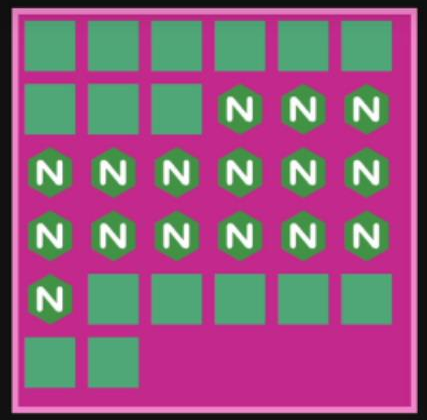
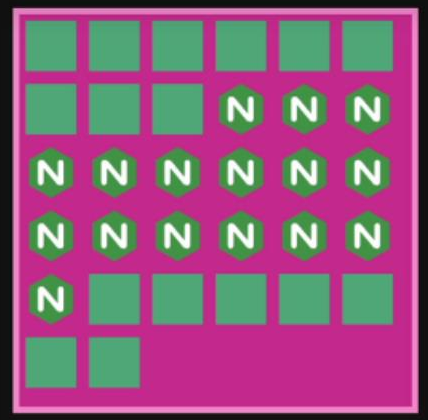
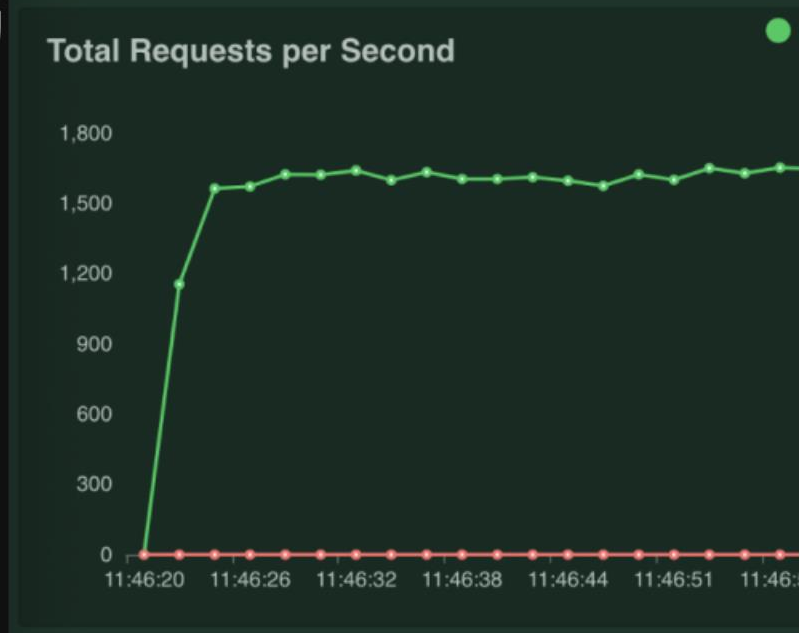
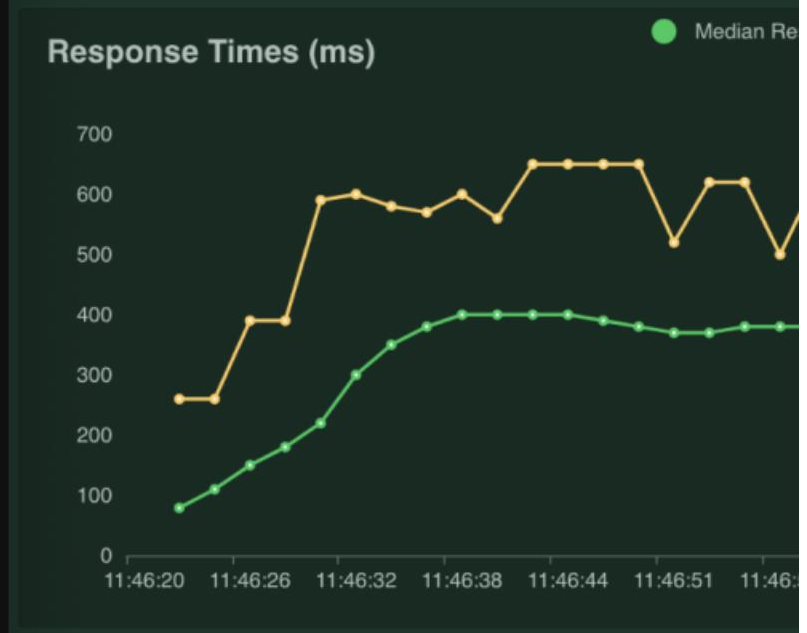
AUTOSCALING NGINX INGRESS CONTROLLER

KEDA









SCALING BASED ON NUMBER OF HTTP REQUESTS

1. **Expose metrics**
2. **Collect & store**
3. **Autoscaler**

> To autoscale on the number of requests you need:
> 1. Metrics.
> 2. A metrics collector.
> 3. An autoscaler to consume the metrics and increase the replicas.

