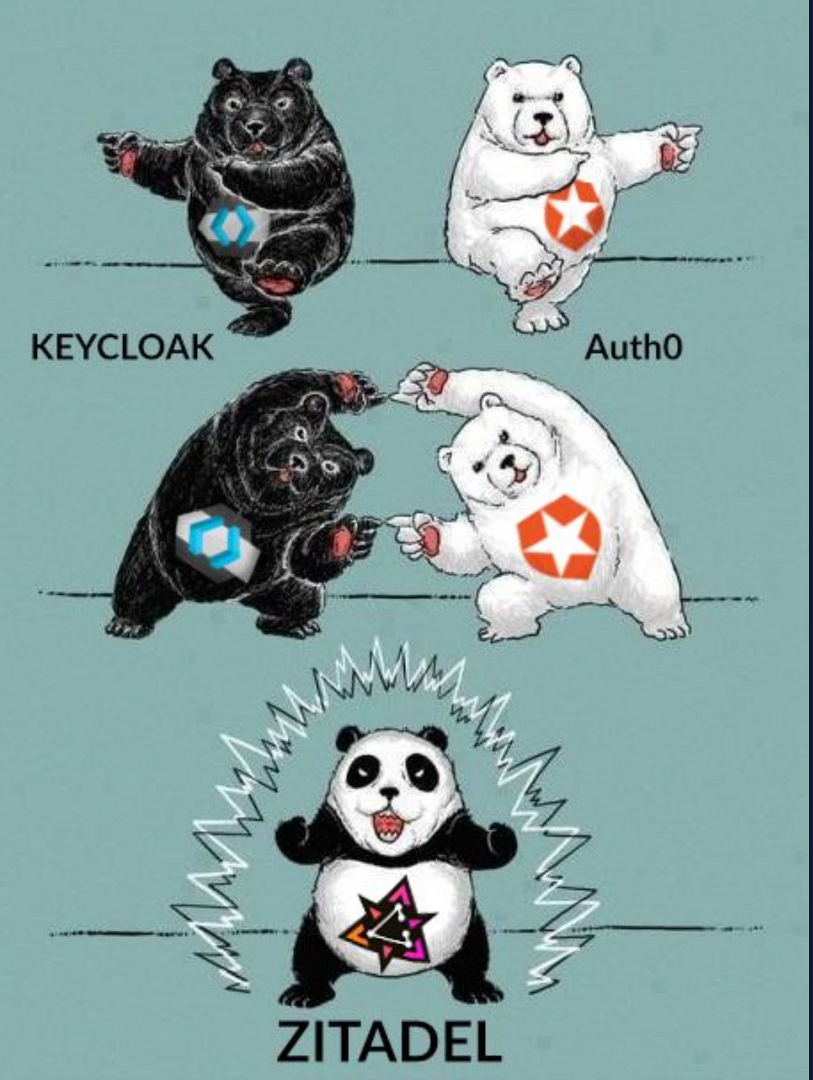
ZITADEL

What is the fuzz around serverless (containers)?

Open Source @ Siemens 2022

Disclaimer

Why I am talking about this?



The best of Auth0 and Keycloak combined.

Built for a serverless era.

Question

How to deploy globally without breaking the bank?

Kubernetes?!

+ Great Tool & High Maturity

+ Many, many, many, … options

o "Serverless" Kubernetes feels strange

- High Base Cost per PoP

- Slow(ish) scaling per Cluster

Serverless, a classification (attempt)

OPENSHIFT
www.coolest-homemade-costumes.com
SERVERLESS
JCB
AWS ECS
HEROKU
KUBERNETES
CAPM Europe SA
DOCKER SWARM

Function as a Service (FaaS)

- Small composable unit
- Tricky shared context problems
- Strict language requirements
- Scale to 0 easy
- No Parallelism

Platform as a Service (PaaS)

- Explicit language support needed (debatable)
- Parallelism support
- Typical slower scaling
- Scale to 0 tricky

Container as a Service (CaaS)

