Performance Testing Ingress for Internet-Scale Workloads

Alexander Brand, Heptio

Kubernetes Ingress

heptio

- Organizations typically have a portfolio of applications that are offered to end users over the Internet
- Kubernetes supports multiple ways of exposing applications to the outside world
- Ingress is the answer if you are looking for layer 7 load balancing

heptio

heptio
Contour

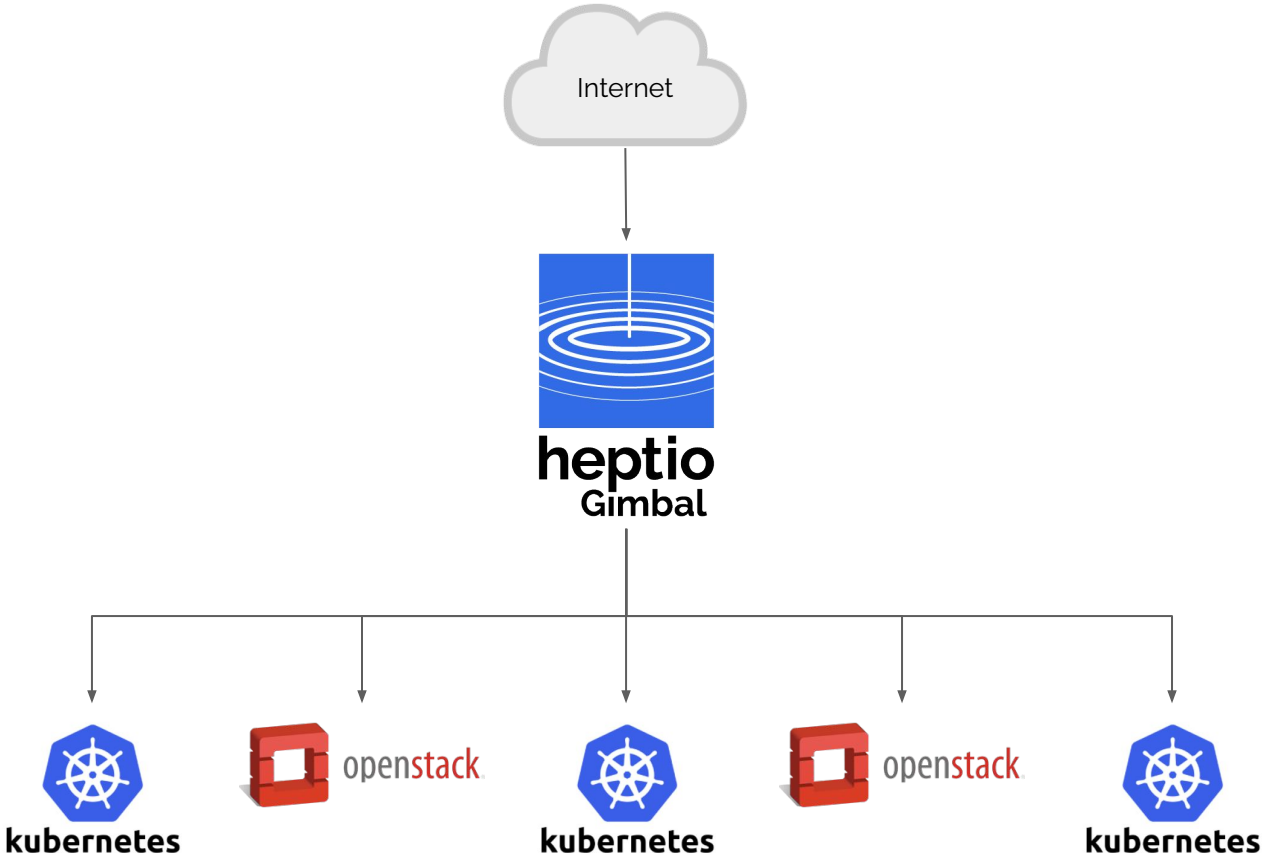
heptio
Gimbal

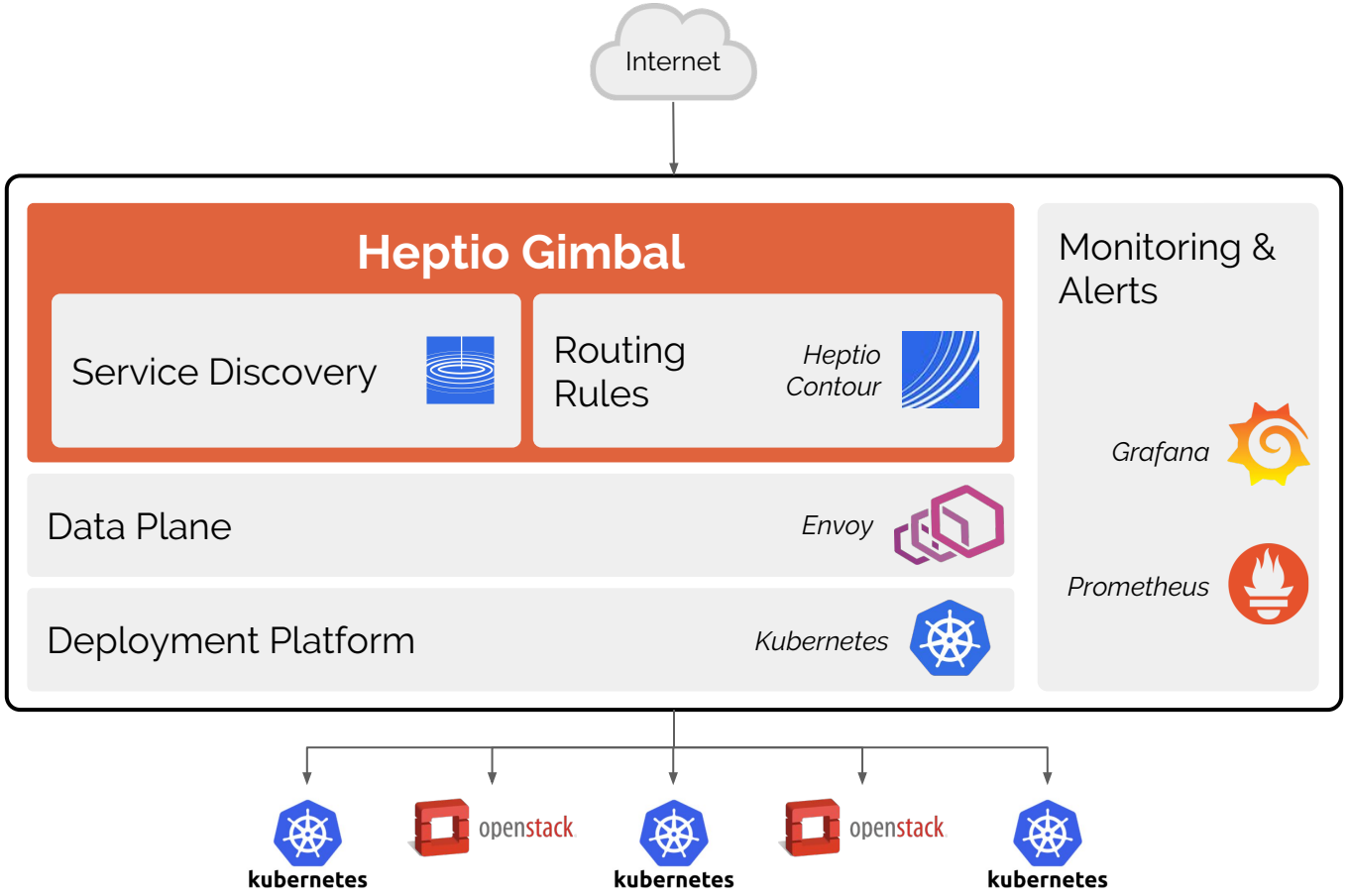
Why did we create Gimbal?

