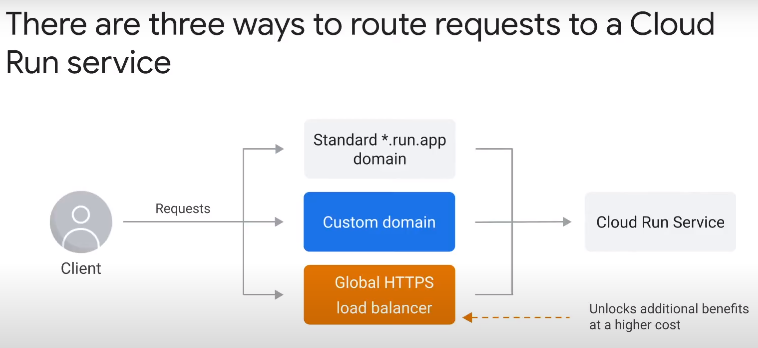
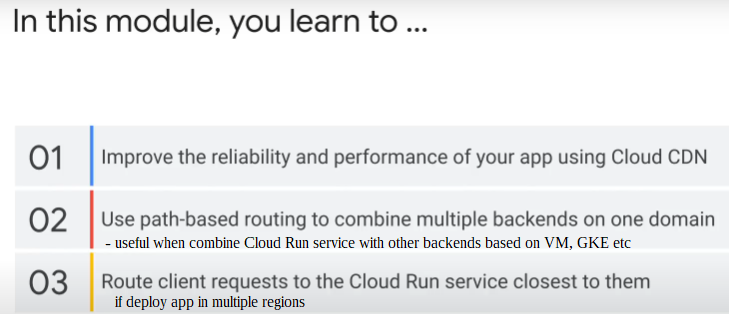
Application Development with Cloud Run

### Serving Requests

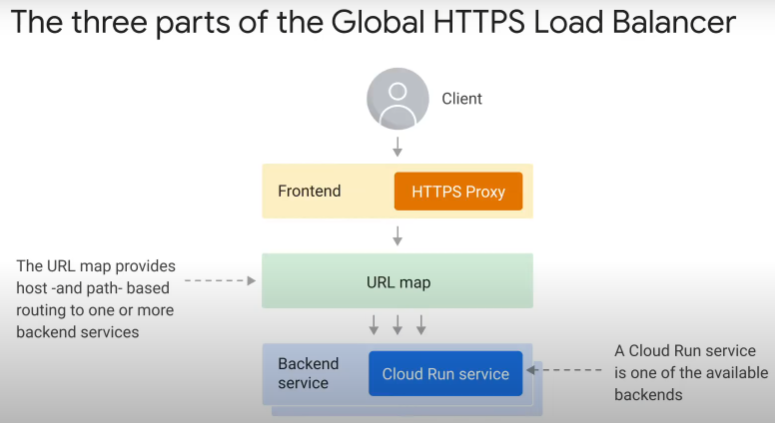
#### Introduction



* If enable own custom domain, Google automatically provisions an SSL certificate and manages renewal.
* Routing through external global HTTPS load balancer gives more features, but higher fixed cost.



#### Global HTTPS Load Balancer

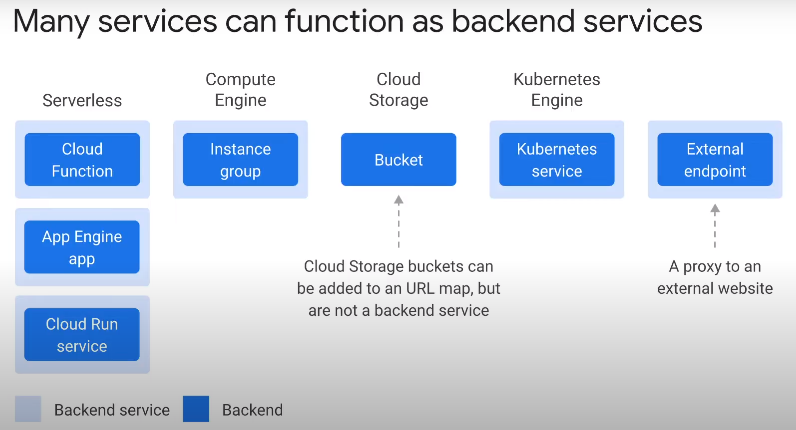


Three parts:

1. **Front end**
   * application interacts with requests
   * single static IP address
   * HTTPS proxy
2. **URL map**
3. **Backend service**
   * made up of one or more backends (e.g. Cloud Run service)