- Composable unit
- Run any language
- Parallelism support
- Scale to 0 easy

The fuzz

Easy Ops / No Ops

Elasticity

Edge Locations

Pay as you go

Observability

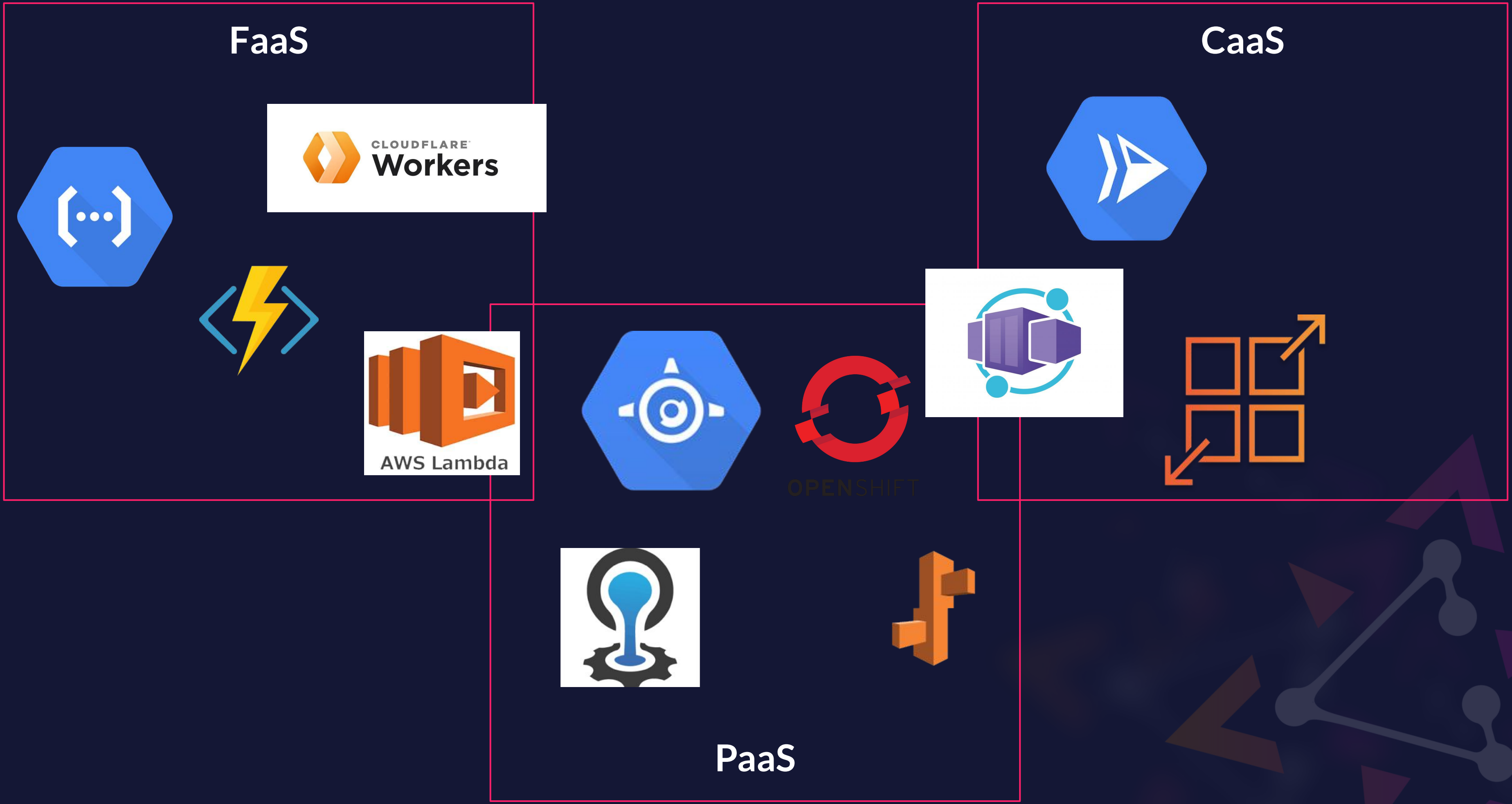
Scale to 0

Scalability

Low Capex

IF THE GRASS IS ALWAYS GREENER ON THE OTHER SIDE
WHEN YOU GET TO THE OTHER SIDE, IS THE SIDE YOU WERE FIRST ON GREENER?
makeameme.org

Some of the “Serverless” Offerings


FaaS
CaaS
CLOUDFLARE Workers
AWS Lambda
OPENSHIFT
PaaS

Why Serverless Containers

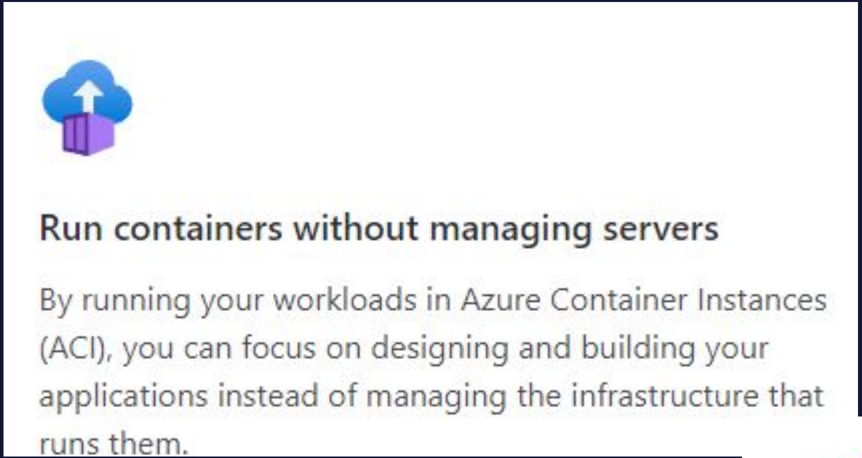
+ K8s cross compatibility
+ Fast Scaling
+ Low Base Price
- Some strange things (more later)