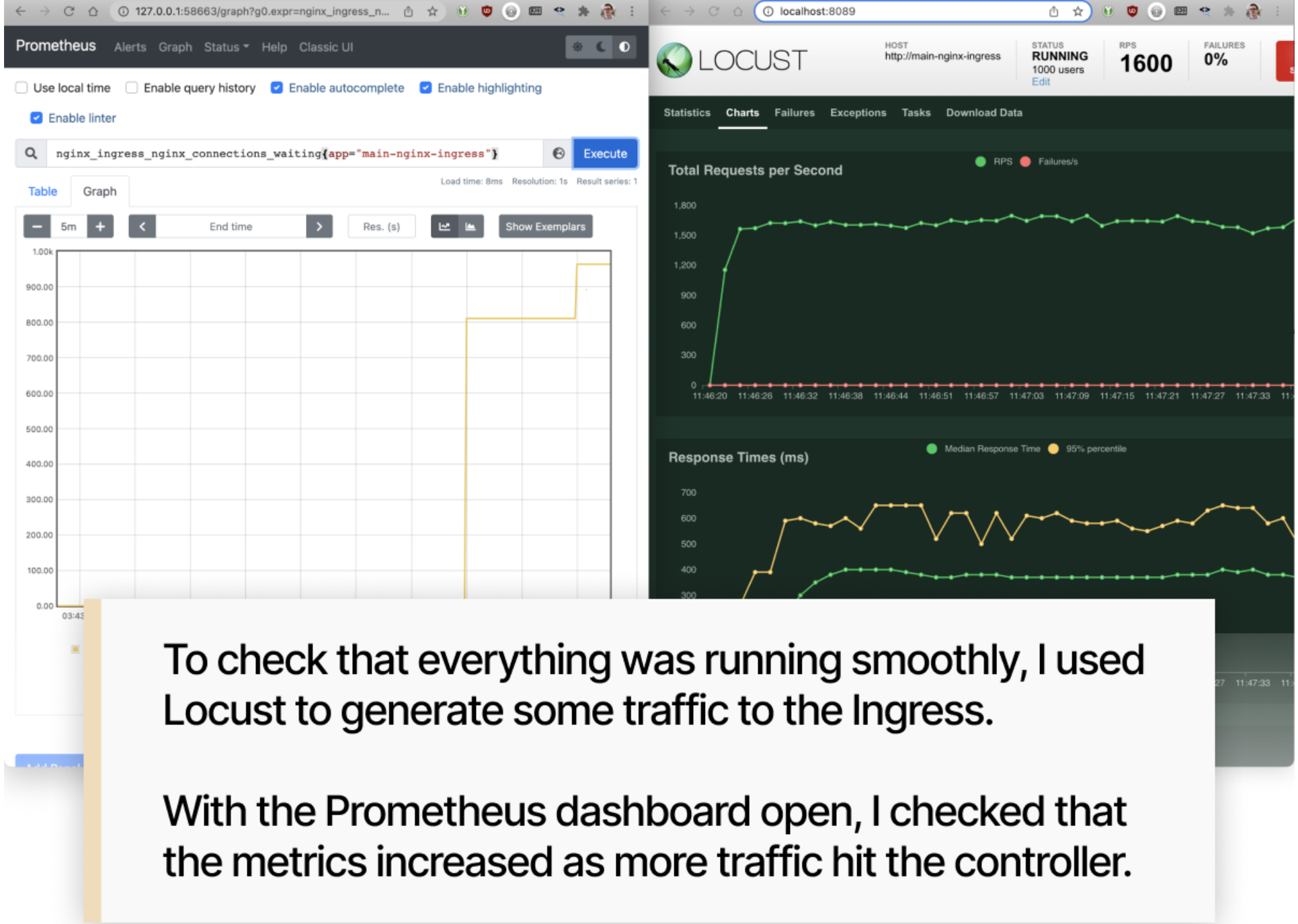
Stub status metrics

1

Name	Type	Descripti
`nginx_connections_accepted`	Counter	Accepted client connections.
`nginx_connections_active`	Gauge	Active client connections.
`nginx_connections_handled`	Counter	Handled client connections.
`nginx_connections_reading`	Gauge	Connections where NGINX is r

Next, you need a way to scrape the metrics.
As you've already guessed, you can install Prometheus to do so.
There are 2 ways to install Prometheus with the operator and without.
Since Nginx-ingress use annotations, the non-operator is the easiest choice.