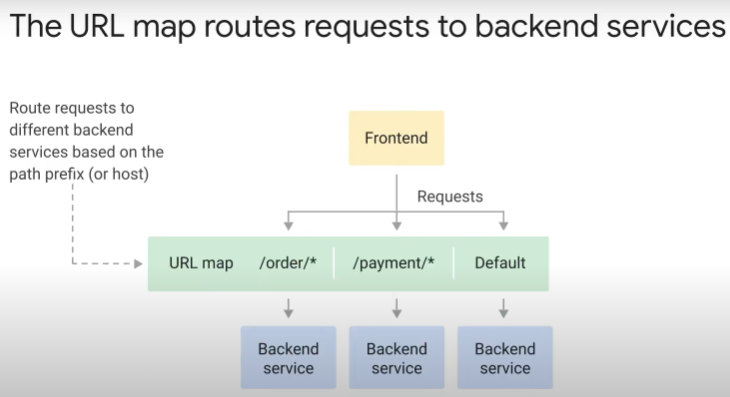
cross-region load balancing – can deploy application in multiple regions, and send client traffic to region nearest to them

Looking into the 3 parts in details:

**Backend service**

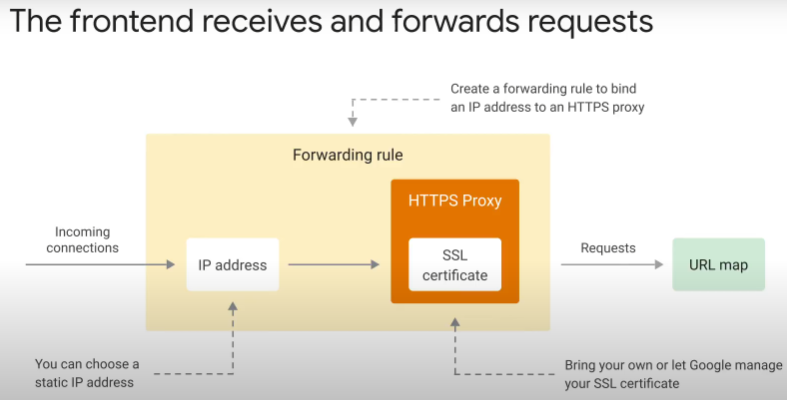
* all the backends in a backend service must be of the same type.
  + e.g. all serverless. Not serverless + GKE.
* backend service is necessary abstraction because
  + backends are regional while url map is global
  + can put multiple backends in different regions together in one backend service
* Cloud Storage bucket is not a backend service.
  + But can be added to URL map
* external HTTPS load balancer will proxy requests to an external endpoint
  + can be used to incrementally rebuild application during cloud migration from on-premise
* Cloud CDN is enabled at the backend service level
* use with global load balancer to distribute traffic globally

**URL map**



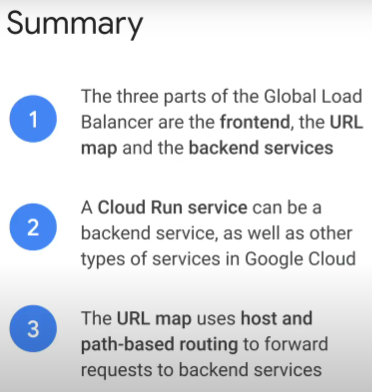
* different path prefix in web request > different backend service
* have backend services in an application with a bucket to host static assets
* can even forward certain route requests to service running on-premise in data center.
* Can use to isolate parts of application from each other

**Frontend**

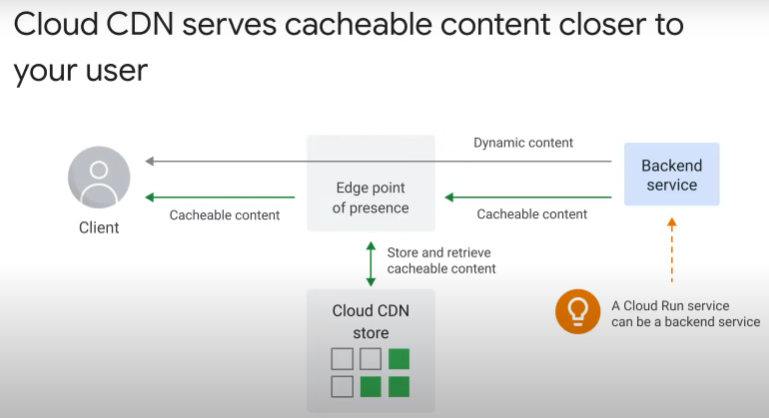


Three components:

1. static IP address – in front of web application
2. HTTPS proxy
   * configure using own SSL certificate, or have Google provision certificate and manage renewal.
   * forward requests to URL map
3. Forwarding rule – directs incoming TCP connections from static IP address to HTTPS proxy



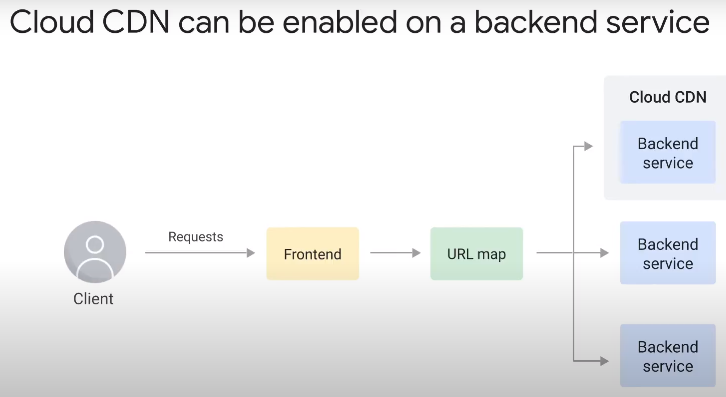
#### Cloud CDN



Why use Cloud CDN?

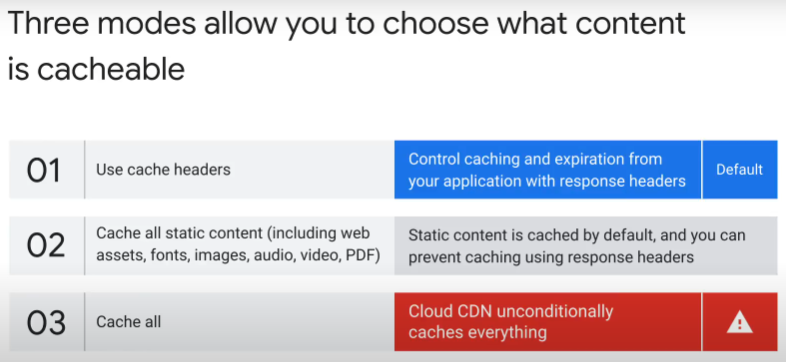
* Content Delivery Network
* serves cacheable content, from network edge point of presence closer to clients
* dynamic content is served directly from backend service
* improves availability and performance of application
* cloud CDN store
  + cached content expires
  + has limited total capacity
    - so some cached content may be removed earlier, before expiry

Cloud CDN is enabled on backend service:

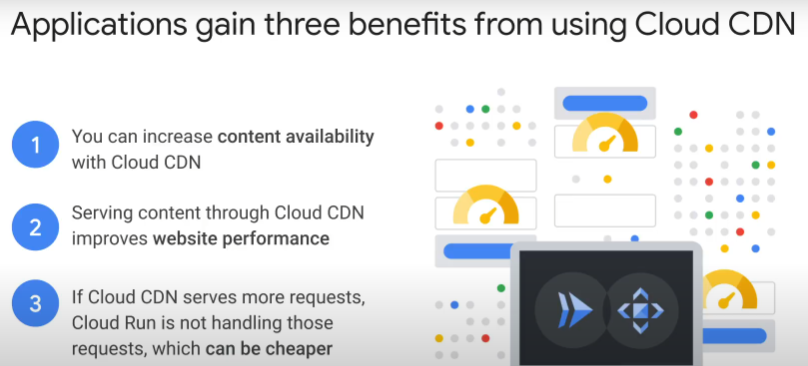


* can be enabled for any backend service
* can use path-based routing to combine backends with Cloud CDN + backends without Cloud CDN

3 modes to choose what content to cache:

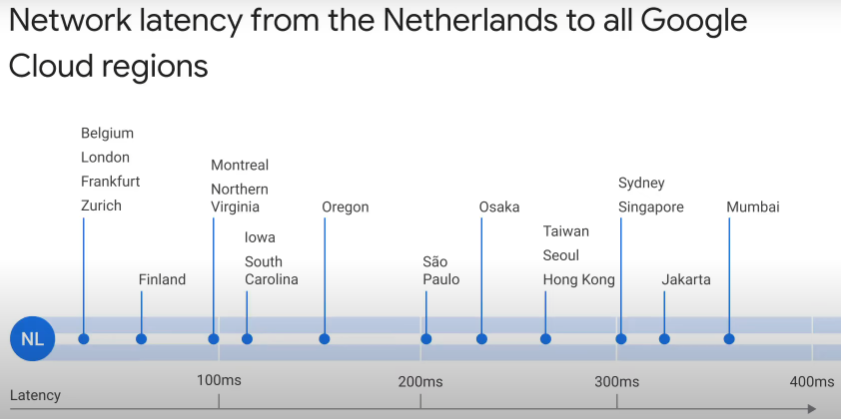


1. when application responds to web request, use response header to tell Cloud CDN how long to cache before fetch again.
2. response header: no cache
3. cache all for 1hr.
   * cannot override cache from application code once response sent.
   * may use a lot of cache storage

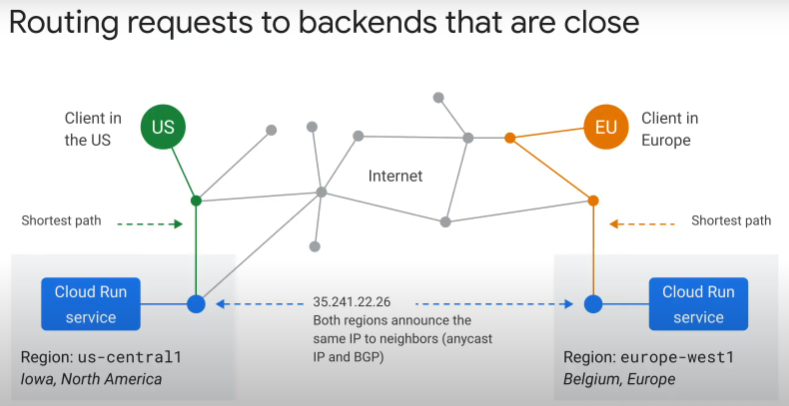


#### Multi-regional applications

Network Latency

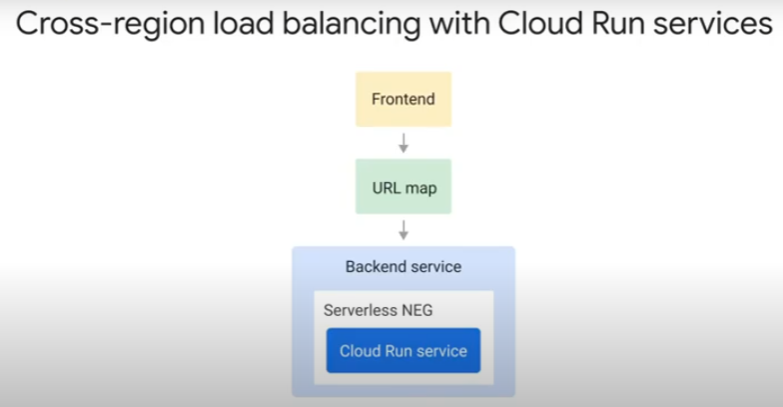
* a reason for serving application from multiple regions, if audience is global
* use **cross-regional load balancing**: global HTTPS load balancing with backend services
* a region is a data centre
* no fast link between Europe and Asia. Traffic usually goes through US.

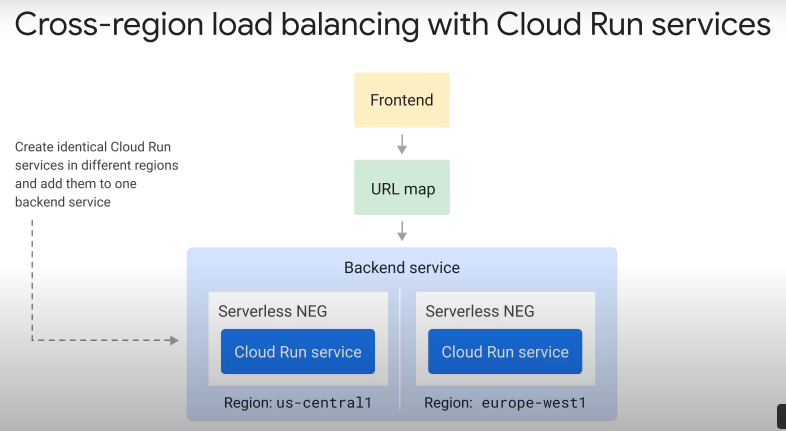
How global HTTPS load balancer implements cross-regional load balancing:



* application deployed in 2 different regions.
* Requests are routed to backend region that is closer in terms of network distance
  + example:
  + US traffic > US data region
  + Europe traffic > Europe data region
* Frontend in both regions use same static IP address, BGP (Border Gateway Protocol) to exchange routing info with other ISPs
* if one region fails, Google Cloud forwards requests from clients in failing region to the nearest working region over internal network.
  + Reliability, uptime benefit
* but no Google health check for failure of backend (e.g. Cloud Run service) or your application.