Challenges

ZITADEL

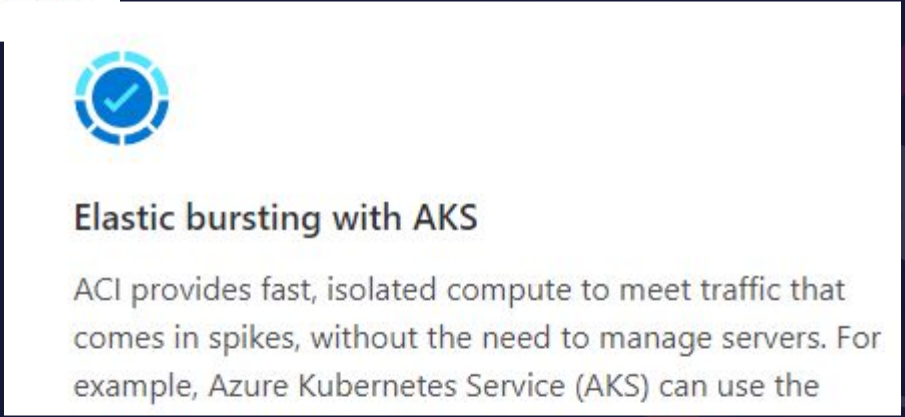
Fun with Load Balancing

First up, M$



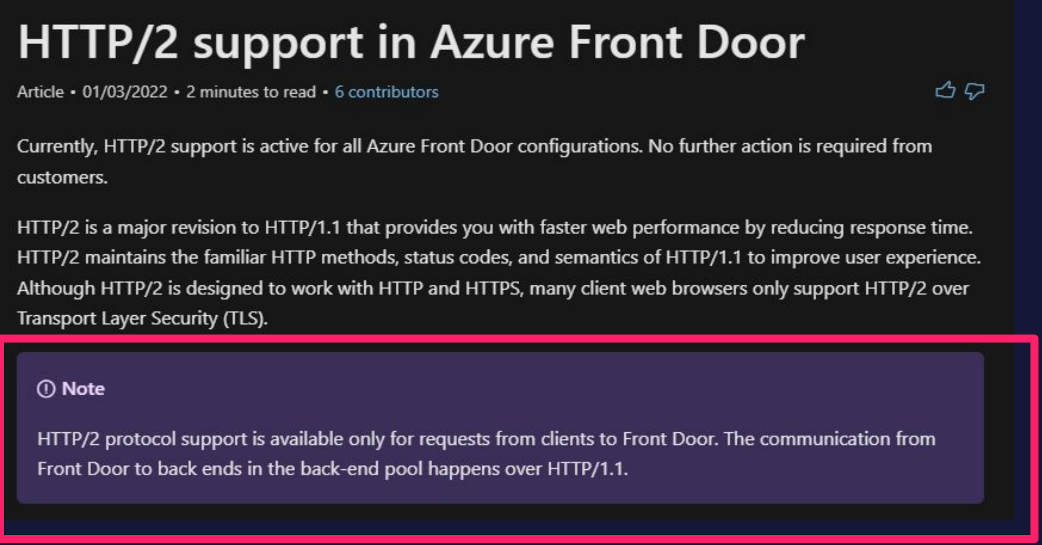
Run containers without managing servers

By running your workloads in Azure Container Instances (ACI), you can focus on designing and building your applications instead of managing the infrastructure that runs them.



Elastic bursting with AKS

ACI provides fast, isolated compute to meet traffic that comes in spikes, without the need to manage servers. For example, Azure Kubernetes Service (AKS) can use the



HTTP/2 support in Azure Front Door

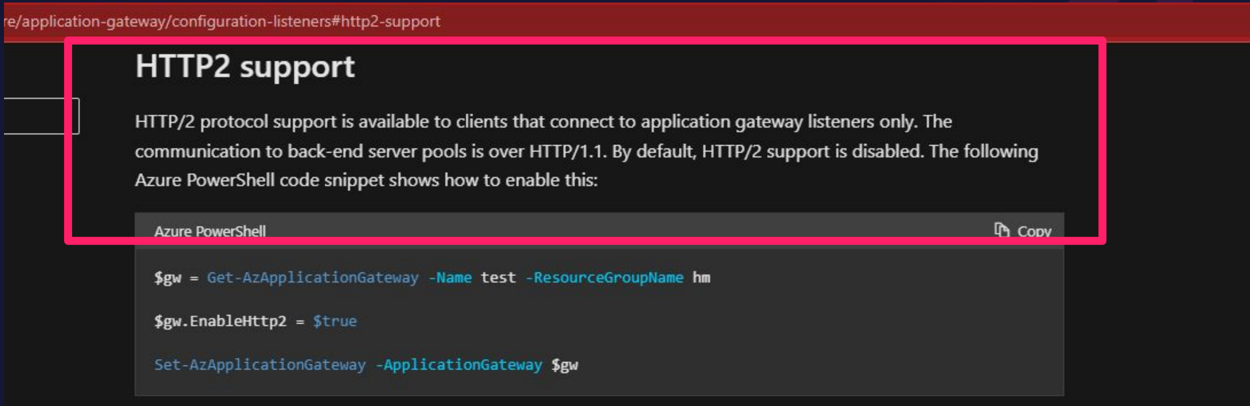
Article • 01/03/2022 • 2 minutes to read • 6 contributors

Currently, HTTP/2 support is active for all Azure Front Door configurations. No further action is required from customers.

HTTP/2 is a major revision to HTTP/1.1 that provides you with faster web performance by reducing response time. HTTP/2 maintains the familiar HTTP methods, status codes, and semantics of HTTP/1.1 to improve user experience. Although HTTP/2 is designed to work with HTTP and HTTPS, many client web browsers only support HTTP/2 over Transport Layer Security (TLS).

Note

HTTP/2 protocol support is available only for requests from clients to Front Door. The communication from Front Door to back ends in the back-end pool happens over HTTP/1.1.



re/application-gateway/configuration-listeners#http2-support

HTTP2 support

HTTP/2 protocol support is available to clients that connect to application gateway listeners only. The communication to back-end server pools is over HTTP/1.1. By default, HTTP/2 support is disabled. The following Azure PowerShell code snippet shows how to enable this:

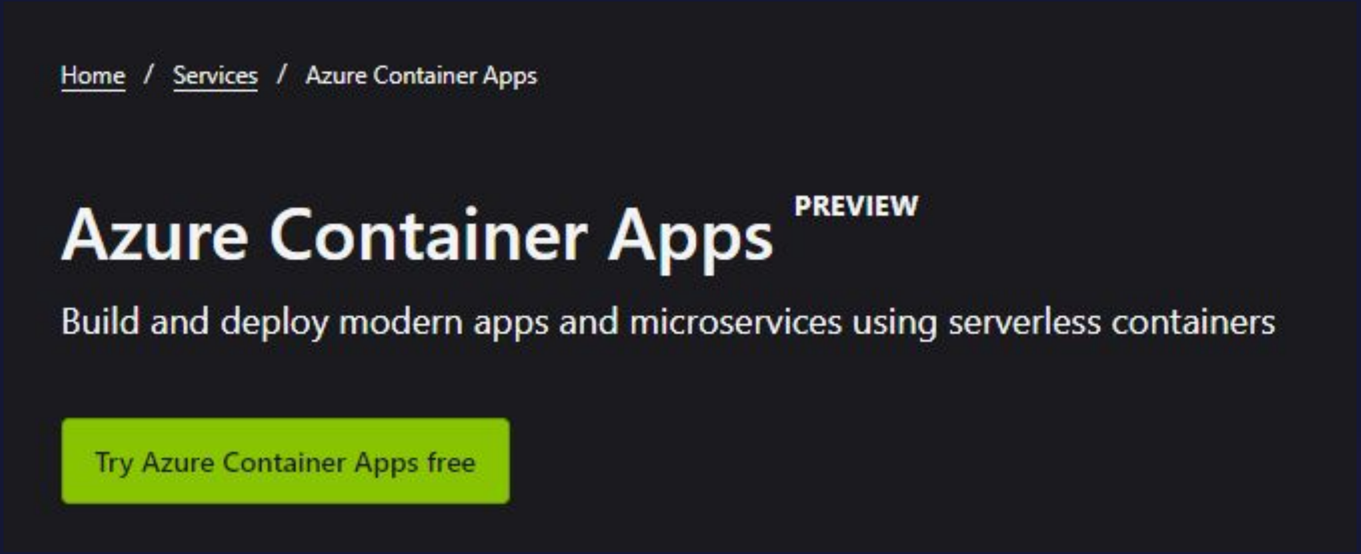
Azure PowerShell

```
$gw = Get-AzApplicationGateway -Name test -ResourceGroupName hm

$gw.EnableHttp2 = $true

Set-AzApplicationGateway -ApplicationGateway $gw
```

