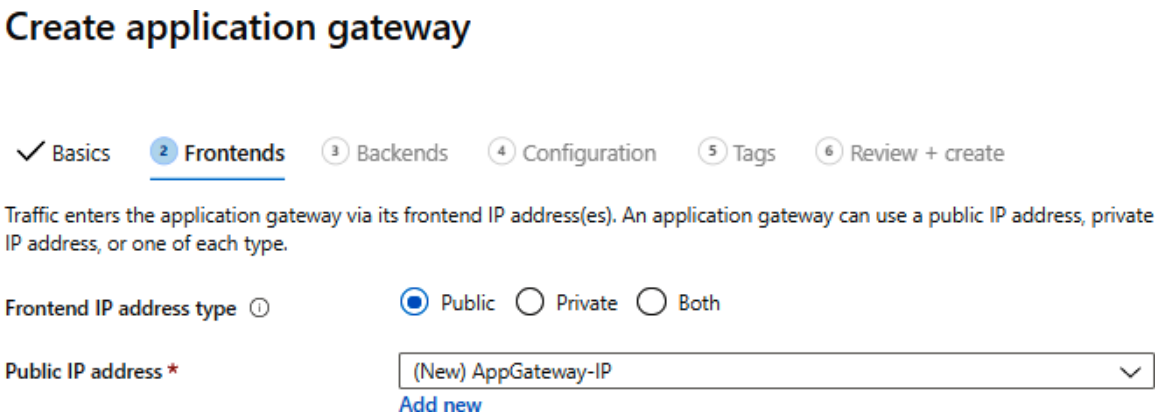

Figure 12.2: Selecting the Frontend IP address type

4. Next is the **Backends** tab. We need to select **Add a backend pool**:

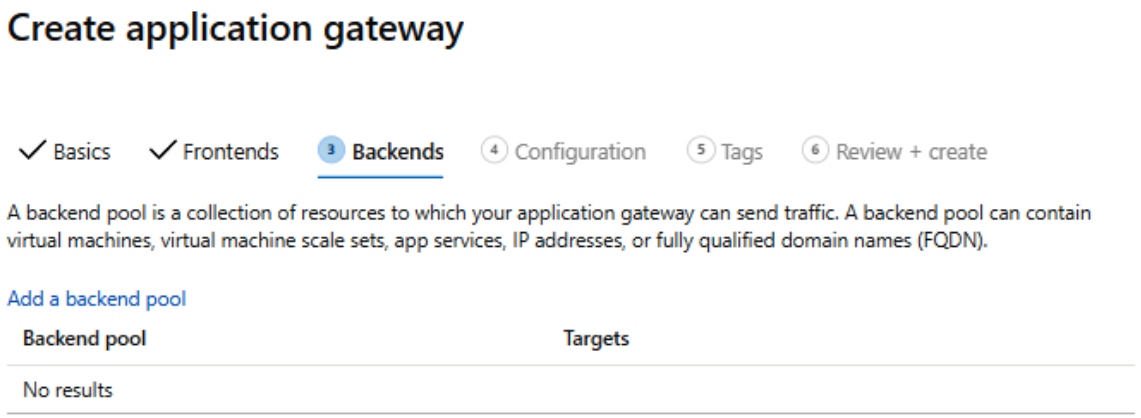

Figure 12.3: Defining backends for application gateway

5. At this point, a new pane will open. We need to provide information for **Name** and choose whether we want to add a backend pool with or without targets. If we choose to add targets at this stage, first, we need to select **Target type**. The available types are virtual machines, virtual machine scale sets, app services, and IP addresses/FQDNs. Based on the type selection, you can add appropriate targets:

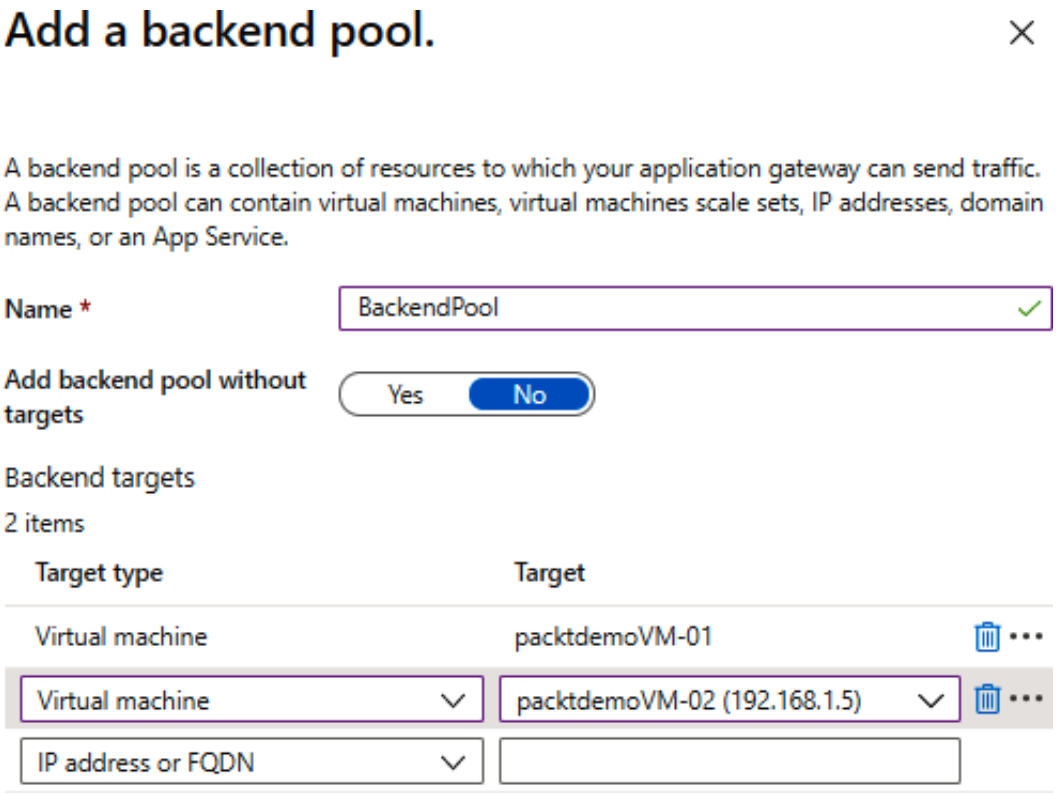

Figure 12.4: Adding a backend pool

6. After we have added a backend pool, we can see related information and proceed. Note that we can add more than one backend pool:

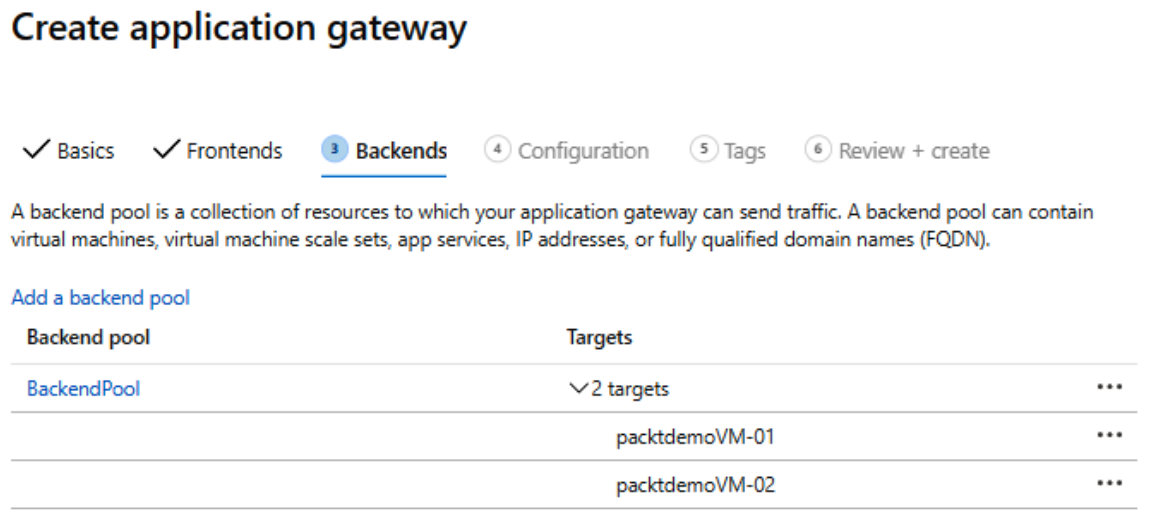

Figure 12.5: Reviewing the configuration for the backend pool

7. In the **Configuration** pane, we can see that the frontends and backends pools are in place, but we are missing a routing rule. This is mandatory in order to proceed, so we must create one by selecting **Add a routing rule**:

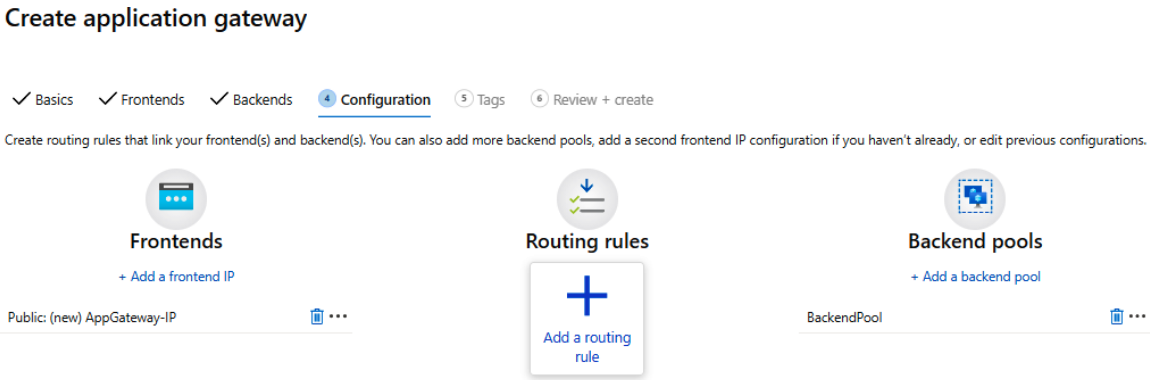

Figure 12.6: Creating a routing rule

8. In the new pane, we must first define a listener. For the listener, we must provide a name, select the **Frontend IP** configuration, and provide a **Port** and **Protocol** that will be monitored. We can also change the **Listener type** radio button and add a redirect URL page for errors (this can only be an Azure storage account URL):