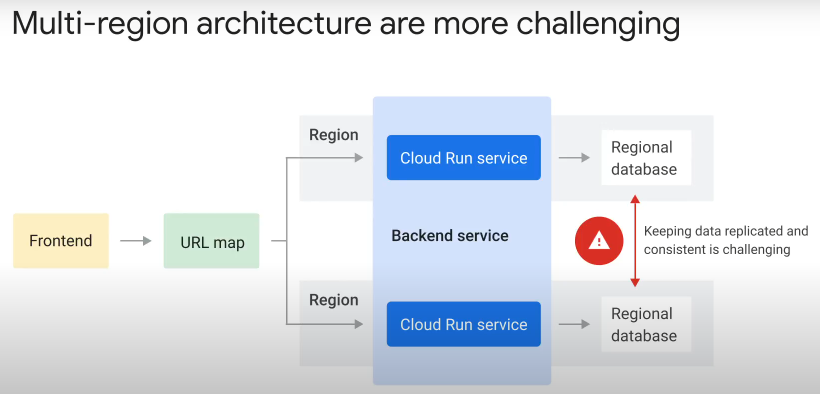
Serverless Network Endpoint Group (NEG) needed for cross-regional load balancing

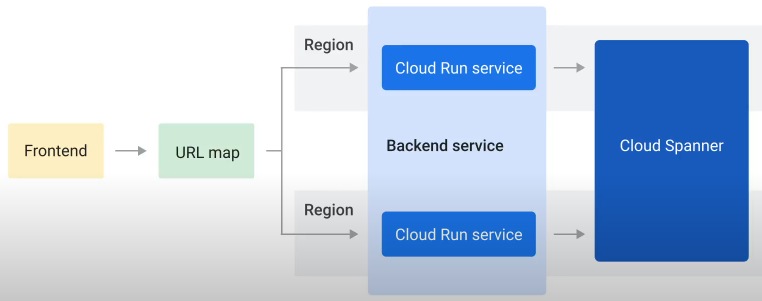


* At the back:
  + backend service added to URL map
  + backend service has backend
  + all serverless products on Google Cloud (App Engine, Cloud Run, Cloud Function) need a “backend” called serverless NEG.
  + The backend and serverless NEG are **regional** resources
  + The backend service and HTTPS load balancer are **global** resources
* example:
* multiple identical Cloud Run services in various regions
* single backend service
* global static IP address in frontend
* may not have HTTP health checks for Cloud Run failover
* web requests routed to nearest region
* => cross-regional load balancing by global HTTPS load balancer.

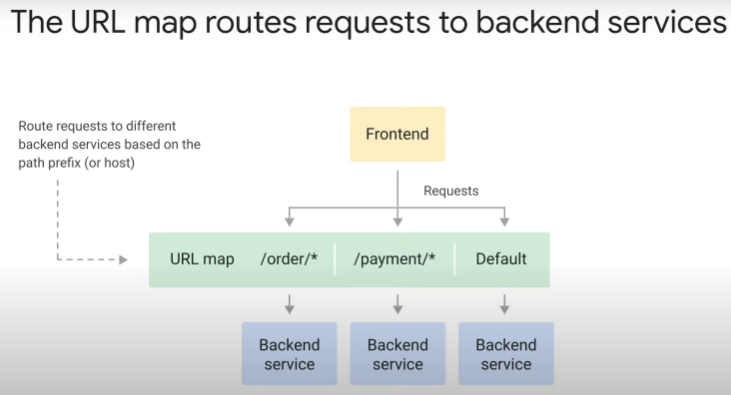
Data is a challenge for multi-regional architecture

* nearly all applications need to store and retrieve data
* want all users in all regions to see same data (consistency)
* challenging if use regional databases, e.g. Cloud SQL
* a solution is to use a multi-regional database, e.g. Cloud Spanner
  + strongly consistent reads, global transactions