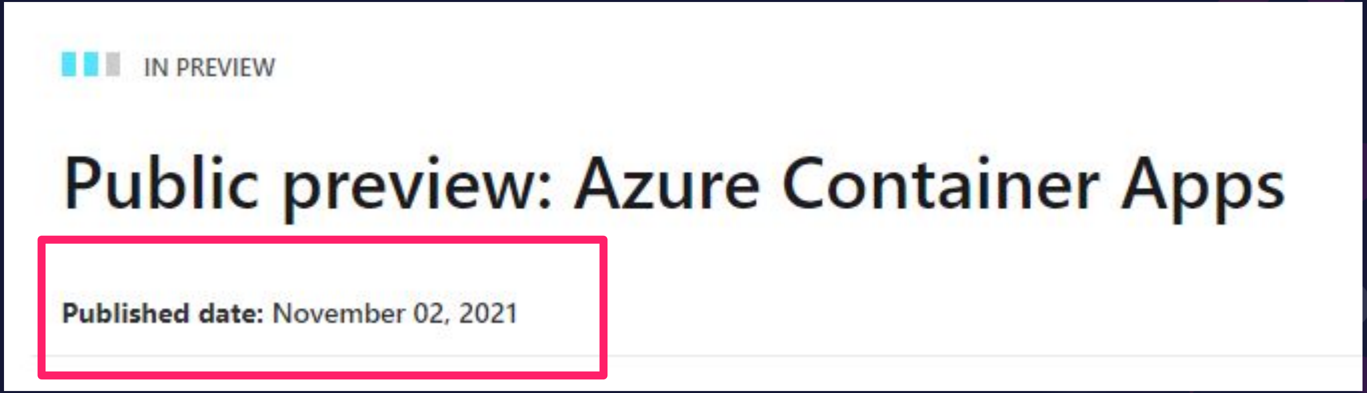
WHY YOU
DO THIS TO ME
makeameme.org



Home / Services / Azure Container Apps

Azure Container Apps PREVIEW

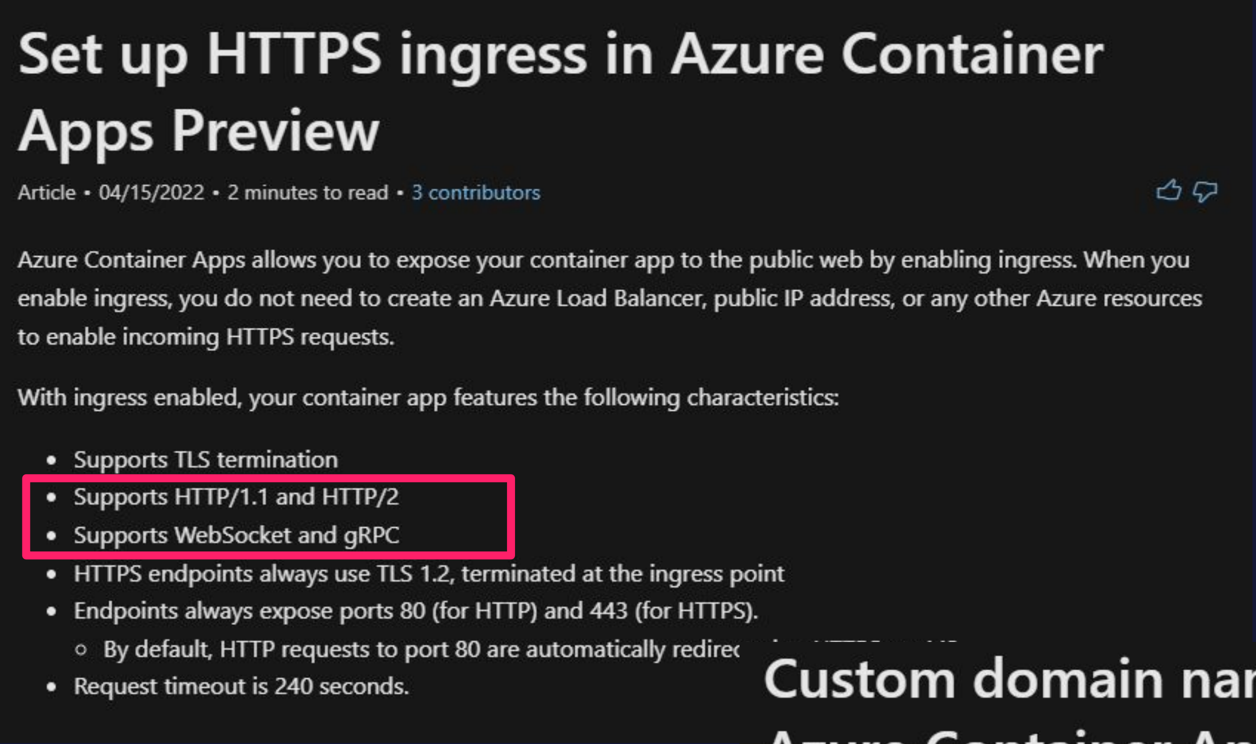
Build and deploy modern apps and microservices using serverless containers