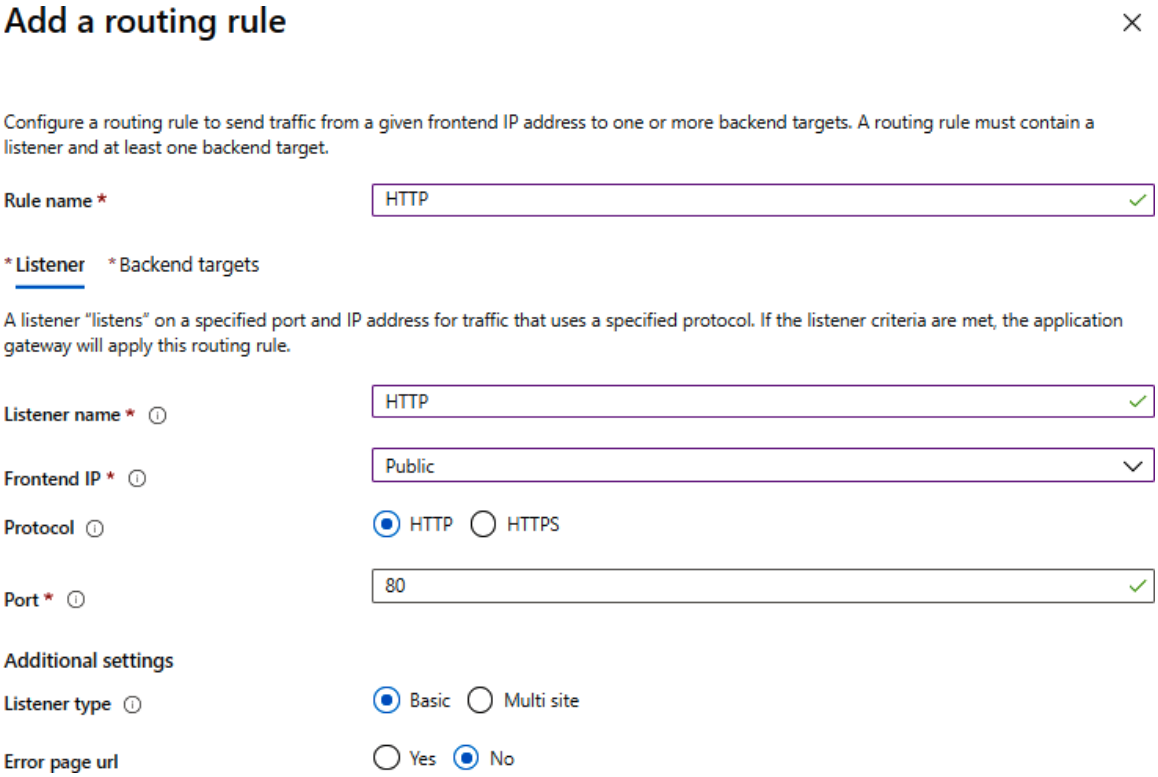

Figure 12.7: Configuring the listener settings for the routing rule

9. For the routing rule, we need to configure **Backend targets** as well. In this section, we need to set **Target type**, **Backend target**, and **HTTP settings**. At this stage, we are still lacking an HTTP setting, so we need to select **Add new** under the **HTTP settings** field:

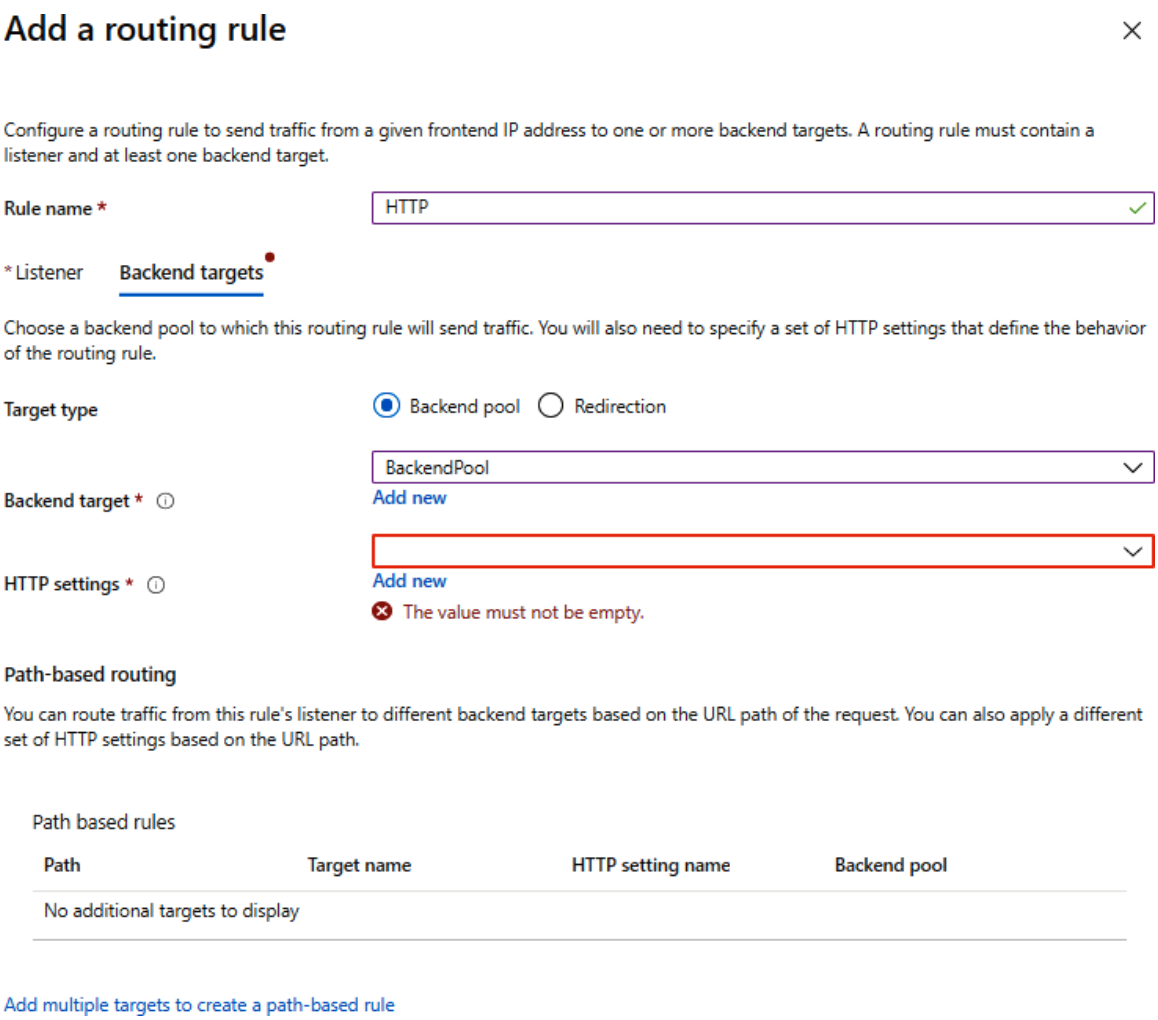

Figure 12.8: Configuring backend targets for the routing rule

10. In the new pane, first, we need to provide our HTTP setting with a name and add details for **Backend protocol** and **Backend port**. We must also enable or disable **Cookie-based affinity** and **Connection draining** before specifying the **Request time-out (seconds)** period. We can enable or disable the **Create custom probes** and **Override with new host name** settings:

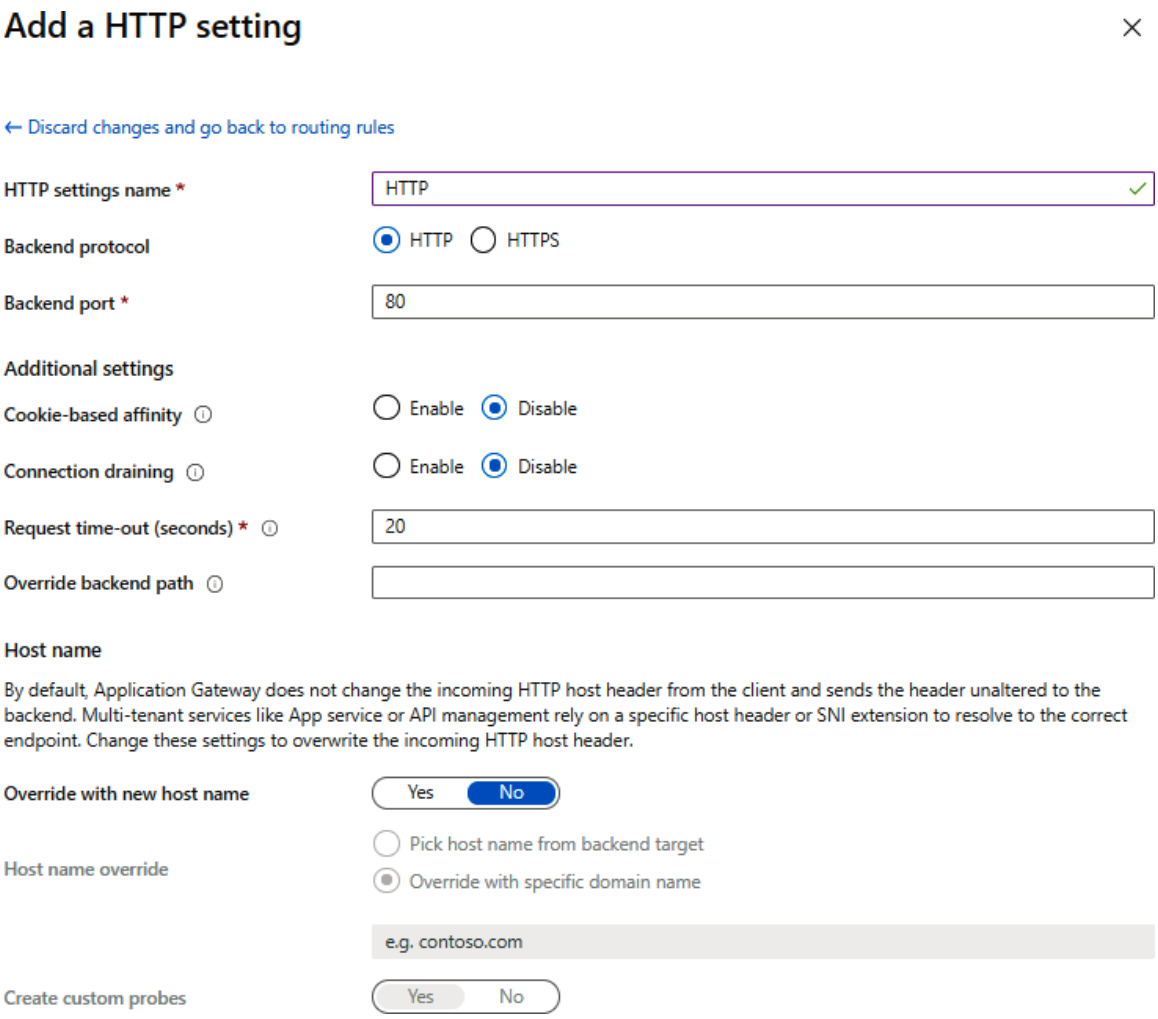

Figure 12.9: Adding an HTTP setting

11. After the HTTP setting is created, it will be automatically added to our routing rule, which we can now finish:

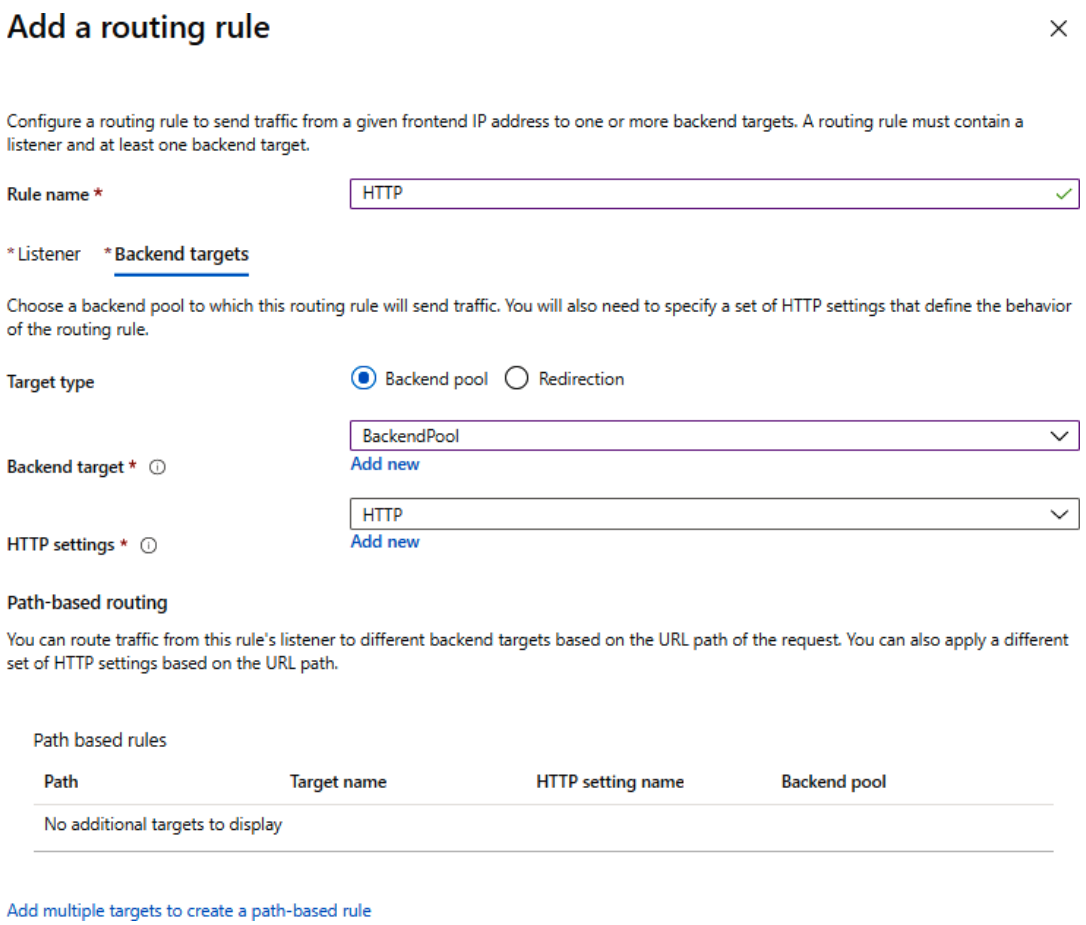

Figure 12.10: Final configuration for adding a routing rule

12. The configuration is now complete, and we can go ahead and deploy our application gateway:

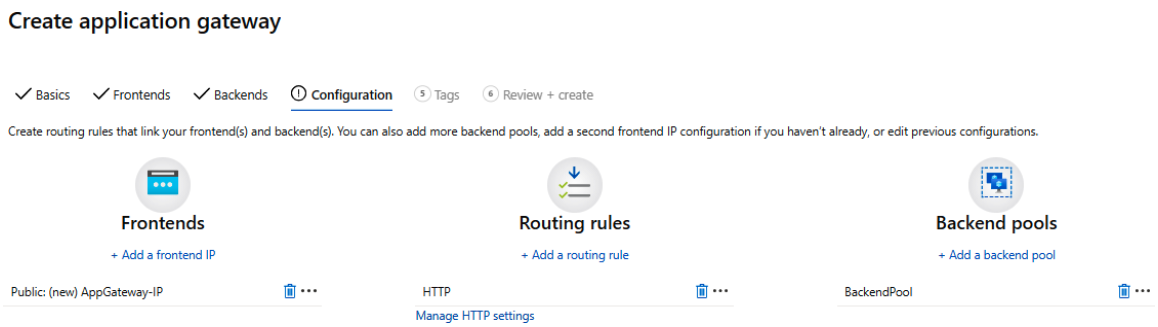

Figure 12.11: Deploying our application gateway

How it works...

Azure Application Gateway is very similar to Azure Load Balancer, with some additional options. It will route traffic coming to the front end of the application gateway to a defined backend based on rules that we define. In addition to routing based on protocols and ports, the application gateway also allows defined routing based on paths and protocols. Using these additional rules, we can route incoming requests to endpoints that are optimized for certain roles. For example, we can have multiple backend pools with different settings that are optimized to perform only specific tasks. Based on the nature of the incoming requests, the application gateway will route the requests to the appropriate backend pool. This approach, along with high availability, will provide better performance by routing each request to a backend pool that will process the request in a more optimized way.

We can set up autoscaling for application gateway (available only for V2) with additional information for the minimum and maximum number of units. This way, application gateway will scale based on demand and ensure that performance is not impacted, even with the maximum number of requests.

Configuring the backend pools

After the application gateway is created, we must define the backend pools. Traffic coming to the front end of the application gateway will be forwarded to the backend pools. Backend pools in application gateways are the same as backend pools in load balancers and are defined as possible destinations where traffic will be routed based on other settings that will be added in future recipes in this chapter.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to add backend pools to our application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** pane, under **Settings**, select **Backend pools**. Select **Add** to add a new backend pool or select an existing one to edit:

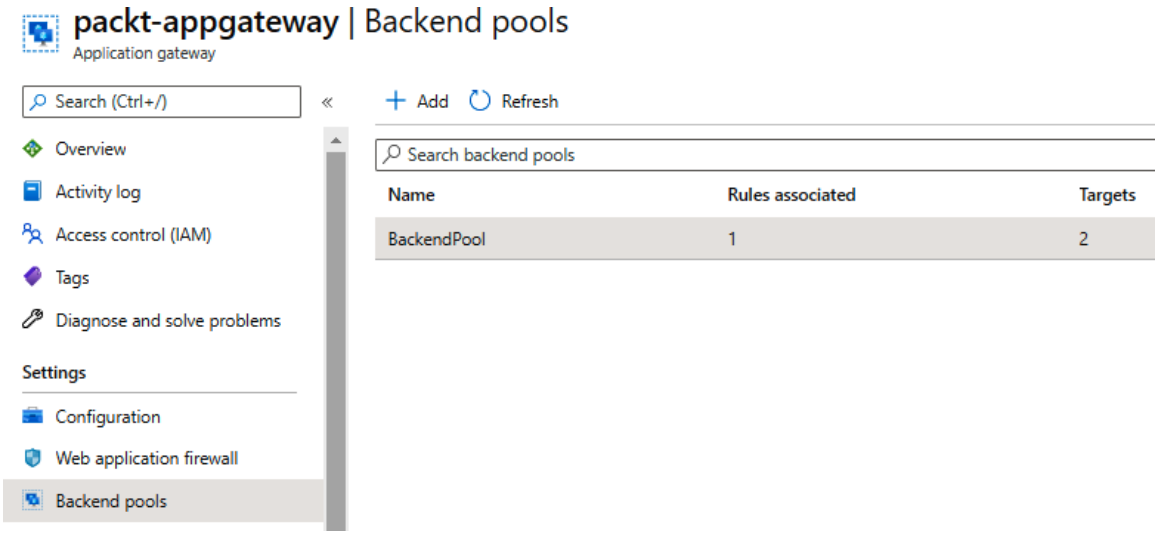

Figure 12.12: Adding a backend pool to our application gateway

3. In the new pane, the only difference between new and existing pools is the name. For a new pool, we must provide the name of the backend pool, and for existing pools, this option is grayed out and cannot be edited. For both new and existing pools, we must provide the type of target. The available types are virtual machines, virtual machine scale sets, app services, and IP addresses/FQDNs. Based on the type selection, you can add appropriate targets:

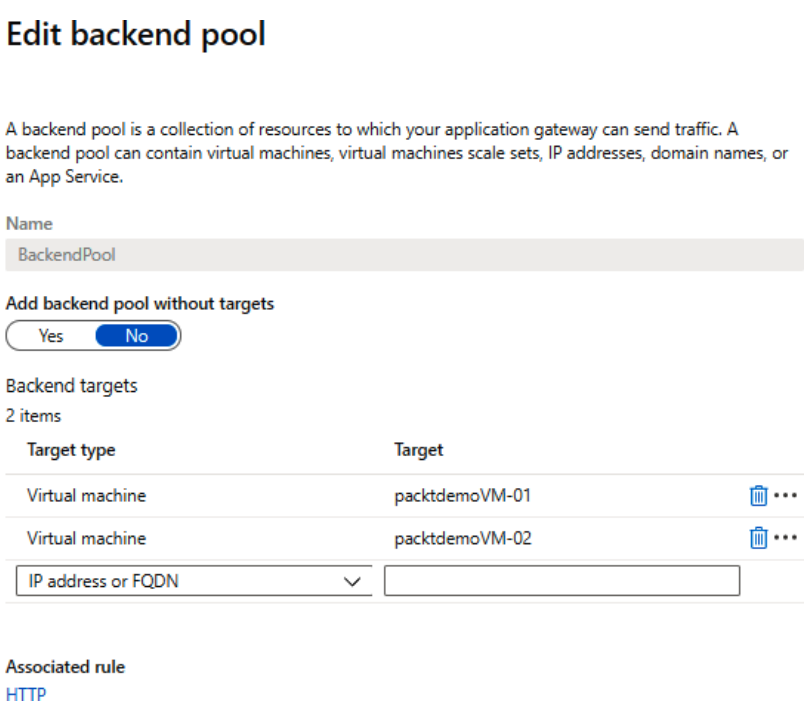

Figure 12.13: Providing the target type for the backend pool

How it works...

With backend pools, we define targets to which traffic will be forwarded. As the application gateway allows us to define routing for each request, it's best to have targets based on performance and types grouped in the same way. For example, if we have multiple web servers, these should be placed in the same backend pool. Servers used for data processing should be placed in a separate pool, and servers used for video in another separate pool. This way, we can separate pools based on performance types, and route traffic based on operations that need to be completed.

This will increase the performance of our application, as each request will be processed by the resource best suited for a specific task. To achieve high availability, we should add more servers to each backend pool.

Configuring HTTP settings

HTTP settings in application gateways are used for validation and various traffic settings. Their main purpose is to ensure that requests are directed to the appropriate backend pool. Some other HTTP settings are also included, such as affinity or connection draining. Override settings are also part of HTTP settings—these will allow you to redirect if an incomplete or incorrect request is sent.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to add HTTP settings to our application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** pane, under **Settings**, select **HTTP settings**. Select **Add** to add a new HTTP setting or select an existing one to edit:

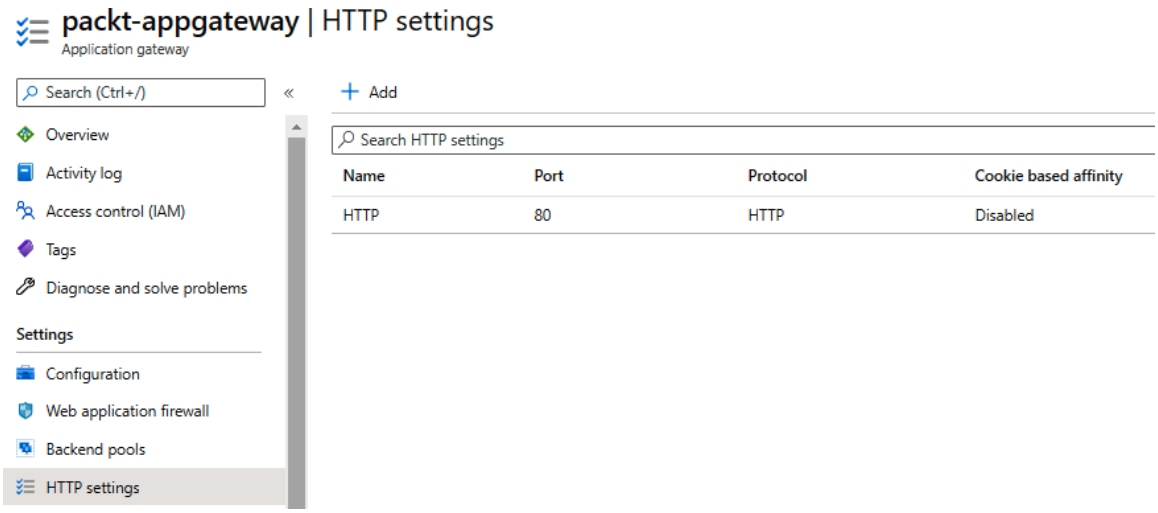

Figure 12.14: Locating HTTP settings in the Application gateway pane

3. In the new pane, first, we need to provide a name (if you are editing an existing HTTP setting, this option will be grayed out). The next options allow us to disable or enable **Cookie-based affinity** and **Connection draining**. Further to this, we select our **Protocol**, **Port**, and the **Request time-out (seconds)** period. Optional settings allow us to configure **Use custom probe** and **Override with new host name**:

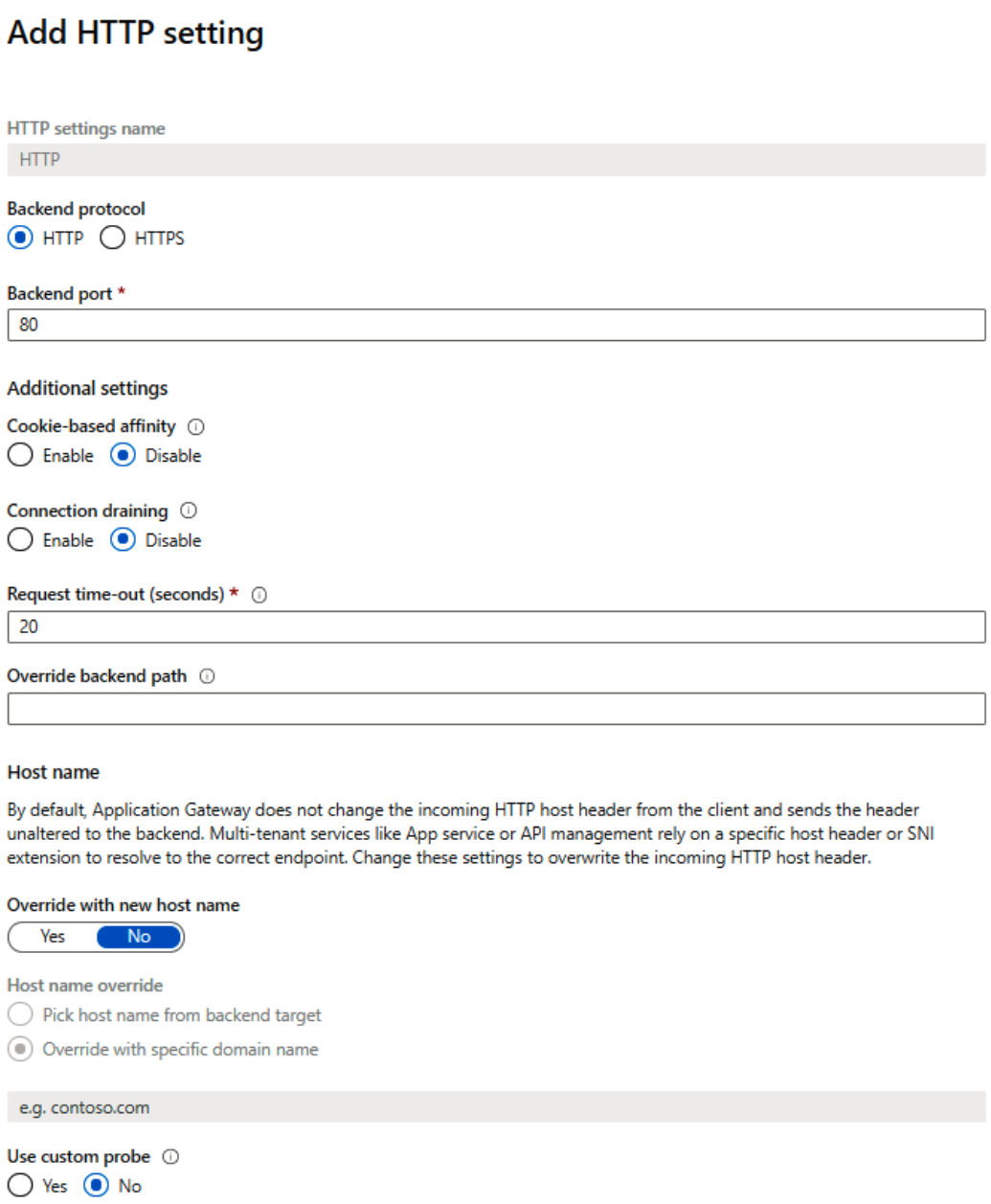

Figure 12.15: Configuring HTTP settings

How it works...