heptio

- Open sourced from co-development project with Yahoo Japan Corporation subsidiary, Actapio
- Operate hundreds of services that are exposed to the Internet
- Run multiple OpenStack and Kubernetes clusters

heptio


Internet
heptio
Gimbal
kubernetes
openstack
kubernetes
openstack
kubernetes


Internet
Heptio Gimbal
Service Discovery
Routing Rules
Heptio Contour
Data Plane
Envoy
Deployment Platform
Kubernetes
Monitoring & Alerts
Grafana
Prometheus
kubernetes
openstack
kubernetes
openstack
kubernetes

Let's talk performance

heptio

Requirements

heptio

- Millions of concurrent connections
- Thousands of services per datacenter
- Tens of thousands of endpoints per datacenter
- < 30ms P99 round-trip time latency

Multiple variables

heptio

- Requests per second
- Concurrent connections
- Response payload size
- Number of services
- Number of endpoints
- Number of Ingress / IngressRoutes
- Number of proxy (Envoy) pods

Multiple Subsystems

heptio

- Data plane
- Control plane
- Discovery system
- Monitoring system

Two pronged approach

heptio

Micro-benchmarks

Understand the impact of a single variable on a specific subsystem.
For example: “Understand the effect of number of concurrent connections on response latency”

Macro-benchmarks

Understand the impact of “realistic” load on the system (or subsystem)
For example: “Understand the effect of 100k CC and 30k RPS on the performance of the system”

Micro-benchmarks

heptio

Example: Concurrent connections vs Latency

heptio

Test Method: Adjust the number of concurrent connections in wrk2, and observe effect on latency

Variable Under Test: # of Concurrent Connections

Test Cases: 10k, 25k, 50k, 100k, 250k, 500k, 1m

Expected: < 30 ms P99 Latency

Micro-benchmarks

heptio

Pros

- Helped us identify bottlenecks/bugs in Contour and the discovery subsystem

panic in contour when there is ×

GitHub, Inc. [US] | https://github.com/heptio/contour/issues/523

Search or jump to... Pull requests Issues Marketplace Explore

heptio / contour

View Repository | Unwatch 55 | Unstar 1,213 | Fork 144

Code | Issues 80 | Pull requests 2 | Insights

panic in contour when there is a high rate of churn of IngressRoute resources #523

Edit | New issue

Closed **alexbrand** opened this issue on Jul 6 · 3 comments

alexbrand commented on Jul 6 Member

While running performance tests for Gimbal, I noticed that Contour was crashing with an index out of range runtime error.

I was able to reproduce the issue using the internal/e2e test framework. The test can be found here: https://github.com/alexbrand/contour/tree/rate-bug

I have seen a couple of different stack traces, but they are all index out of range inside the `route.pb.go` file from envoyproxy/go-control-plane. This is one of them:

```
panic: runtime error: index out of range

goroutine 396 [running]:
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2/route.
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2.(*Rout
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/envoyproxy/go-control-plane/envoy/api/v2.(*Rout
        /Users/abrand/go/src/github.com/heptio/contour/vendor/github.com/envoyproxy/go-cont
github.com/heptio/contour/vendor/github.com/gogo/protobuf/proto.Marshal(0x1cfa860, 0xc42040
```