IN PREVIEW

Public preview: Azure Container Apps

Published date: November 02, 2021



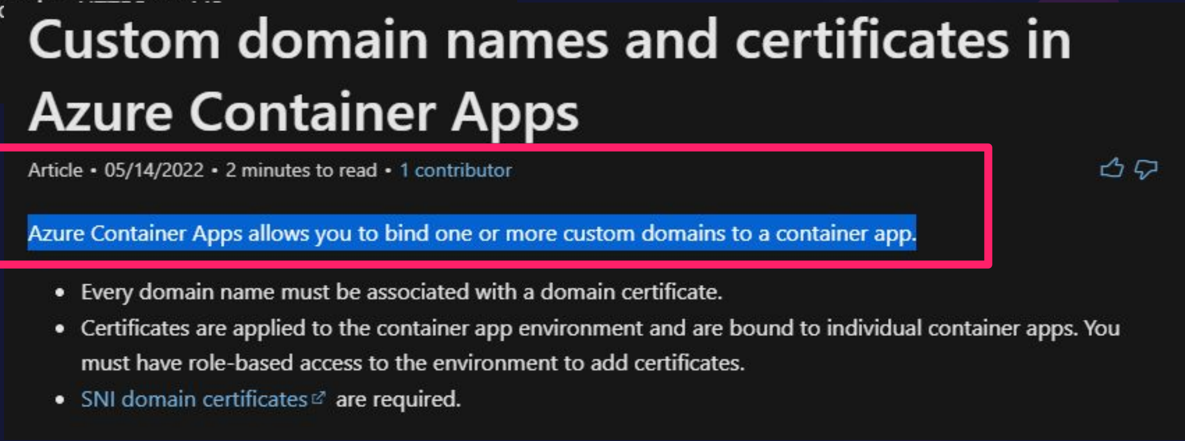
Set up HTTPS ingress in Azure Container Apps Preview

Article • 04/15/2022 • 2 minutes to read • 3 contributors

Azure Container Apps allows you to expose your container app to the public web by enabling ingress. When you enable ingress, you do not need to create an Azure Load Balancer, public IP address, or any other Azure resources to enable incoming HTTPS requests.

With ingress enabled, your container app features the following characteristics:

- Supports TLS termination
- Supports HTTP/1.1 and HTTP/2
- Supports WebSocket and gRPC
- HTTPS endpoints always use TLS 1.2, terminated at the ingress point
- Endpoints always expose ports 80 (for HTTP) and 443 (for HTTPS).
  - By default, HTTP requests to port 80 are automatically redirec
- Request timeout is 240 seconds.