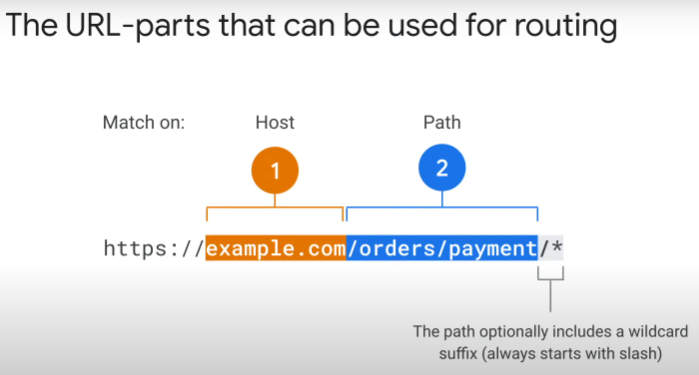




#### Host and path-based routing



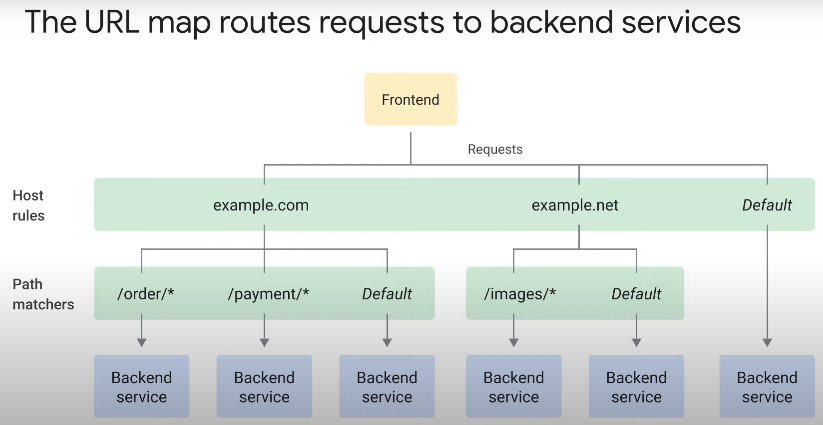
* Global HTTPS load balancer helps to integrate multiple different backends using host and path-based routing.
* Recall: global load balancer: frontend + URL map + backend.



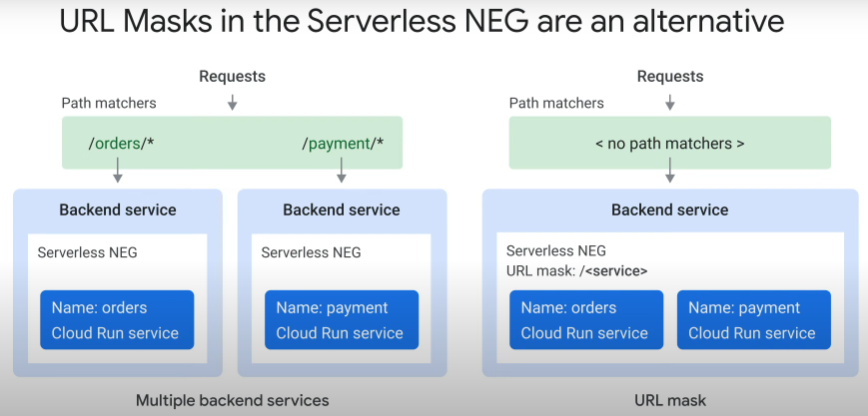
* The URL map matches requests/route to backend services.
* 2 parts:

|  |  |
| --- | --- |
| **Host name** | **Path name** |
| can have default route | can have default route |
| e.g. xxx.com, xxx.net, xxx.sg | option to include a wildcard suffix   * must start with slash “/” * must end with wilcard “\*” |

* The method to combine different serverless backends with bucket to host static assets
  + or, forward certain requests to on-premises services in data center



URL masks in serverless NEG

* single serverless NEG backend can point to single serverless backend (Cloud Run, Cloud Function or App Engine services)
* if backends consist of multiple severless services

|  |  |
| --- | --- |
| single serverless NEG backend can point to single serverless backend (Cloud Run, Cloud Function or App Engine services) | generic URL mask in single serverless NEG can point to multiple serverless backends.  Easier to configure with just 1 serverless NEG. |
|  | URL mask is a template in the URL schema  e.g. /<service> |
|  | serverless NEG uses template to extract service name from the incoming request URL, and map to correct backend |
|  | optional |