Assignees
davecheney

Labels
bug
p0 - Hair On Fire

Projects
None yet

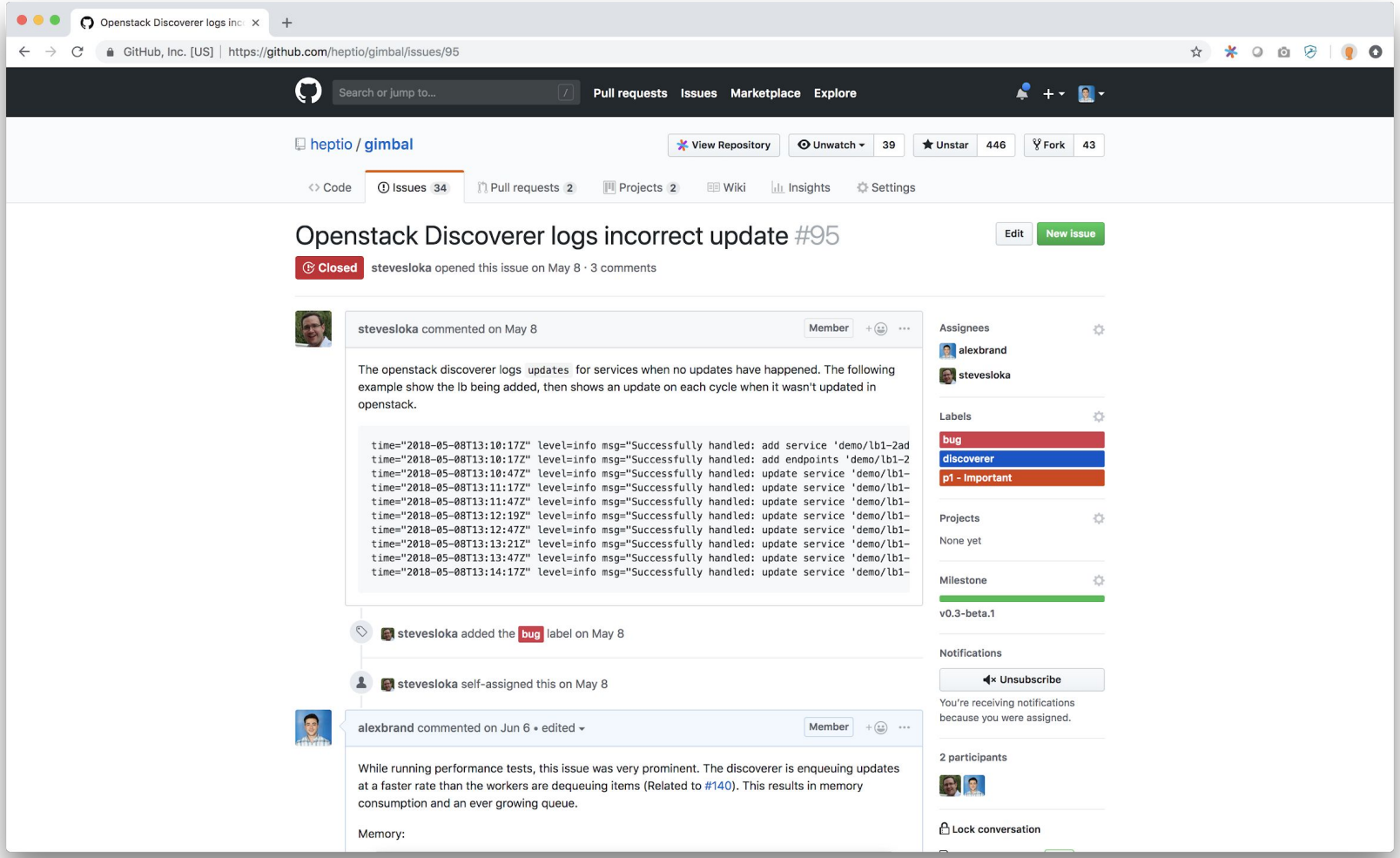
Milestone
0.6.0-beta.1

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

3 participants

Lock conversation

heptio


Openstack Discoverer logs incor
GitHub, Inc. [US] | https://github.com/heptio/gimbal/issues/95
Search or jump to...
Pull requests
Issues
Marketplace
Explore
heptio / gimbal
View Repository
Unwatch 39
Unstar 446
Fork 43
Code
Issues 34
Pull requests 2
Projects 2
Wiki
Insights
Settings
Openstack Discoverer logs incorrect update #95
Edit
New issue
Closed
stevesloka opened this issue on May 8 · 3 comments
stevesloka commented on May 8
Member
The openstack discoverer logs updates for services when no updates have happened. The following example show the lb being added, then shows an update on each cycle when it wasn't updated in openstack.
time="2018-05-08T13:10:17Z" level=info msg="Successfully handled: add service 'demo/lb1-2ad
time="2018-05-08T13:10:17Z" level=info msg="Successfully handled: add endpoints 'demo/lb1-2
time="2018-05-08T13:10:47Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:11:17Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:11:47Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:12:19Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:12:47Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:13:21Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:13:47Z" level=info msg="Successfully handled: update service 'demo/lb1-
time="2018-05-08T13:14:17Z" level=info msg="Successfully handled: update service 'demo/lb1-
stevesloka added the bug label on May 8
stevesloka self-assigned this on May 8
alexbrand commented on Jun 6 • edited
Member
While running performance tests, this issue was very prominent. The discoverer is enqueuing updates at a faster rate than the workers are dequeuing items (Related to #140). This results in memory consumption and an ever growing queue.
Memory:
Assignees
alexbrand
stevesloka
Labels
bug
discoverer
p1 - Important
Projects
None yet
Milestone
v0.3-beta.1
Notifications
Unsubscribe
You're receiving notifications because you were assigned.
2 participants
Lock conversation

Openstack Discoverer logs inc

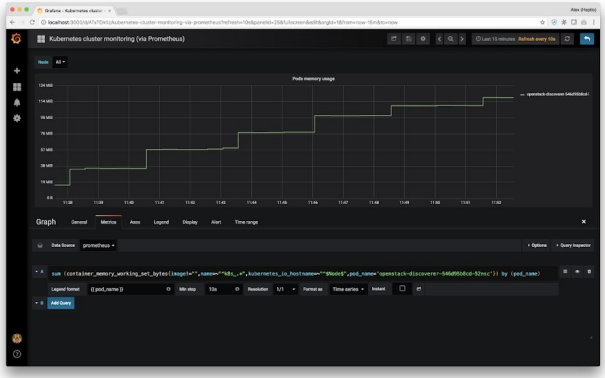
GitHub, Inc. [US] | https://github.com/heptio/gimbal/issues/95

alexbrand commented on Jun 6 • edited

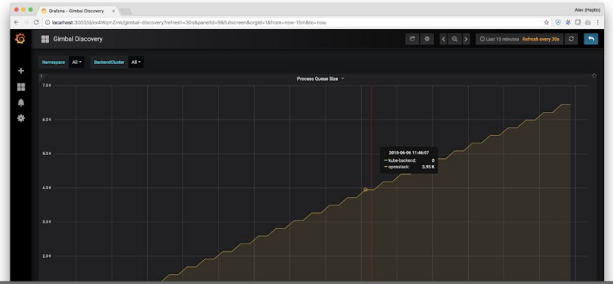
Member

While running performance tests, this issue was very prominent. The discoverer is enqueuing updates at a faster rate than the workers are dequeuing items (Related to #140). This results in memory consumption and an ever growing queue.

Memory:



Queue:



You're receiving notifications because you were assigned.