As previously mentioned, the main purpose of HTTP settings is to ensure that requests are directed to the correct backend pool. However, various other options are available. Cookie-based affinity allows us to route requests from the same source to the same target server in the backend pool. Connection draining will control the behavior when the server is removed from the backend pool. If this is enabled, the server will help maintain in-flight requests to the same server. Override settings allow us to override the path of the URL to a different path or a completely new domain, before forwarding the request to the backend pool.

Configuring listeners

Listeners in an application gateway listen for any incoming requests. After a new request is detected, it's forwarded to the backend pool based on the rules and settings we have defined. In this recipe, we will add a new listener to our application gateway.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to add a listener to an application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** pane, under **Settings**, select **Listeners**, then select **Add listener** to add a new listener, or edit an existing one:

Figure 12.16: Adding a new listener through the Azure portal

3. In the new pane, we need to provide a name for the listener (if you are editing an existing listener, this option will be grayed out), select the **Frontend IP** configuration, and provide the **Port** and **Protocol** that will be monitored. Additionally, we can set up the **Listener type** and a custom URL page for errors:

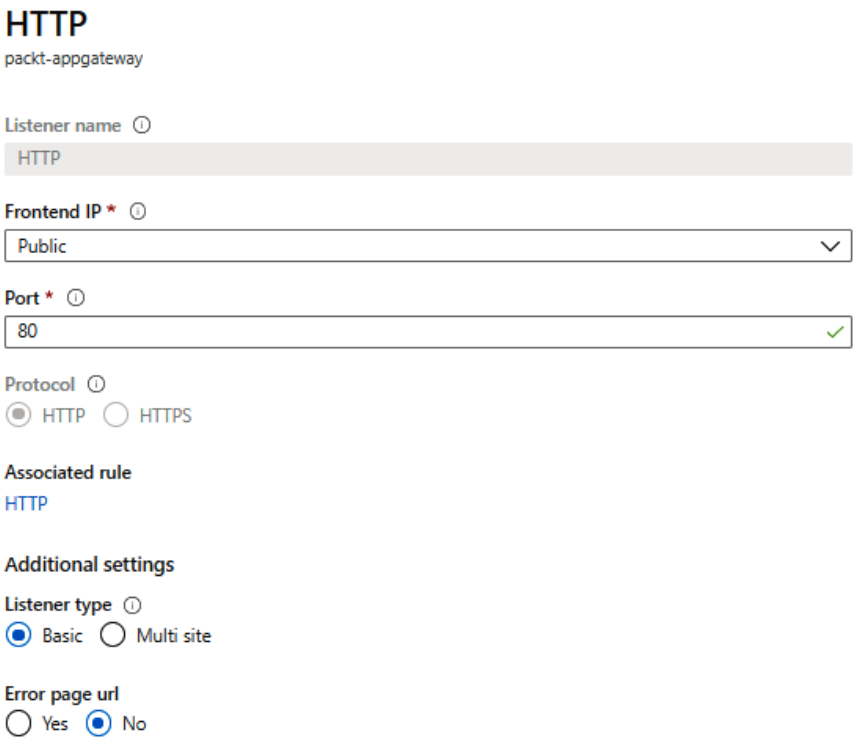

Figure 12.17: Configuring the listener settings for our application gateway

How it works...

A listener monitors for new requests coming to the application gateway. Each listener monitors only one frontend IP address and only one port. If we have two frontend IPs (one public and one private) and traffic coming in over multiple protocols and ports, we must create a listener for each IP address and each port that traffic may be coming to.

The basic type of listener is used when the listener listens to a single domain; it's usually used when we host a single application behind an application gateway. A multi-site listener is used when we have more than one application behind the application gateway and we need to configure routing based on a host name or domain name.

Configuring rules

Rules in application gateways are used to determine how traffic flows. Different settings determine where a specific request is forwarded to and how this is done.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to add a rule to the application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** pane, under **Settings**, select **Rules**. Add a new rule or select an existing one to edit:

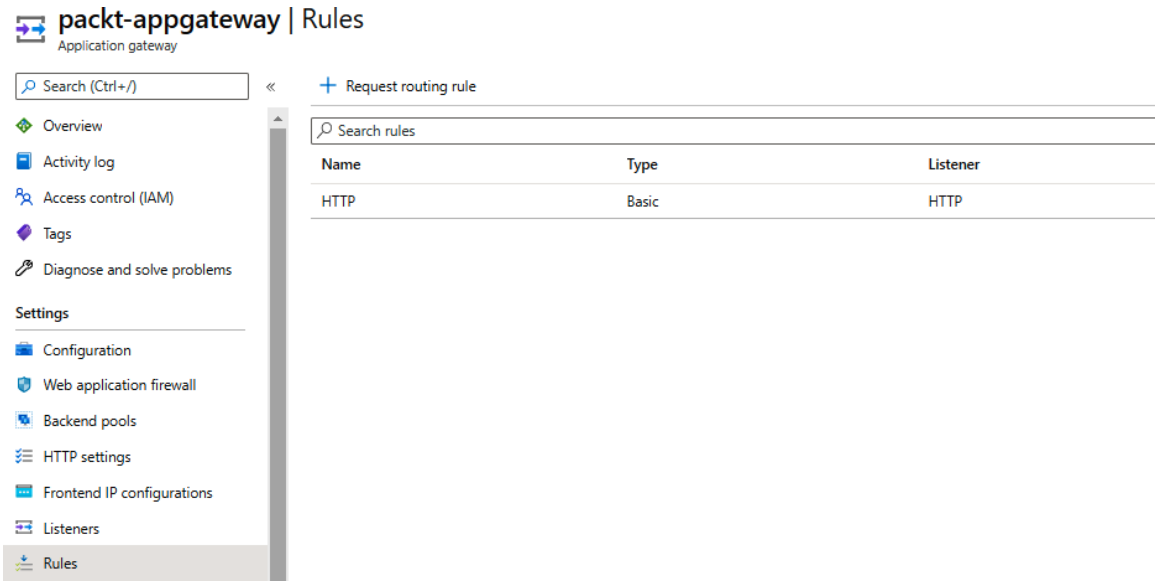

Figure 12.18: Adding a routing rule for our application gateway

3. In the new pane, we must provide a name for the new rule (if you are editing an existing rule, this option is grayed out) and select the **Listener**, as shown in *Figure 12.19*:

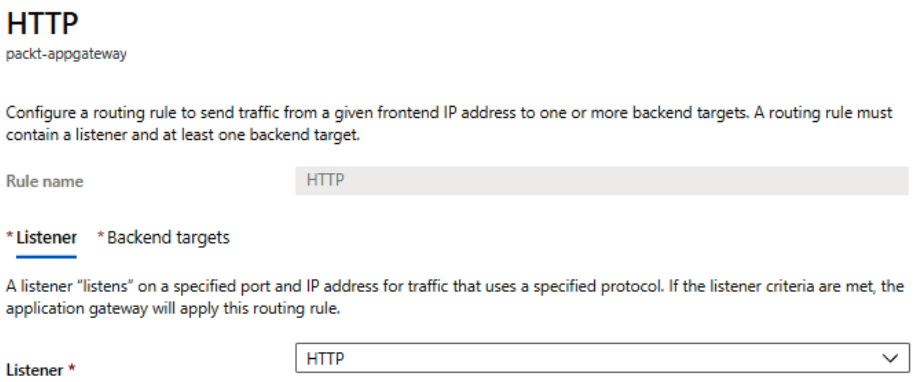

Figure 12.19: Configuring the routing rule

4. We also need to set up a backend target, where we need to define **Target type** and select options for **Backend target** and **HTTP settings**:

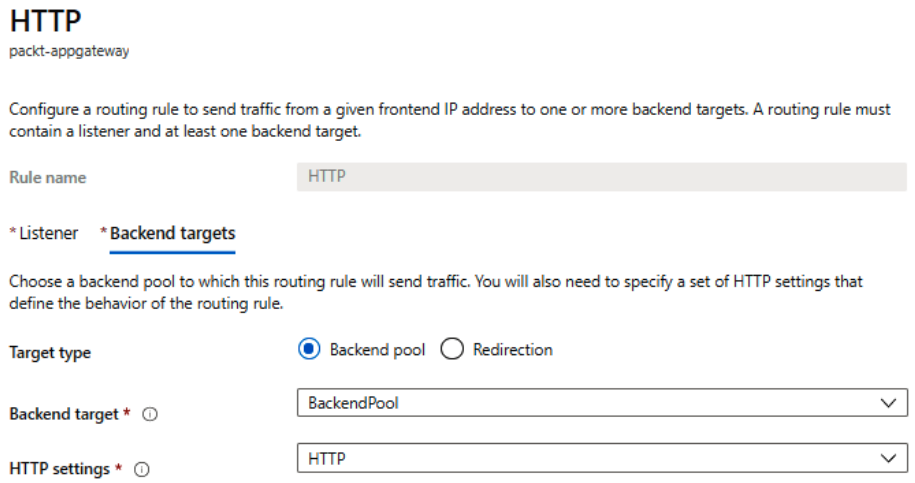

Figure 12.20: Setting up a backend target for our routing rule

How it works...