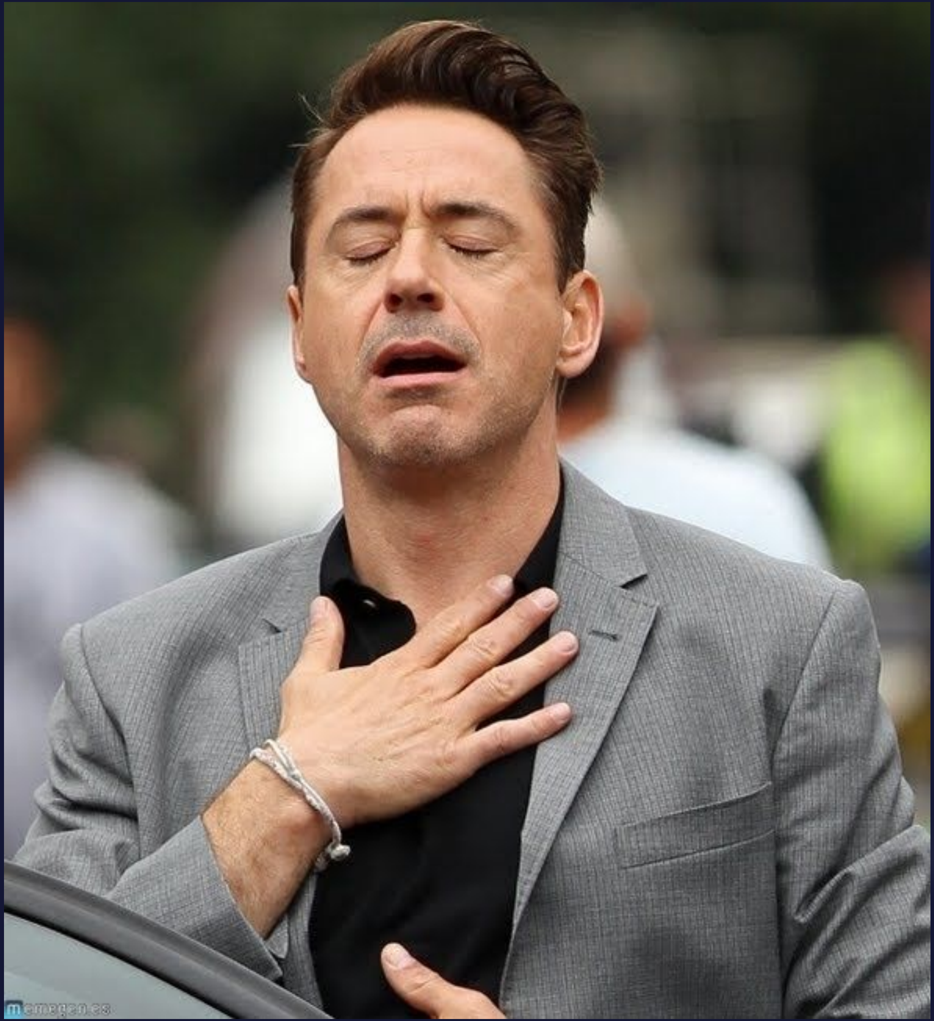


Custom domain names and certificates in Azure Container Apps

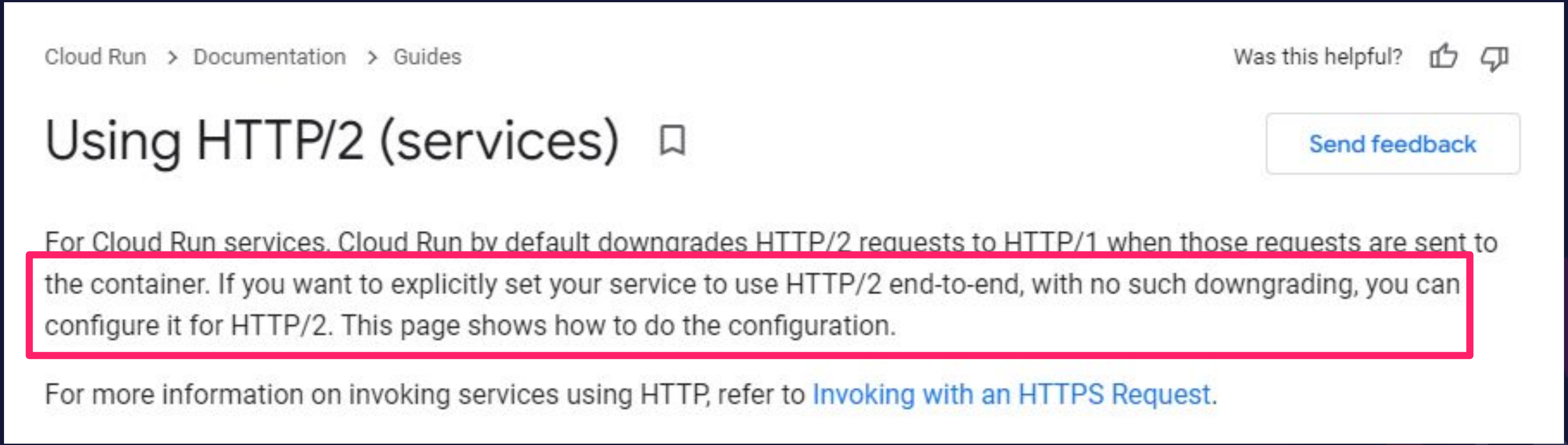
Article • 05/14/2022 • 2 minutes to read • 1 contributor

Azure Container Apps allows you to bind one or more custom domains to a container app.

- Every domain name must be associated with a domain certificate.
- Certificates are applied to the container app environment and are bound to individual container apps. You must have role-based access to the environment to add certificates.
- SNI domain certificates are required.


memegen.es

How about Google?


Cloud Run > Documentation > Guides

Was this helpful?

Using HTTP/2 (services)

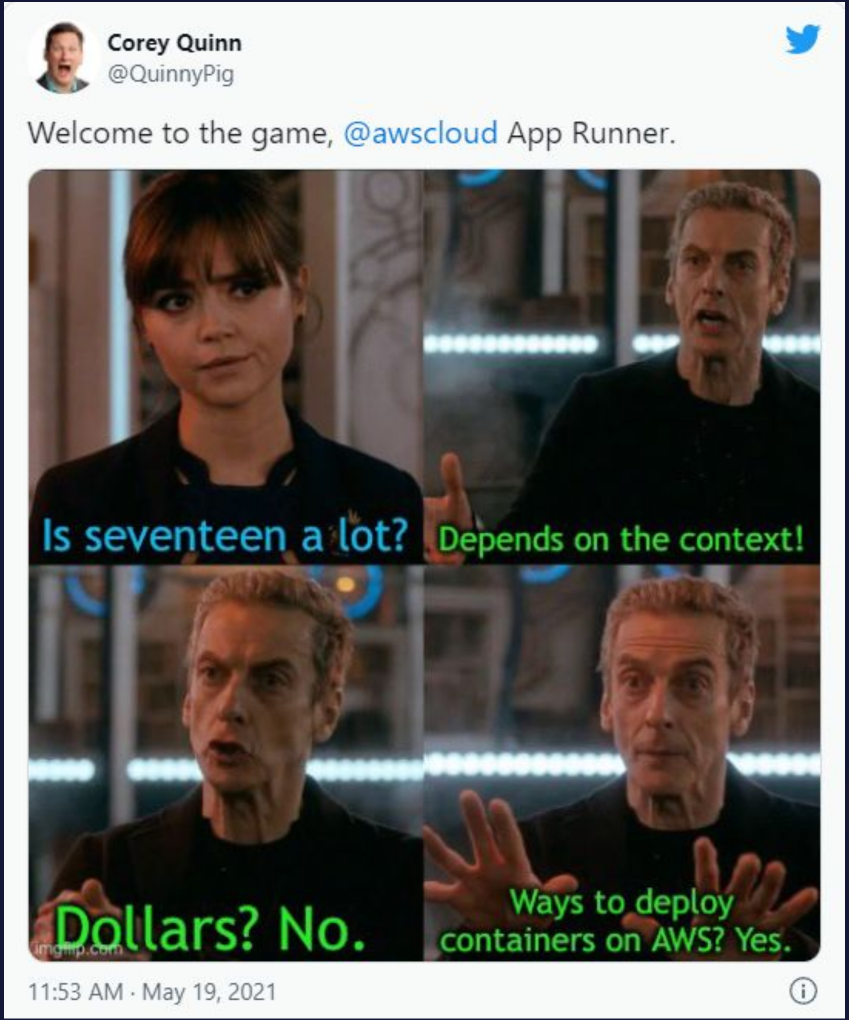
Send feedback

For Cloud Run services, Cloud Run by default downgrades HTTP/2 requests to HTTP/1 when those requests are sent to the container. If you want to explicitly set your service to use HTTP/2 end-to-end, with no such downgrading, you can configure it for HTTP/2. This page shows how to do the configuration.

For more information on invoking services using HTTP, refer to Invoking with an HTTPS Request.

well i suppose it could be worse
ICANHASCHEEZBURGER.COM

How about AWS?


Corey Quinn
@QuinnyPig

Welcome to the game, @awscloud App Runner.