2 participants

Lock conversation

Transfer this issue Beta

heptio

Micro-benchmarks

heptio

Pros

- Helped us identify bottlenecks/bugs in Contour and the discovery subsystem
- Gave us confidence that the control plane (Contour) could handle a large number of Services, Endpoints, Ingress and IngressRoute

Micro-benchmarks

heptio

Pros

- Helped us identify bottlenecks/bugs in Contour and the discovery subsystem
- Gave us confidence that the control plane (Contour) could handle a large number of Services, Endpoints, Ingress and IngressRoute
- Time is your friend - Less time needed to setup and run

Micro-benchmarks

heptio

Cons

- Evaluates the system through a narrow lens

Micro-benchmarks

heptio

Cons

- Evaluates the system through a narrow lens
- Doesn't necessarily reflect real world usage

heptio

Macro-benchmarks

Macro-benchmarks

heptio

- Test the system under “realistic” load
- Measure and evaluate multiple metrics
- Gives you an idea of where the bottlenecks are, and how the system should be scaled to handle more load
- Depending on hardware availability, budget, etc, you might have to scale the test down

Our approach

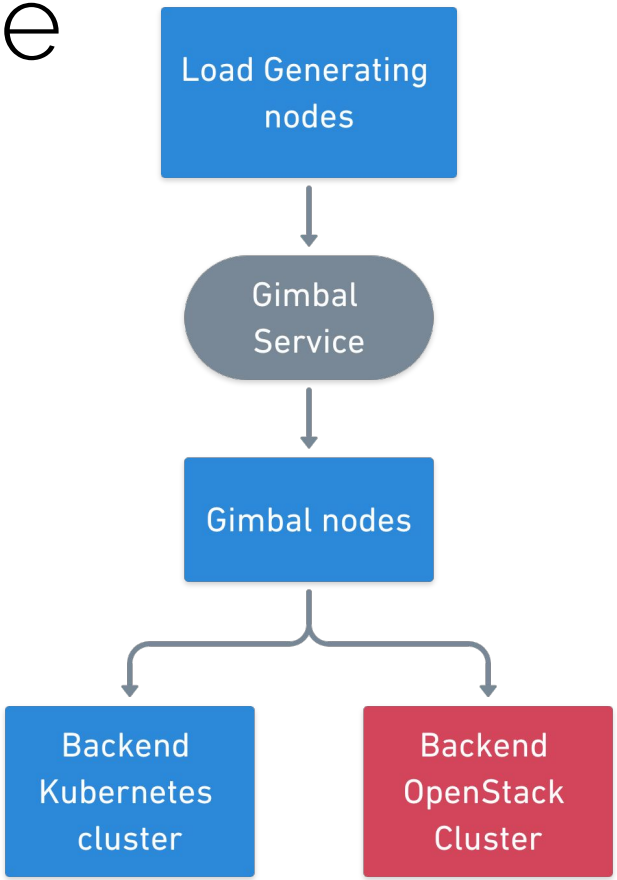
heptio

Run tests at three different scales and ensure resource utilization scales linearly

100k CC & 10k RPS	200k CC & 10k RPS	300k CC & 10k RPS
100k CC & 20k RPS	200k CC & 20k RPS	300k CC & 20k RPS
100k CC & 30k RPS	200k CC & 30k RPS	300k CC & 30k RPS

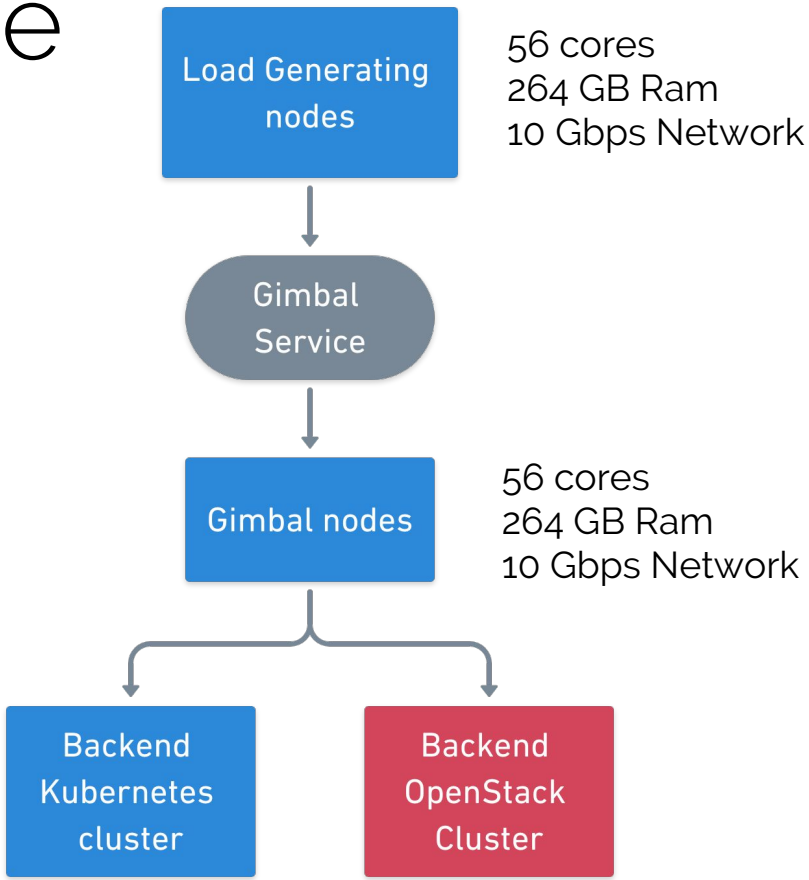
CC = concurrent connections
RPS = requests per second