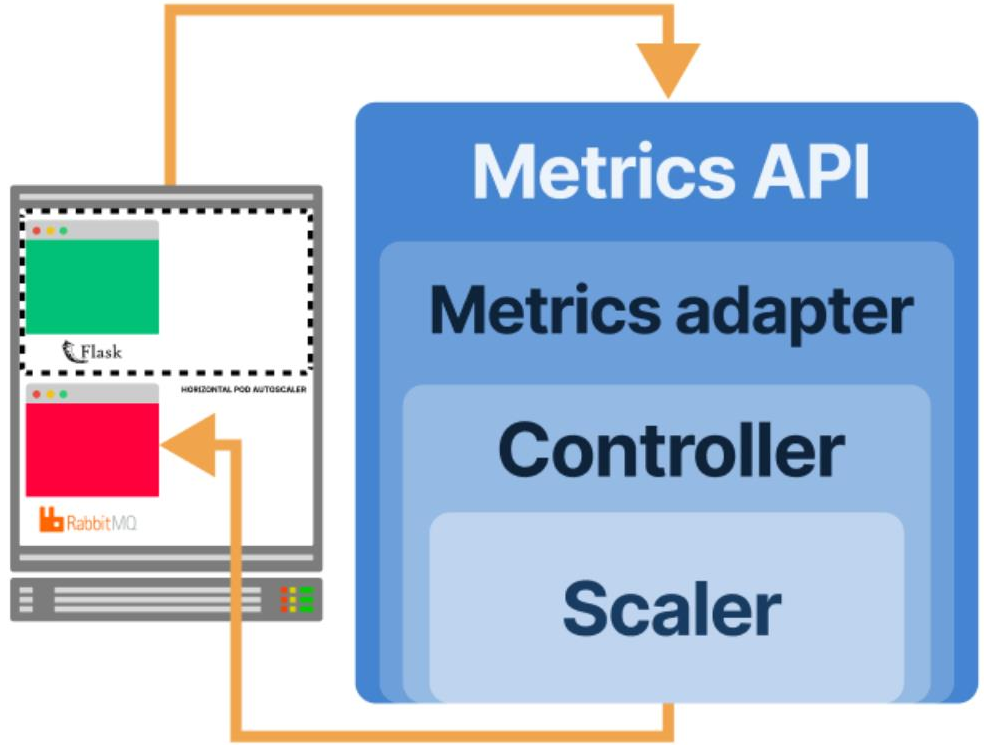
> Next, you need a way to scrape the metrics.
> As you've already guessed, you can install Prometheus to do so.
> Since Nginx-ingress use annotations for Prometheus, I installed the Prometheus server without Kubernetes operator.



To check that everything was running smoothly, I used Locust to generate some traffic to the Ingress.

With the Prometheus dashboard open, I checked that the metrics increased as more traffic hit the controller.