Is seventeen a lot?
Depends on the context!
Dollars? No.
Ways to deploy containers on AWS? Yes.
imgflip.com

11:53 AM · May 19, 2021

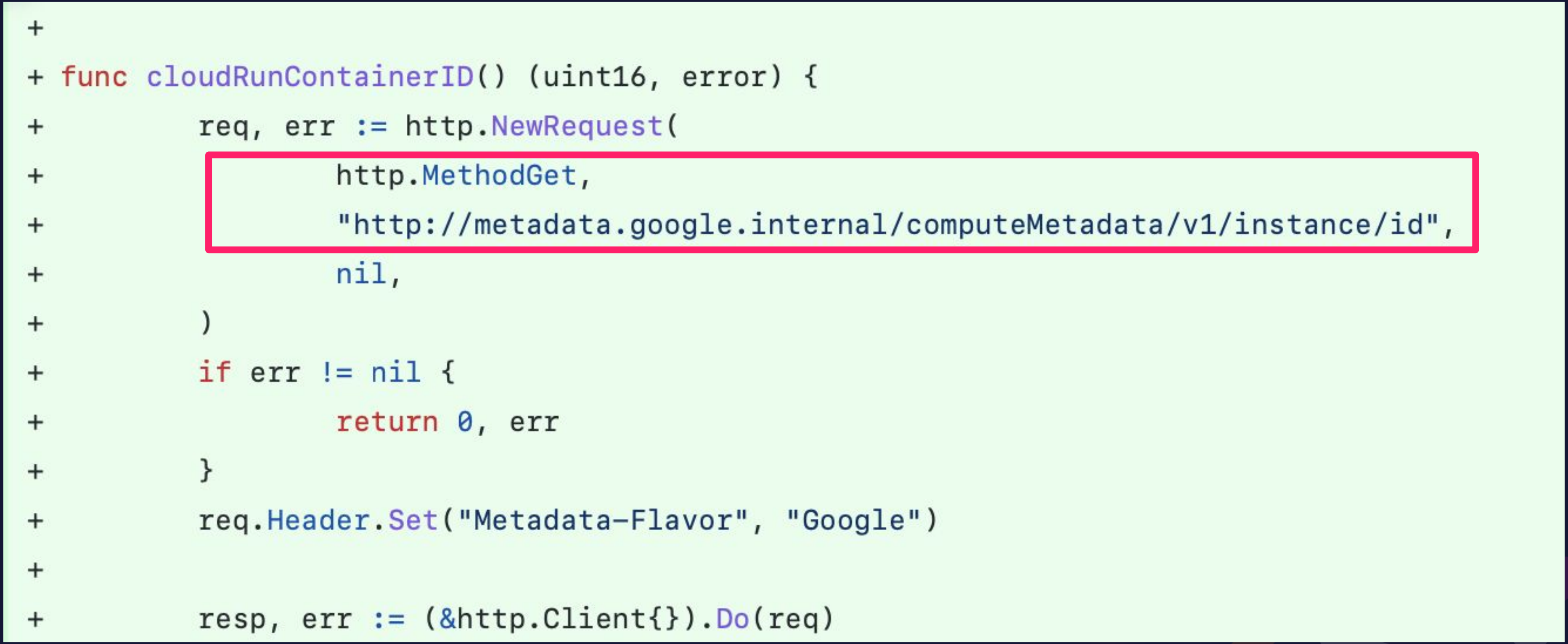
Unique IDs

If you use the “machine” to create unique IDs be aware!

- Hostnames are not reliable
- IPs Address are not reliable

+ Metadata server can be called to discover

o Runtime contract may be close to Knative



```
+
+ func cloudRunContainerID() (uint16, error) {
+         req, err := http.NewRequest(
+                 http.MethodGet,
+                 "http://metadata.google.internal/computeMetadata/v1/instance/id",
+                 nil,
+         )
+         if err != nil {
+                 return 0, err
+         }
+         req.Header.Set("Metadata-Flavor", "Google")
+
+         resp, err := (&http.Client{}).Do(req)
```

Pricing

THESE AREN'T THE PRICES
YOU'RE LOOKING FOR
memegenerator.net

Keep an eye on the price

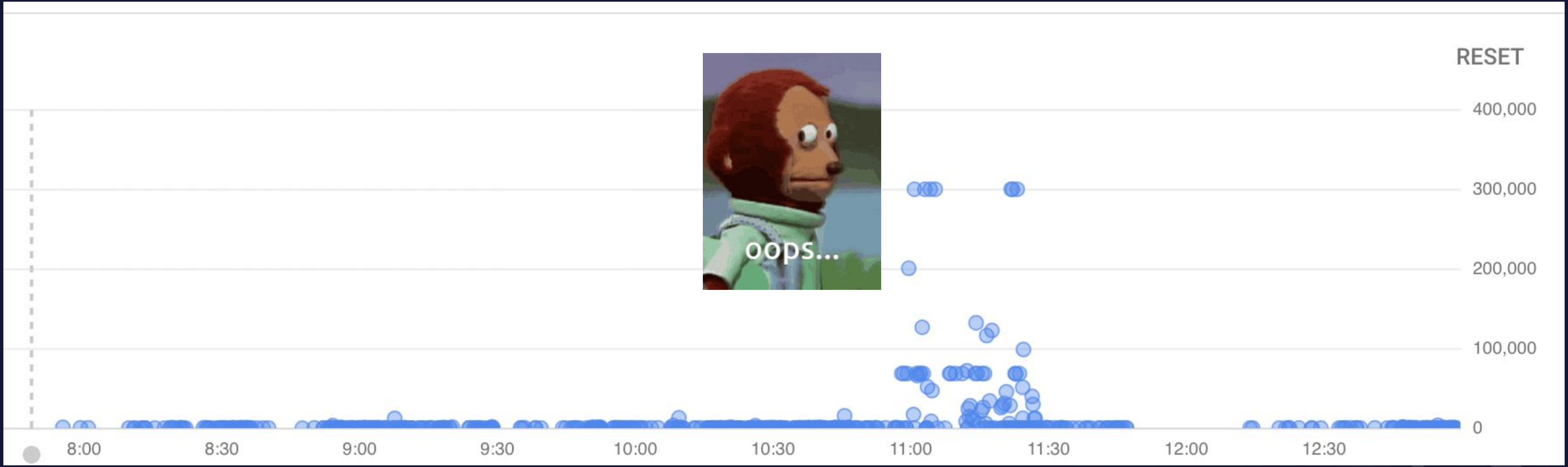
- Sometimes strange pricing ideas

+ Pay what you use/request

o burning infrastructure is discouraged

o setup budgets and rate limits!!