Infrastructure

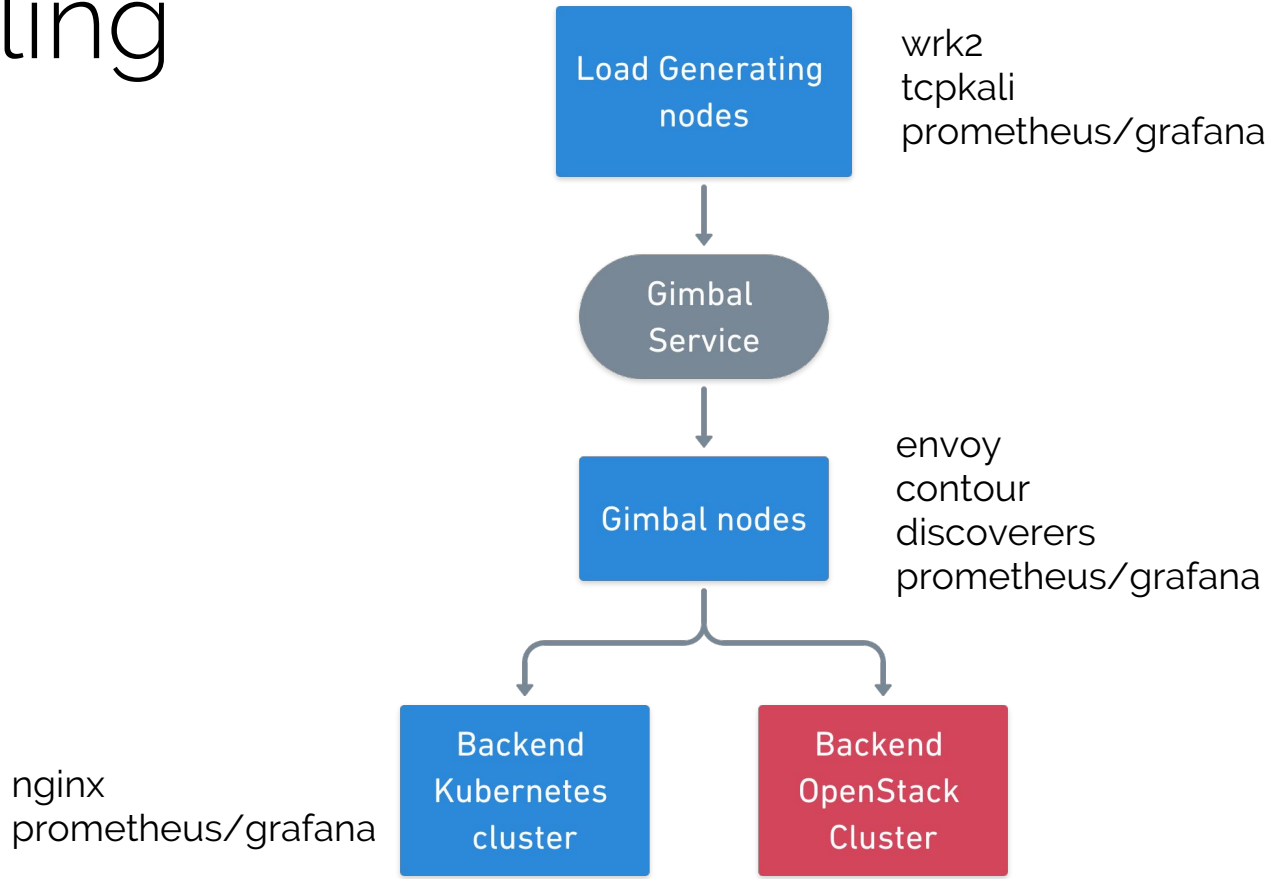


heptio

Infrastructure



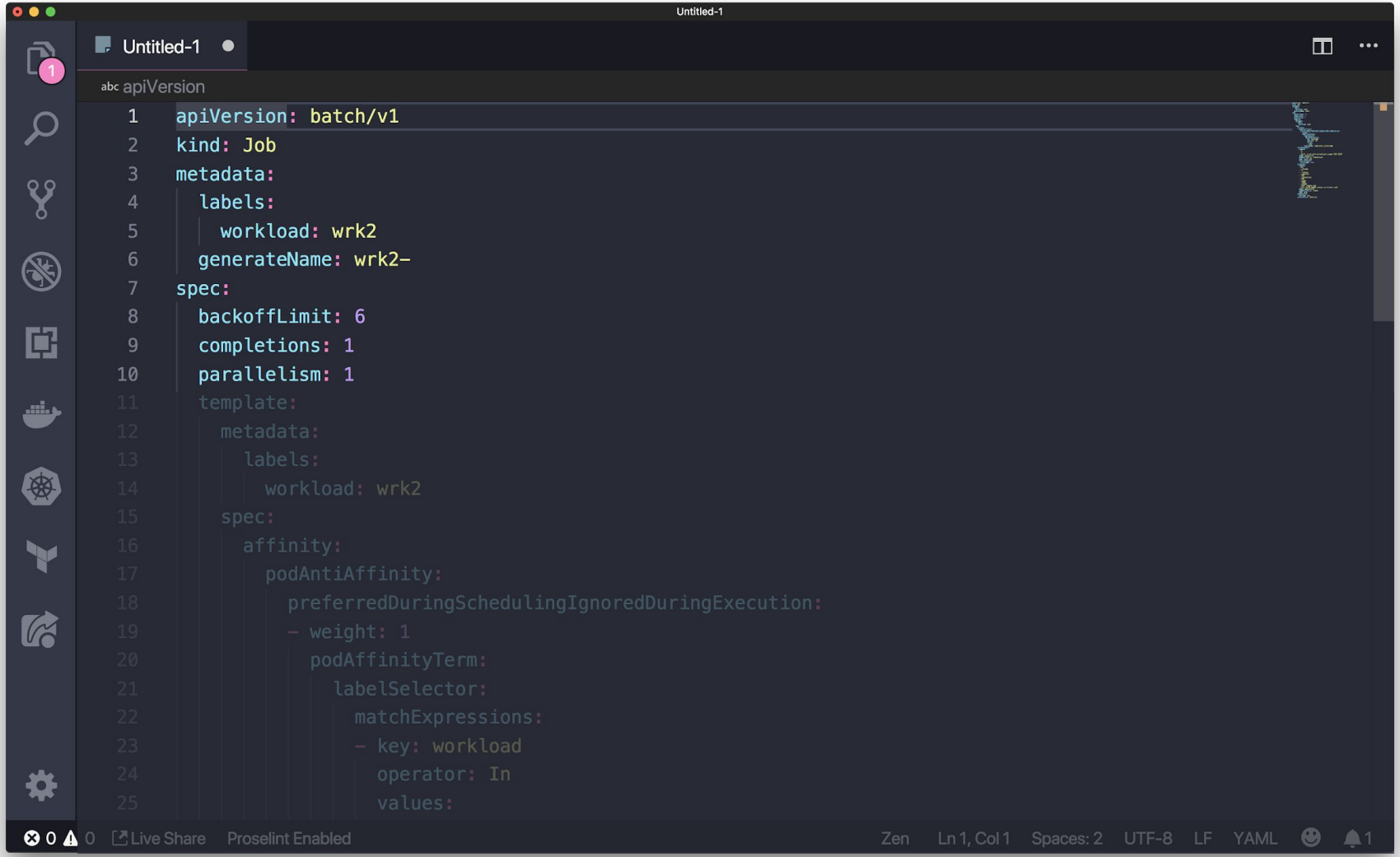
Tooling

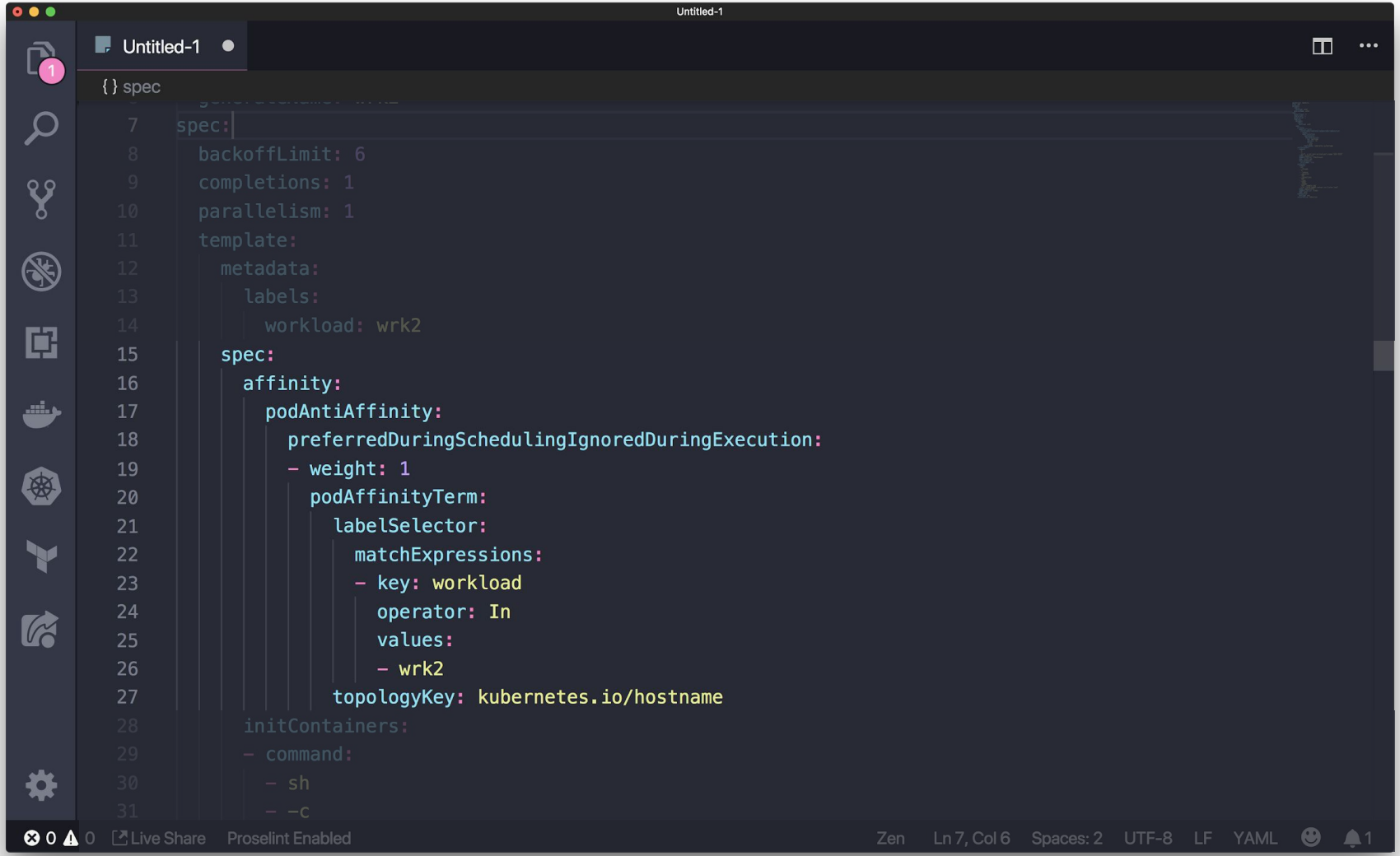

Load Generating nodes
wrk2
tcpkali
prometheus/grafana
Gimbal Service
Gimbal nodes
envoy
contour
discoverers
prometheus/grafana
Backend Kubernetes cluster
Backend OpenStack Cluster
nginx
prometheus/grafana