Using rules, we can tie some previously created settings together. We define a listener that specifies which request on what IP address we are expecting on which port. Then, these requests are forwarded to the backend pool; forwarding is performed based on the HTTP settings. Optionally, we can also add redirection to the rules.

Configuring probes

Probes in application gateway are used to monitor the health of the backend targets. Each endpoint is monitored, and if one is found to be unhealthy, it is temporarily taken out of rotation and requests are not forwarded. Once the status changes, it's added back. This prevents requests from being sent to unhealthy endpoints that can't serve the request.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to add a probe to our application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** pane, under **Settings**, select **Health probes**. Select **Add** to add the new probe:

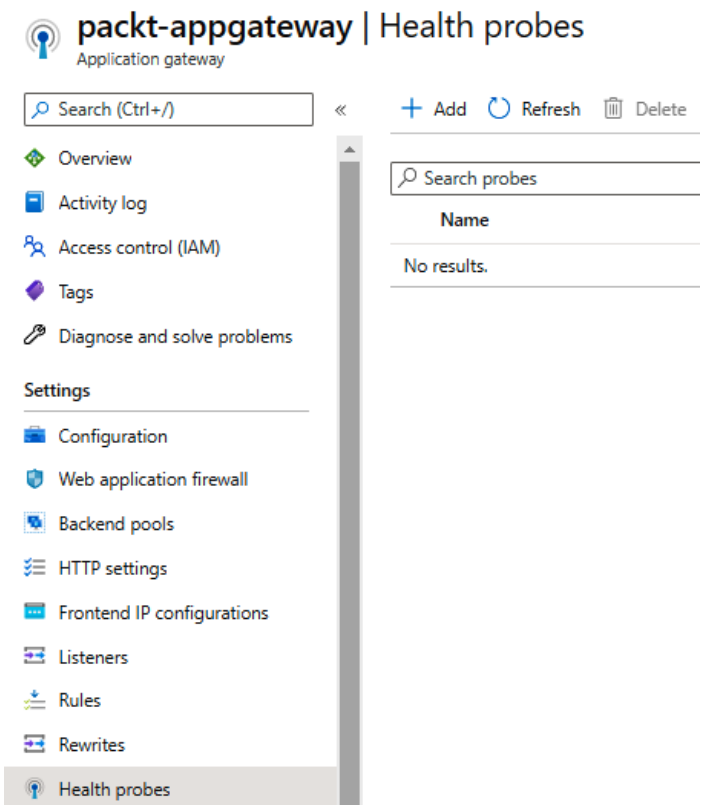

Figure 12.21: Adding a new health probe

3. In the new pane, we must provide the **Name** of the probe (this option will be grayed out if an existing probe is edited), along with the **Protocol**, **Host**, and **Path**. We also need to set the **Interval (seconds)**, **Timeout (seconds)**, and **Unhealthy threshold** settings. We can also choose to configure **Use probe matching conditions** and associate **HTTP settings**:

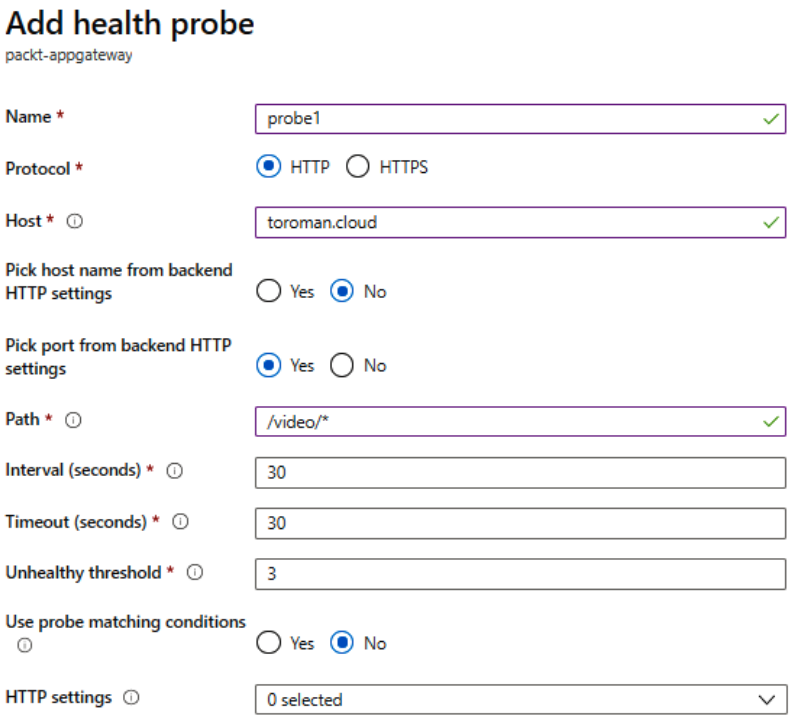

Figure 12.22: Configuring the health probe details

How it works...

Protocol, **Host**, and **Path** define what probe is being monitored. **Interval** defines how often checks are performed. **Timeout** defines how much time must pass before the check is declared to have failed. Finally, **Unhealthy threshold** is used to set how many failed checks must occur before the endpoint is declared unavailable.

Configuring a Web Application Firewall (WAF)

WAF is an additional setting for the application gateway. It's used to increase the security of applications behind the application gateway, and it also provides centralized protection.

Getting ready

To enable a WAF, we must set the application gateway to the WAF tier. To do so, we must do the following:

1. In the **Application gateway** pane, go to **Web application firewall**, under **Settings**. Change the **Tier** selection from **Standard V2** to **WAF V2** and select **Save**:

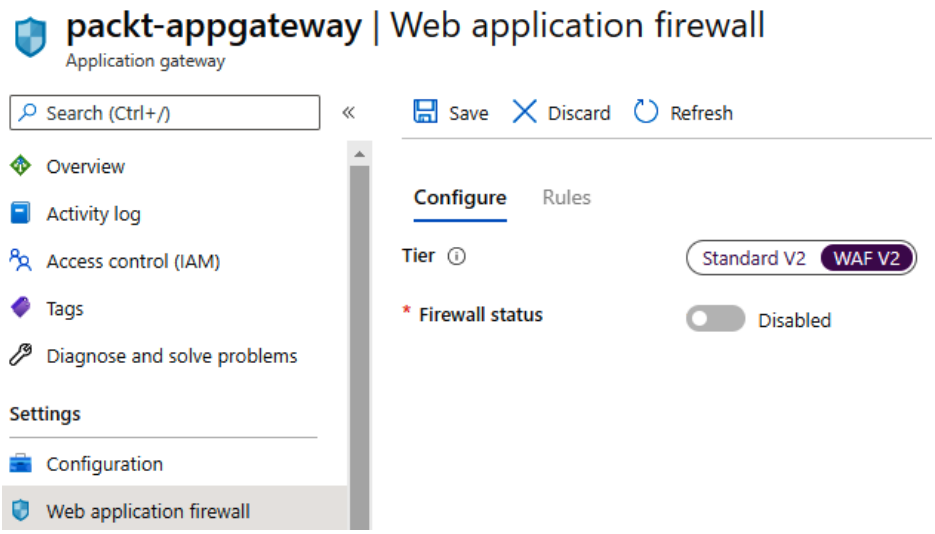

Figure 12.23: Setting the application gateway to the WAF V2 tier

How to do it...

After the application gateway is set to WAF, we can enable and set the firewall rules. To do so, we must do the following:

1. In the **Application gateway** pane, go to **Web application firewall**, under **Settings**, and enable **Firewall status**. After we set **Firewall status** to **Enabled**, a new set of options will appear:

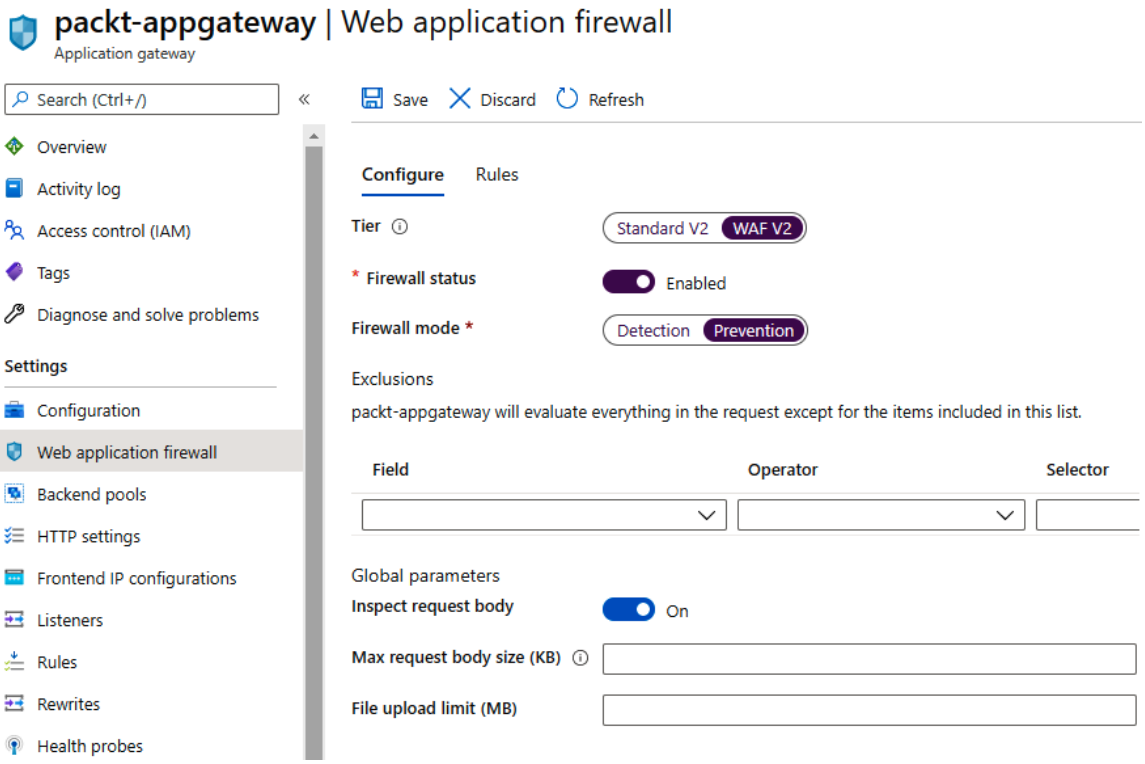

Figure 12.24: Enabling a WAF for our application gateway

2. We must select a **Firewall mode**, set an exclusion list, and specify the **Global parameters** as follows:

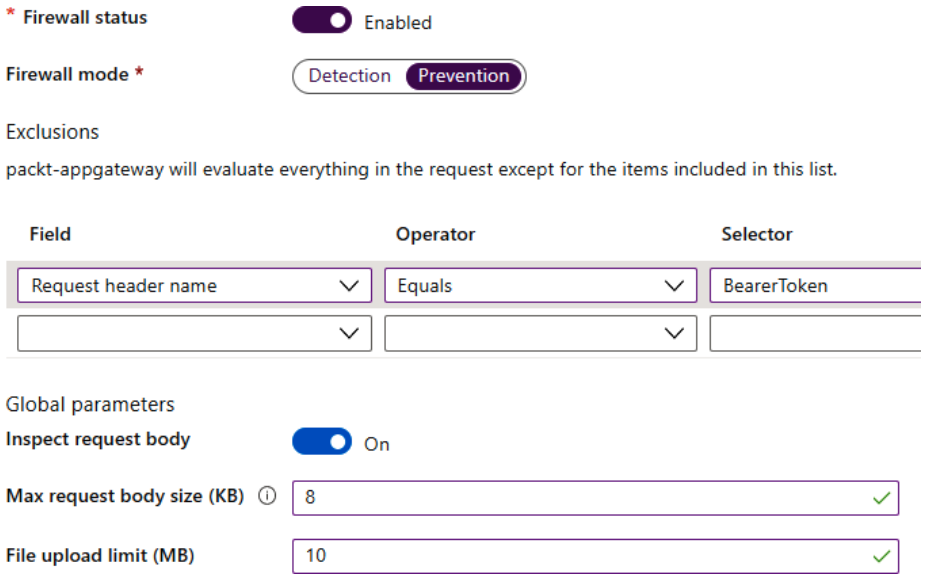

Figure 12.25: Configuring the WAF

How it works...

The WAF feature helps increase security by checking all incoming traffic. As this can slow down performance, we can exclude some items that are creating false positives, especially when it comes to items of significant size. Excluded items will not be inspected. A WAF can work in two modes: detection and prevention. Detection will only detect if a malicious request is sent, while prevention will stop any such request.

Customizing WAF rules

A WAF comes with a predetermined set of rules. These rules are enforced to increase application security and prevent malicious requests. We can change these rules to address specific issues or requirements as needed.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to change the WAF rules, we must do the following:

1. Select **Web application firewall** under **Settings** in the **Application gateway** pane.

2. Select **Rules** in the WAF settings. Select **Enabled** under **Advanced rule configuration**, as shown in *Figure 12.26*:

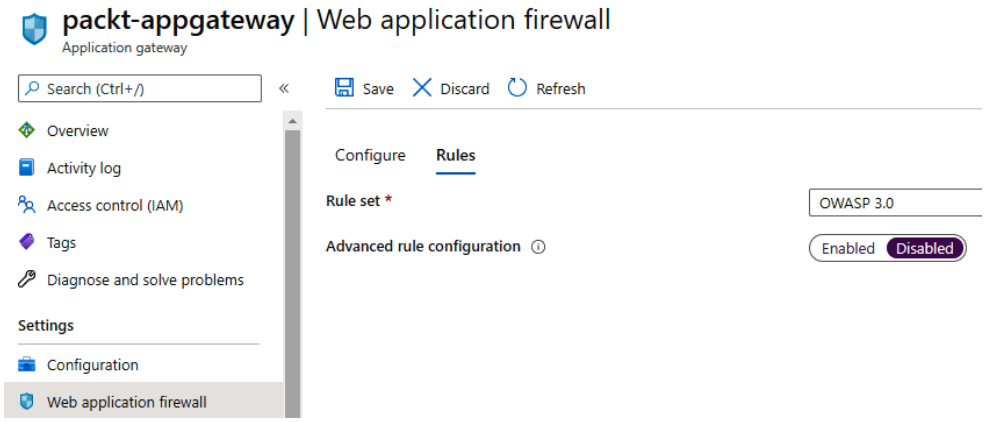

Figure 12.26: Enabling Advanced rule configuration

3. The rules will appear in the form of a list. We can check or uncheck boxes to enable or disable rules:

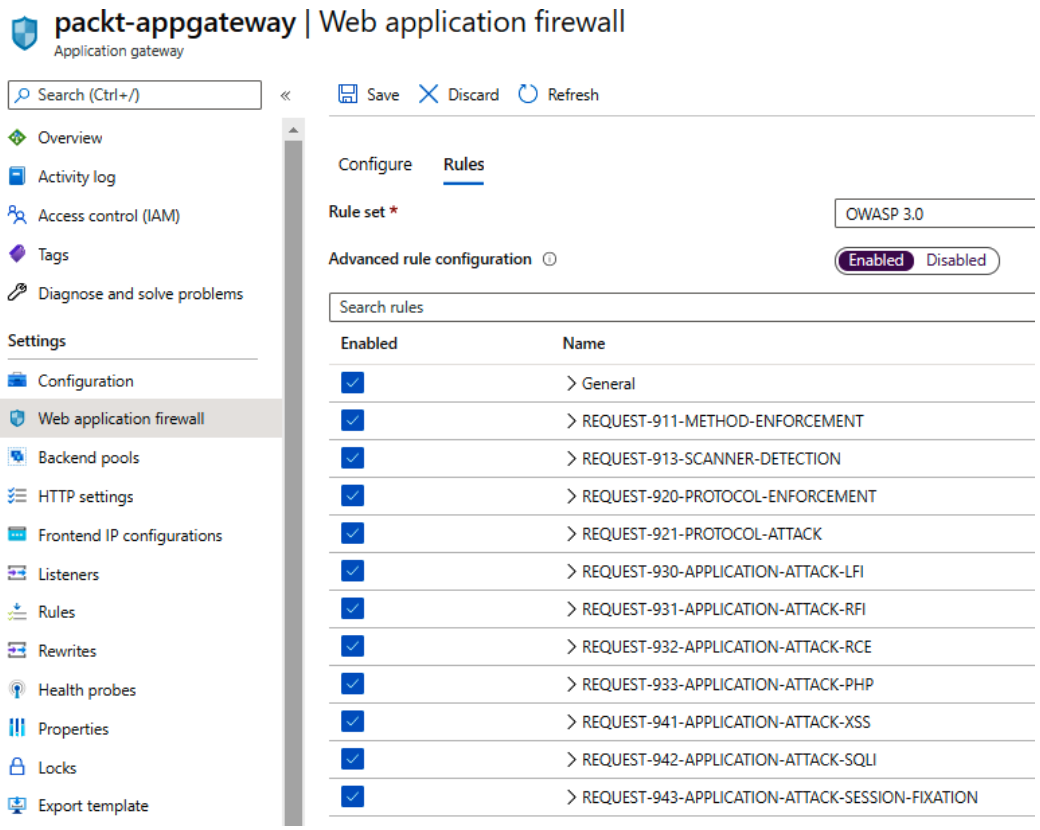

Figure 12.27: Customizing WAF rules in the Application gateway pane

How it works...

A WAF comes with all rules activated by default. This can slow down performance, so we can disable some of the rules if needed. Also, there are three rule sets available—**OWASP 2.2.9**, **OWASP 3.0**, and **OWASP 3.1**. The default (and recommended) rule set is **OWASP 3.0**, but we can switch between rule sets as required.

Creating a WAF policy

A WAF policy allows us to handle WAF settings and configurations as a separate resource. By doing so, we can apply the same policy to multiple resources instead of individual application gateways. A WAF policy can be associated with Application Gateway, Front Door, or CDN.

Getting ready

Before you start, open the browser and go to the Azure portal at https://portal.azure.com.

How to do it...