3



KEDA

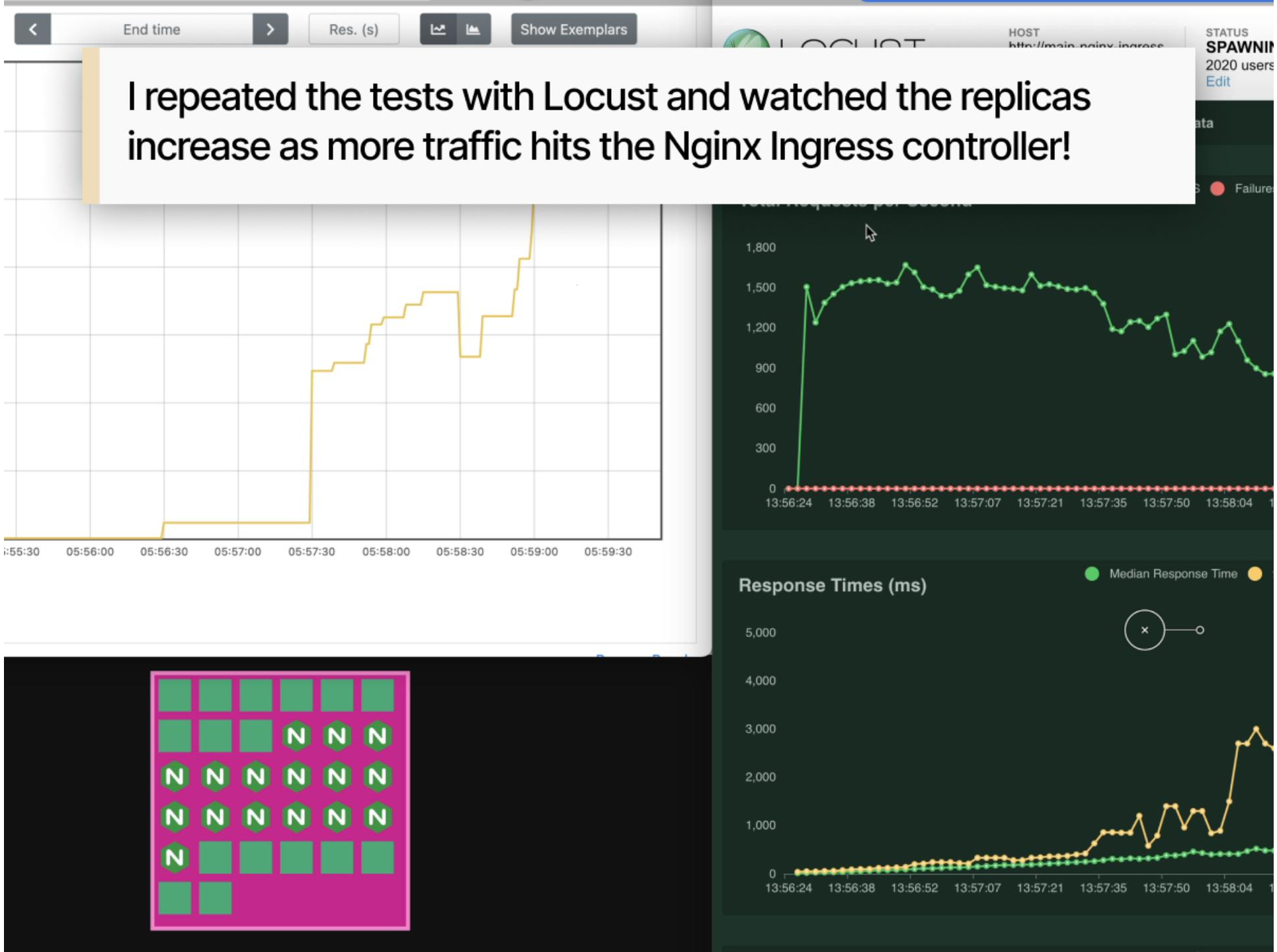
The last piece of the puzzle is the autoscaler. I decided to go with KEDA because:

1. It's an **autoscaler with a metrics server** (so I don't need to install 2 different tools).
2. It's **easier to configure than the Prometheus adapter.**
3. I can use the **HPA with PromQL.**

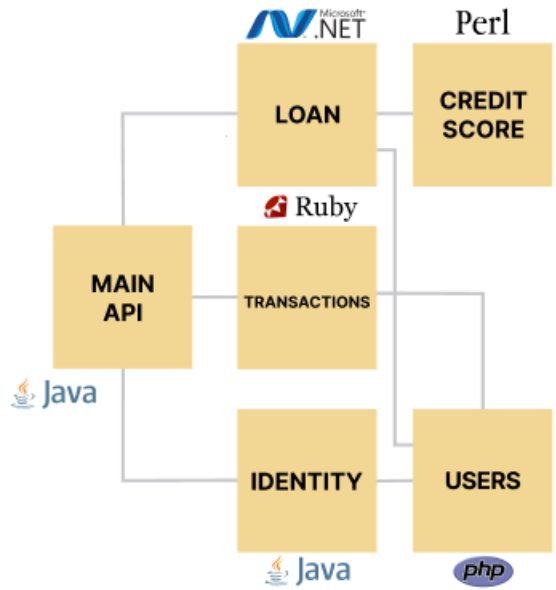
```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
 name: nginx-scale
spec:
 scaleTargetRef:
   kind: Deployment
   name: main-nginx-ingress
 minReplicaCount: 1
 maxReplicaCount: 20
 cooldownPeriod: 30
 pollingInterval: 1
 triggers:
 - type: prometheus
   metadata:
     serverAddress: http://prometheus-server
     metricName: nginx_connections_waiting_keda
     query: |
       sum(nginx_ingress_nginx_connections_waiting{app="main-nginx-ingress"})
     threshold: "100"
```

> Once I installed KEDA, the only thing I had to do was create a ScaledObject and configure the source of the metrics (Prometheus) and scale the Pods (with a PromQL query). **KEDA takes care of connecting the dots and creates the HPA automatically for me.**