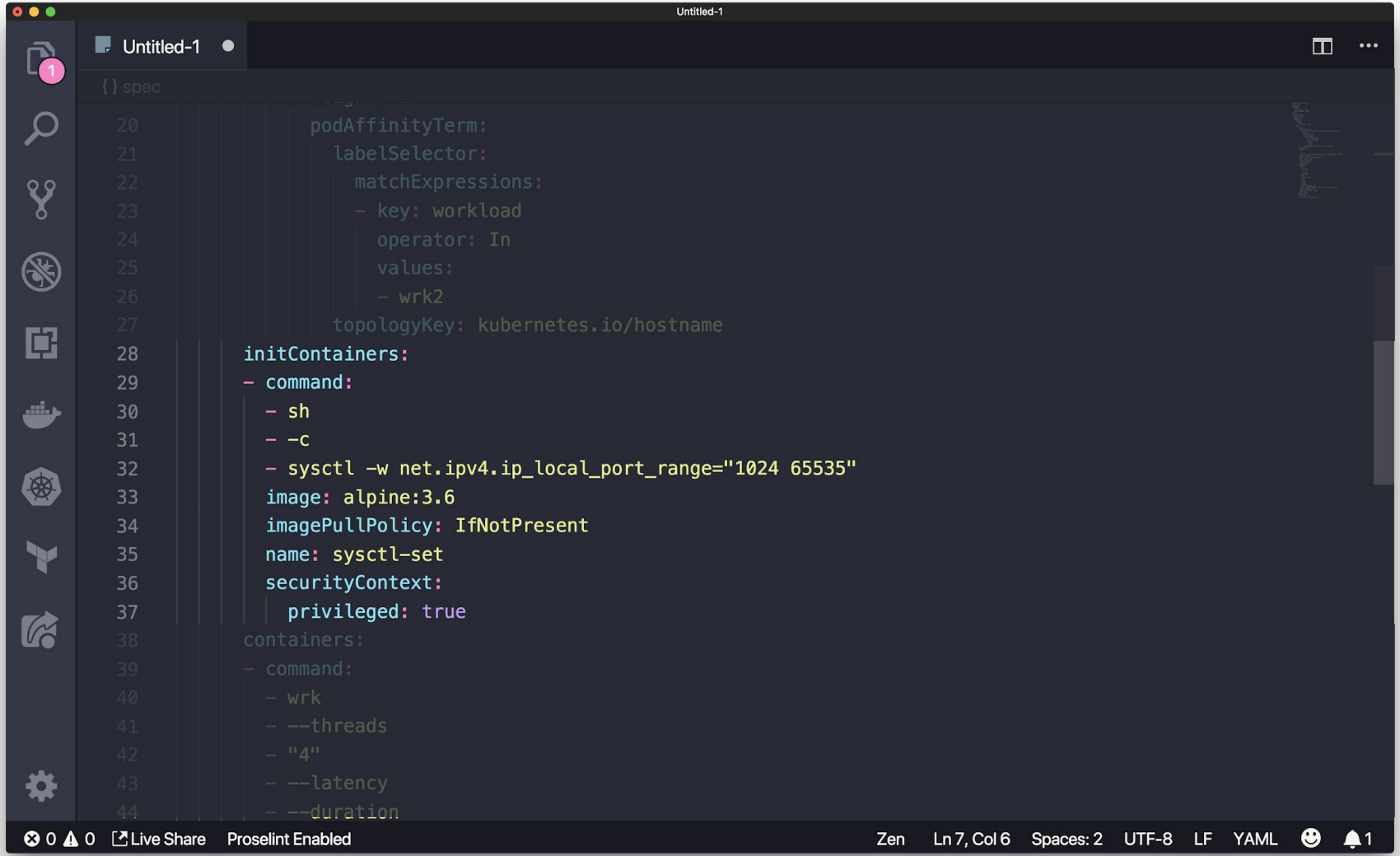
wrk2

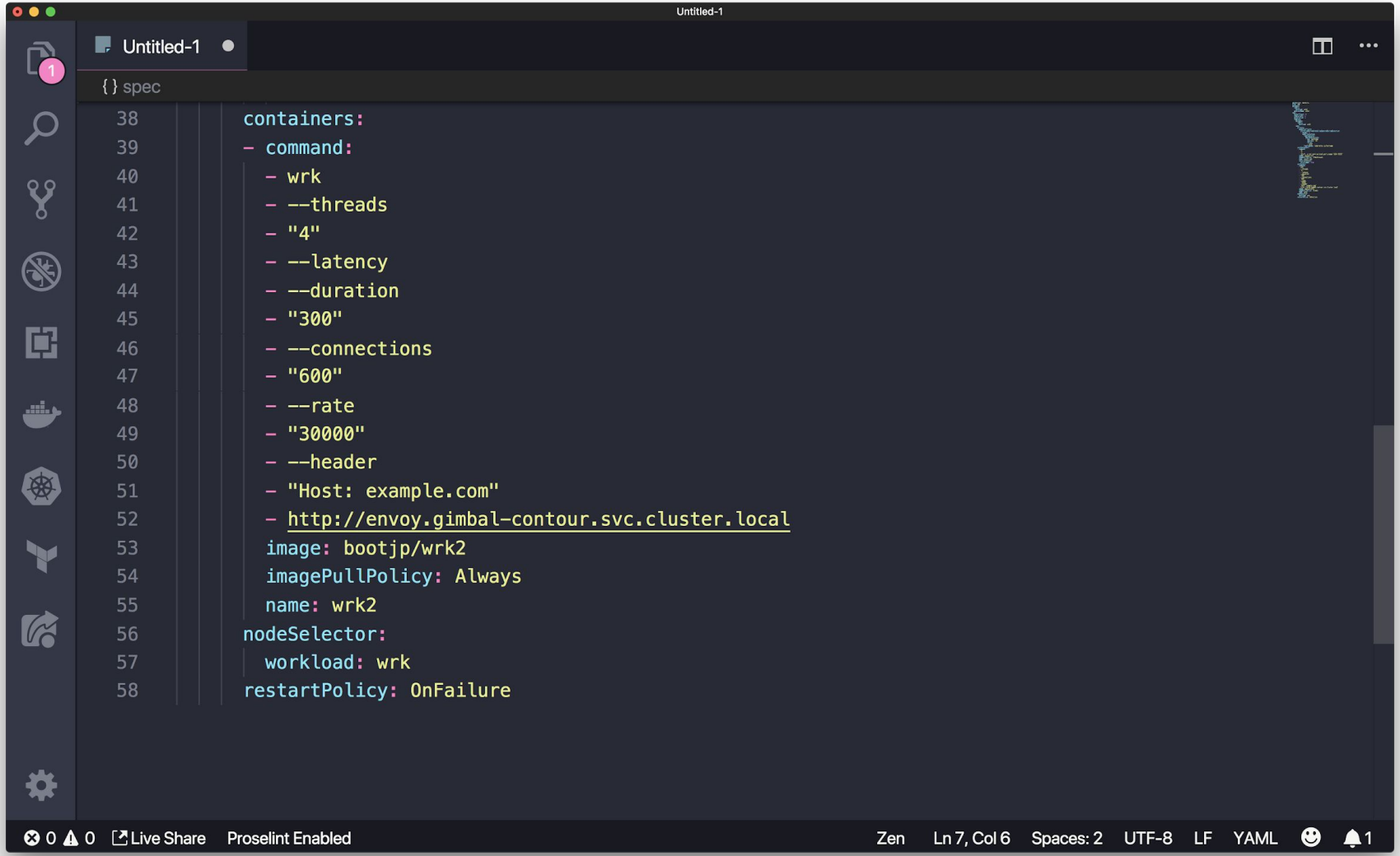
heptio

- HTTP benchmarking tool with accurate latency measurements
- Can generate significant load from a single, multi-core machine
- Deployed as a Kubernetes Job
- https://github.com/giltene/wrk2