In order to create a new application gateway, we must do the following:

1. In the Azure portal, select **Create a resource** and choose **Web Application Firewall** under **Networking** (or search for `Web Application Firewall` in the search bar).
2. In the new pane, we must complete the **Basics** section first. We need to set what the policy is going to be used for (Application Gateway, Front Door, or CDN), configure **Subscription** and **Resource group**, and fill in the **Policy name** and **Location** fields. Additionally, we can set whether the policy will be enabled or disabled once it's created:

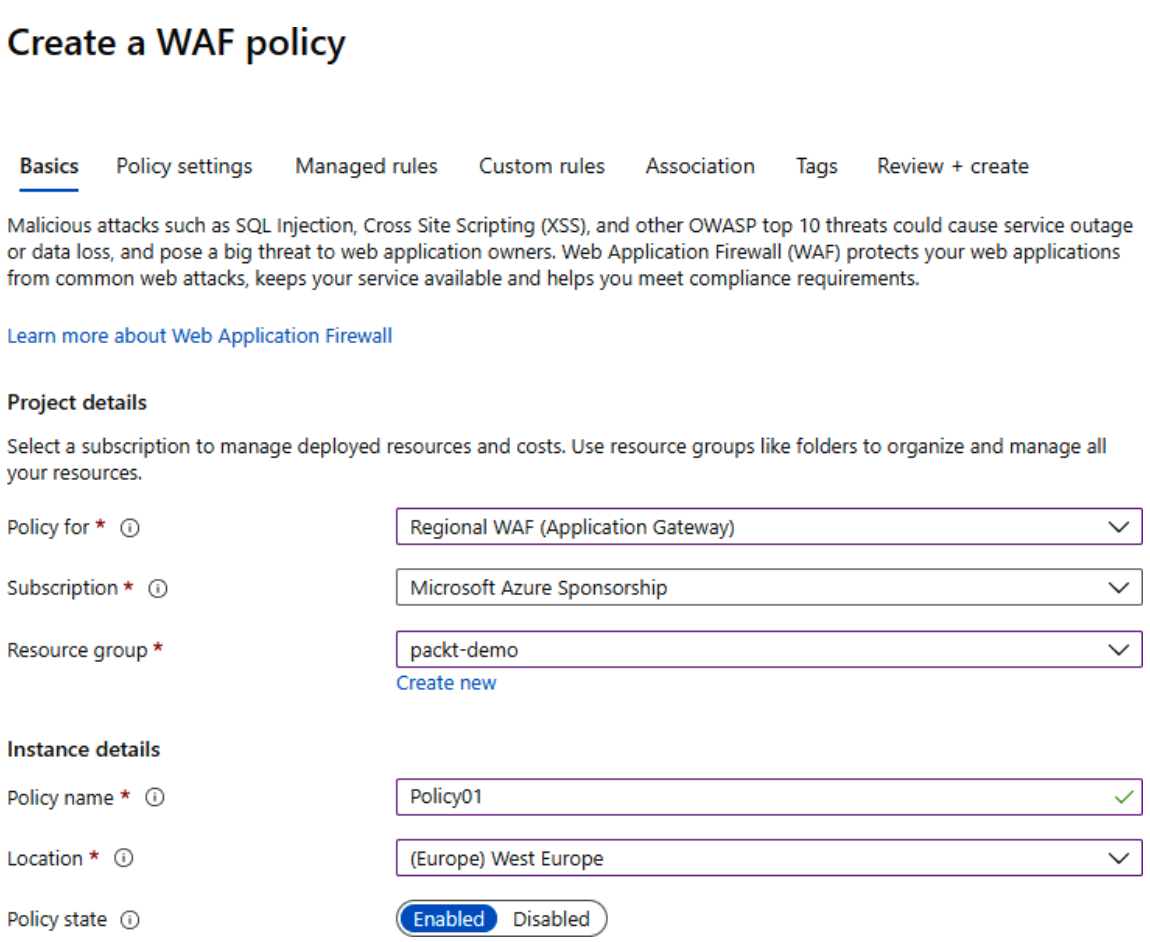

Figure 12.28: Creating a new WAF policy

3. In **Policy settings**, we can set **Mode** to **Detection** or **Prevention**, along with **Exclusions** and **Global parameters**:

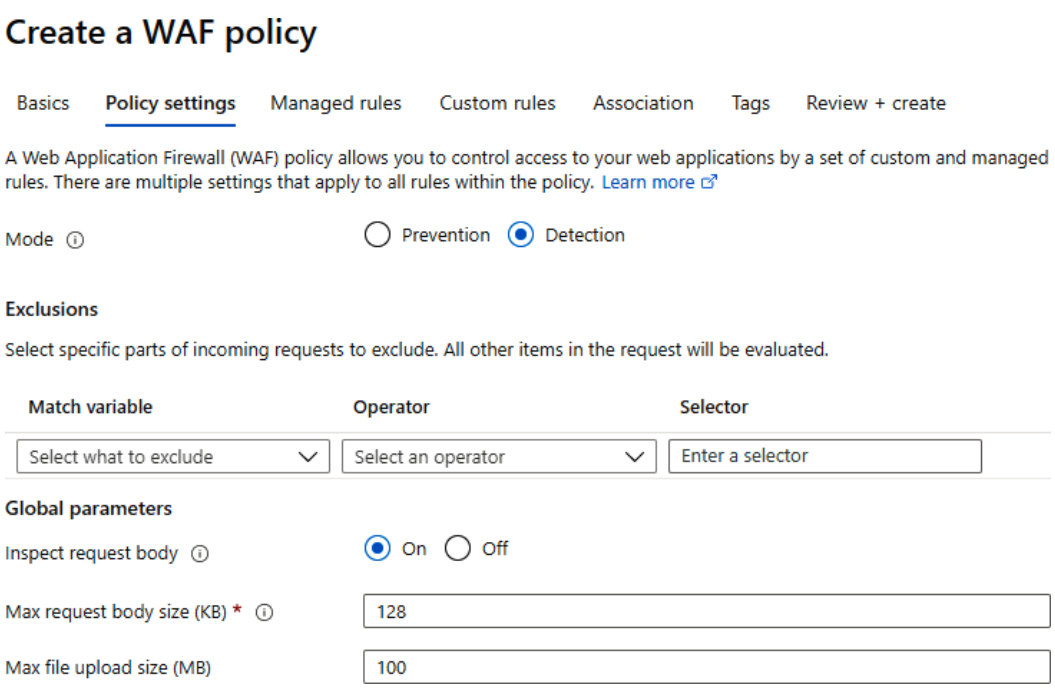

Figure 12.29: Configuring policy settings for your WAF policy

4. Under **Managed rules**, we can select a rule set (**OWASP 2.2.9**, **OWASP 3.0**, or **OWASP 3.1**) and disable some rules if needed (it is not recommended to disable rules unless necessary):

Figure 12.30: Setting rules for your WAF policy

5. Under **Custom rules**, we can add additional rules if needed. Select **Add custom rule** to add one:

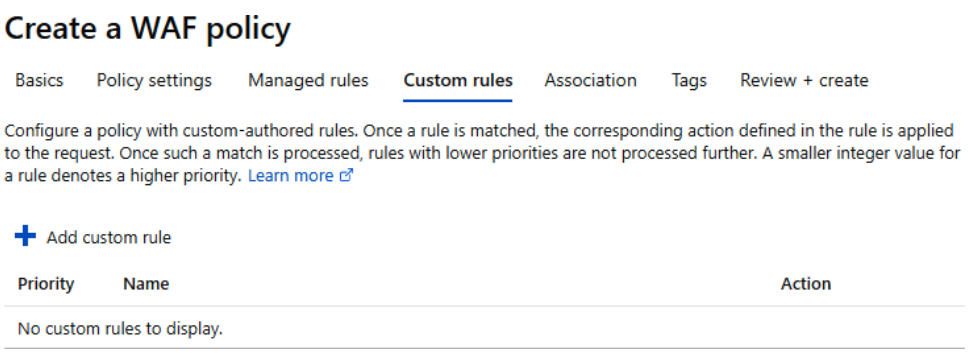

Figure 12.31: Adding a custom rule to our WAF policy

6. This will open a new pane that will allow us to define a custom rule. We need to fill in the **Custom rule name** field and set **Priority** to **1**. Under **Conditions**, we are creating a match type and variables that need to be matched in order to trigger the rule. Finally, we set a response (allow, deny, or log):

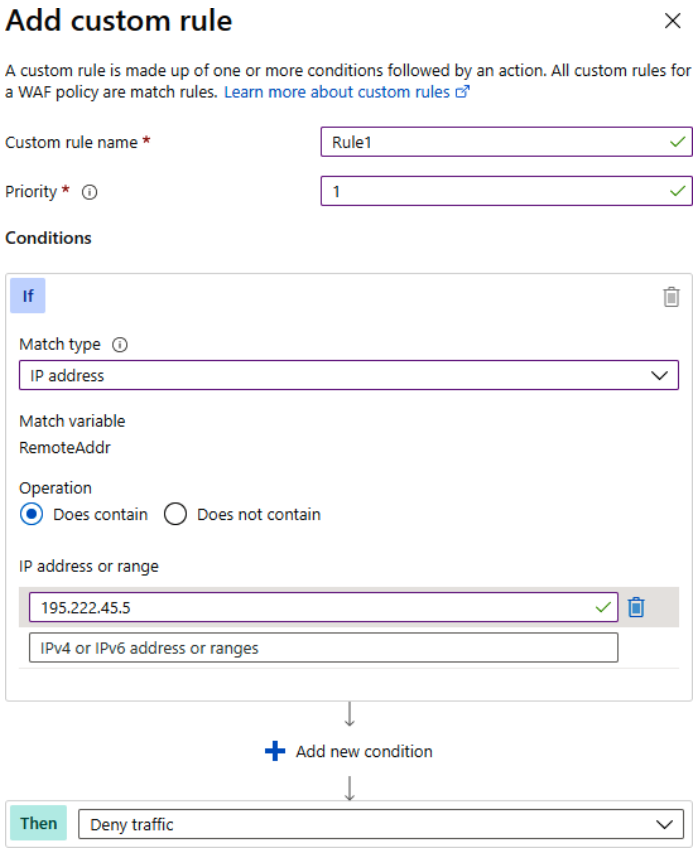

Figure 12.32: Defining conditions for your custom rule