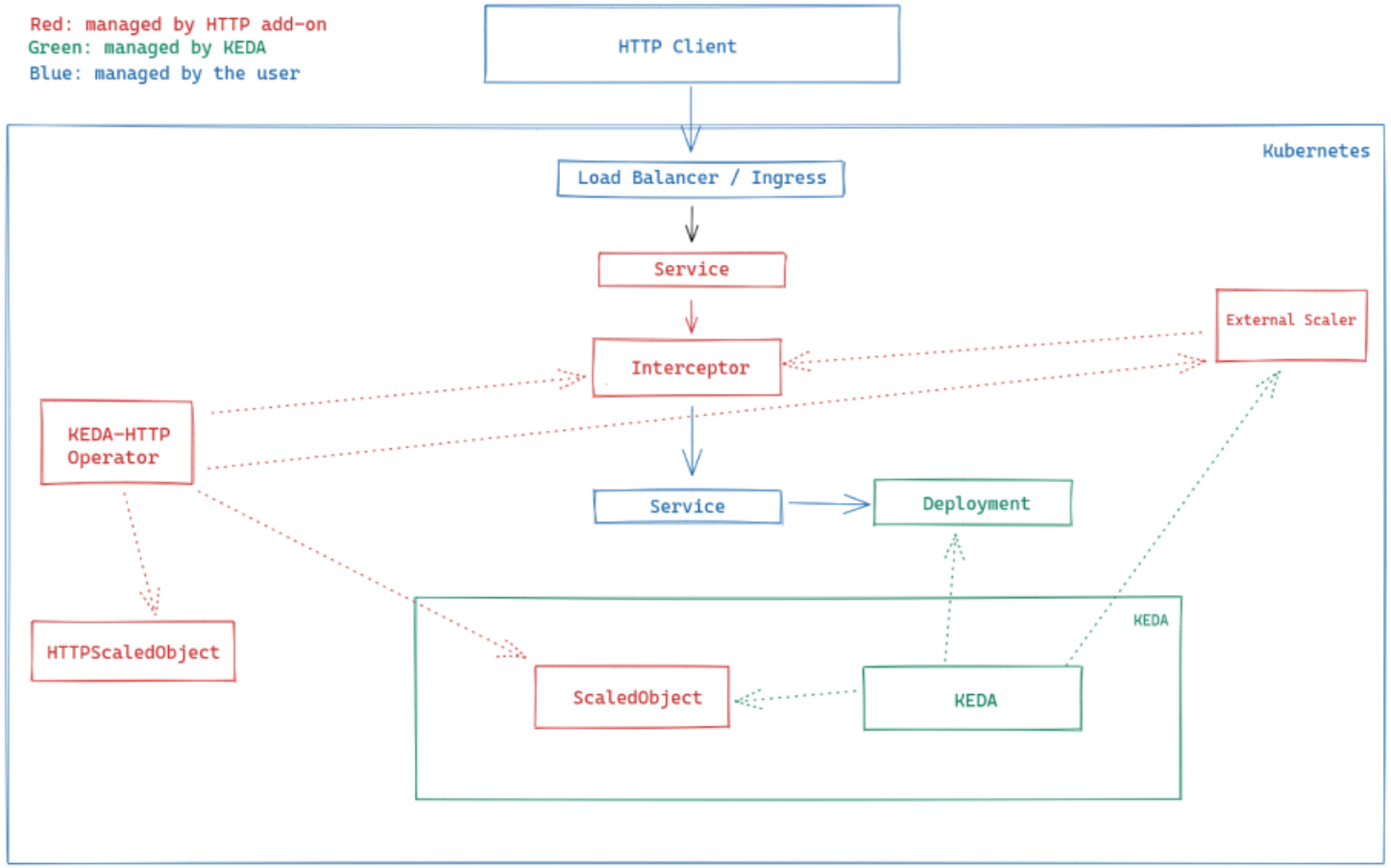
I repeated the tests with Locust and watched the replicas increase as more traffic hits the Nginx Ingress controller!



CAN YOU AUTOSCALE ALL MICROSERVICES ON THE NUMBER OF REQUESTS RECEIVED?



Unless apps expose HTTP metrics, you can't scale them based on traffic.
However, there's a workaround.
KEDA ships with an HTTP add-on to enable HTTP scaling
https://github.com/kedacore/http-add-on



> KEDA injects a sidecar proxy in your pod so that all the HTTP traffic is routed there first.
> Then it measures the number of requests and shares the metrics.
> With that data at hand, you can trigger the autoscaler finally.

That's it! Thanks for reading this far!

During my research, I found a few helpful links and I collected them here:

https://gist.github.com/danielepolencic/b10d94fa8a9cc877a3ccfaa0200ca6f3