# HTTP redirect based on Client Region

## Use Case:

While doing Redirects, many times you need to change the location or URL based on who the Client is and where he is coming from. In this example we are looking at which location or region Client belongs to and accordingly modify the URL.

## F5 iRules:

```
if { ([matchclass [IP::client_addr] eq $::region1clients]) {
   if { [active_members pool_region1servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region1"
   } else if { [active_members  pool_region2servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region2"
   } else if { [active_members pool_region3servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region3"
   } else if { [active_members pool_region4servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region4"
       }
} else if { ([matchclass [IP::client_addr] eq $::region2clients]) {
   if { [active_members pool_region2servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region2"
   } else if { [active_members pool_region3servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region3"
   } else if { [active_members pool_region4servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region4"
   } else if { [active_members pool_region1servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region1"
       }
} else if { ([matchclass [IP::client_addr] eq $::region3clients]) {
   if { [active_members pool_region3servers] > 0 } {
       HTTP::redirect "http://[HTTP::host]/region3"
```

```
      } else if { [active_members pool_region4servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region4"
      } else if { [active_members pool_region1servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region1"
      } else if { [active_members pool_region2servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region2"
          }
   } else if { ([matchclass [IP::client_addr] eq $::region4clients]) {
      if { [active_members pool_region4servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region4"
      } else if { [active_members pool_region1servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region1"
      } else if { [active_members pool_region2servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region2"
      } else if { [active_members pool_region3servers] > 0 } {
          HTTP::redirect "http://[HTTP::host]/region3"
          }
    }
}
```

## NetScaler Solution:

```
add patset region1
bind patset region1 10.102.58.200
bind patset region1 10.102.58.201
bind patset region1 10.102.58.202
bind patset region1 10.102.58.203

add patset region2
bind patset region2 10.102.59.200
bind patset region2 10.102.59.201
bind patset region2 10.102.59.202
bind patset region2 10.102.59.203

add patset region3
bind patset region3 10.102.10.200
bind patset region3 10.102.10.201
bind patset region3 10.102.10.202
bind patset region3 10.102.10.203

add patset region4
bind patset region4 10.102.58.400
bind patset region4 10.102.58.401
bind patset region4 10.102.58.402
bind patset region4 10.102.58.403

add responder action region1_act redirect HTTP.REQ.HOSTNAME.APPEND("/Region1")
add responder policy region1_pol
'CLIENT.IP.SRC.TYPECAST_TEXT_T.CONTAINS_ANY("region1")' region1_act

add responder action region2_act redirect HTTP.REQ.HOSTNAME.APPEND("/Region2")
add responder policy region2_pol
'CLIENT.IP.SRC.TYPECAST_TEXT_T.CONTAINS_ANY("region2")' region2_act

add responder action region3_act redirect HTTP.REQ.HOSTNAME.APPEND("/Region3")
add responder policy region3_pol
'CLIENT.IP.SRC.TYPECAST_TEXT_T.CONTAINS_ANY("region3")' region3_act

add responder action region4_act redirect HTTP.REQ.HOSTNAME.APPEND("/Region4")
add responder policy region4_pol
'CLIENT.IP.SRC.TYPECAST_TEXT_T.CONTAINS_ANY("region4")' region4_act
```

Here NetScaler is looking at the incoming Client source IP address and allocating it into a region. Based on the region allocated the resulting URL will have the extension.