Cold Start Times


RESET
400,000
300,000
200,000
100,000
0
8:00
8:30
9:00
9:30
10:00
10:30
11:00
11:30
12:00
12:30
oops...

Mind the cold start

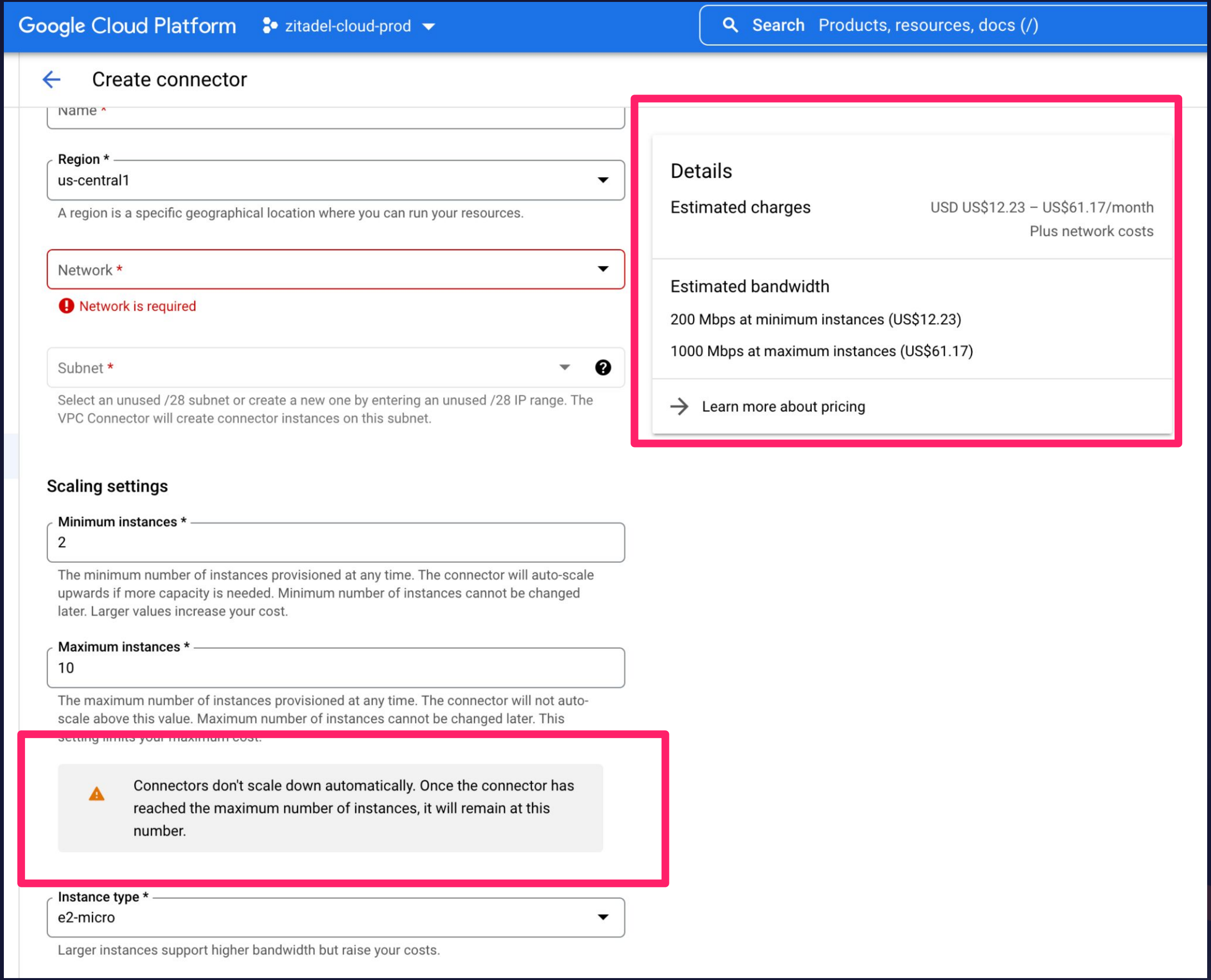
- Container Startup (100-1000ms)
- If you do startup verification this can hurt >10000ms

+ Still makes a fast(er) scaling

Networking (extended)

Fancy Networking does cost

- VPC Interconnect option
- Outbound NAT Gateway
- Content Delivery
- Web Application Firewall



Create connector

Name *

Region *
us-central1

A region is a specific geographical location where you can run your resources.

Network *

Network is required

Subnet *

Select an unused /28 subnet or create a new one by entering an unused /28 IP range. The VPC Connector will create connector instances on this subnet.

Scaling settings

Minimum instances *
2

The minimum number of instances provisioned at any time. The connector will auto-scale upwards if more capacity is needed. Minimum number of instances cannot be changed later. Larger values increase your cost.

Maximum instances *
10

The maximum number of instances provisioned at any time. The connector will not auto-scale above this value. Maximum number of instances cannot be changed later. This setting limits your maximum cost.

Connectors don't scale down automatically. Once the connector has reached the maximum number of instances, it will remain at this number.

Instance type *
e2-micro

Larger instances support higher bandwidth but raise your costs.

Details

Estimated charges USD US$12.23 – US$61.17/month
Plus network costs

Estimated bandwidth

200 Mbps at minimum instances (US$12.23)

1000 Mbps at maximum instances (US$61.17)

Learn more about pricing

SERVERLESS
ME
VPC CONNECTOR
ME
SERVERLESS
imgflip.com

Conclusion

Conclusion

Serverless is great for Geo Distributed and Bursty workloads as well as Developers.

Otherwise: Pick your poison

Contact

zitadel.com/contact

hi@zitadel.com

GitHub

Linkedin

Twitter

Discord

Try out ZITADEL for free, no strings attached

zitadel.com