Untitled-1
Untitled-1
abc apiVersion
apiVersion: batch/v1
kind: Job
metadata:
labels:
workload: wrk2
generateName: wrk2-
spec:
backoffLimit: 6
completions: 1
parallelism: 1
template:
metadata:
labels:
workload: wrk2
spec:
affinity:
podAntiAffinity:
preferredDuringSchedulingIgnoredDuringExecution:
- weight: 1
podAffinityTerm:
labelSelector:
matchExpressions:
- key: workload
operator: In
values:
0 0
Live Share
Proselint Enabled
Zen
Ln 1, Col 1
Spaces: 2
UTF-8
LF
YAML
1


Untitled-1
Untitled-1
{} spec
spec:
backoffLimit: 6
completions: 1
parallelism: 1
template:
metadata:
labels:
workload: wrk2
spec:
affinity:
podAntiAffinity:
preferredDuringSchedulingIgnoredDuringExecution:
- weight: 1
podAffinityTerm:
labelSelector:
matchExpressions:
- key: workload
operator: In
values:
- wrk2
topologyKey: kubernetes.io/hostname
initContainers:
- command:
- sh
- -c
0 0
Live Share
Proselint Enabled
Zen
Ln 7, Col 6
Spaces: 2
UTF-8
LF
YAML
1

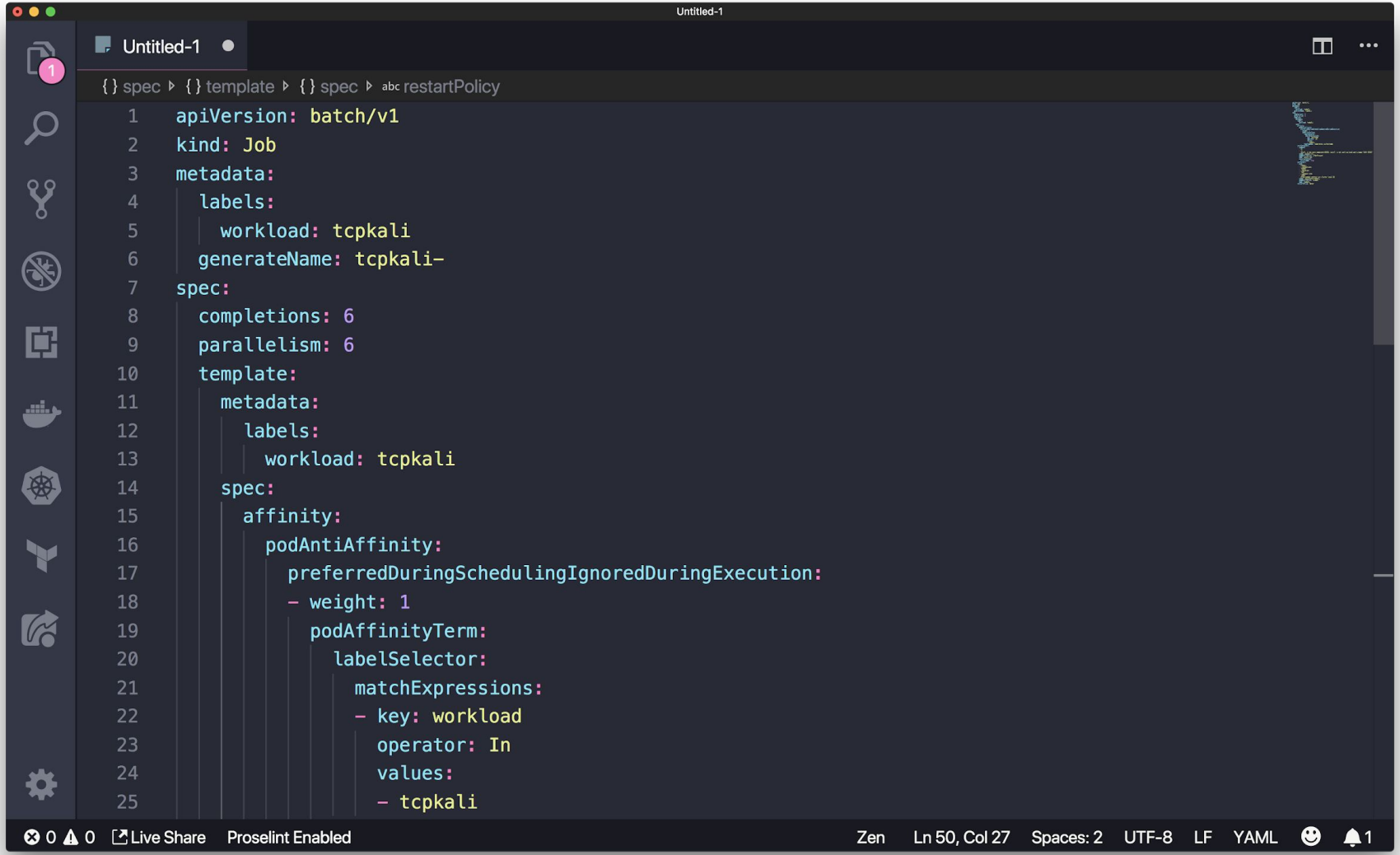

Untitled-1
Untitled-1
{} spec
podAffinityTerm:
labelSelector:
matchExpressions:
- key: workload
operator: In
values:
- wrk2
topologyKey: kubernetes.io/hostname
initContainers:
- command:
- sh
- -c
- sysctl -w net.ipv4.ip_local_port_range="1024 65535"
image: alpine:3.6
imagePullPolicy: IfNotPresent
name: sysctl-set
securityContext:
privileged: true
containers:
- command:
- wrk
- --threads
- "4"
- --latency
- --duration
0 0
Live Share
Proselint Enabled
Zen
Ln 7, Col 6
Spaces: 2
UTF-8
LF
YAML
1

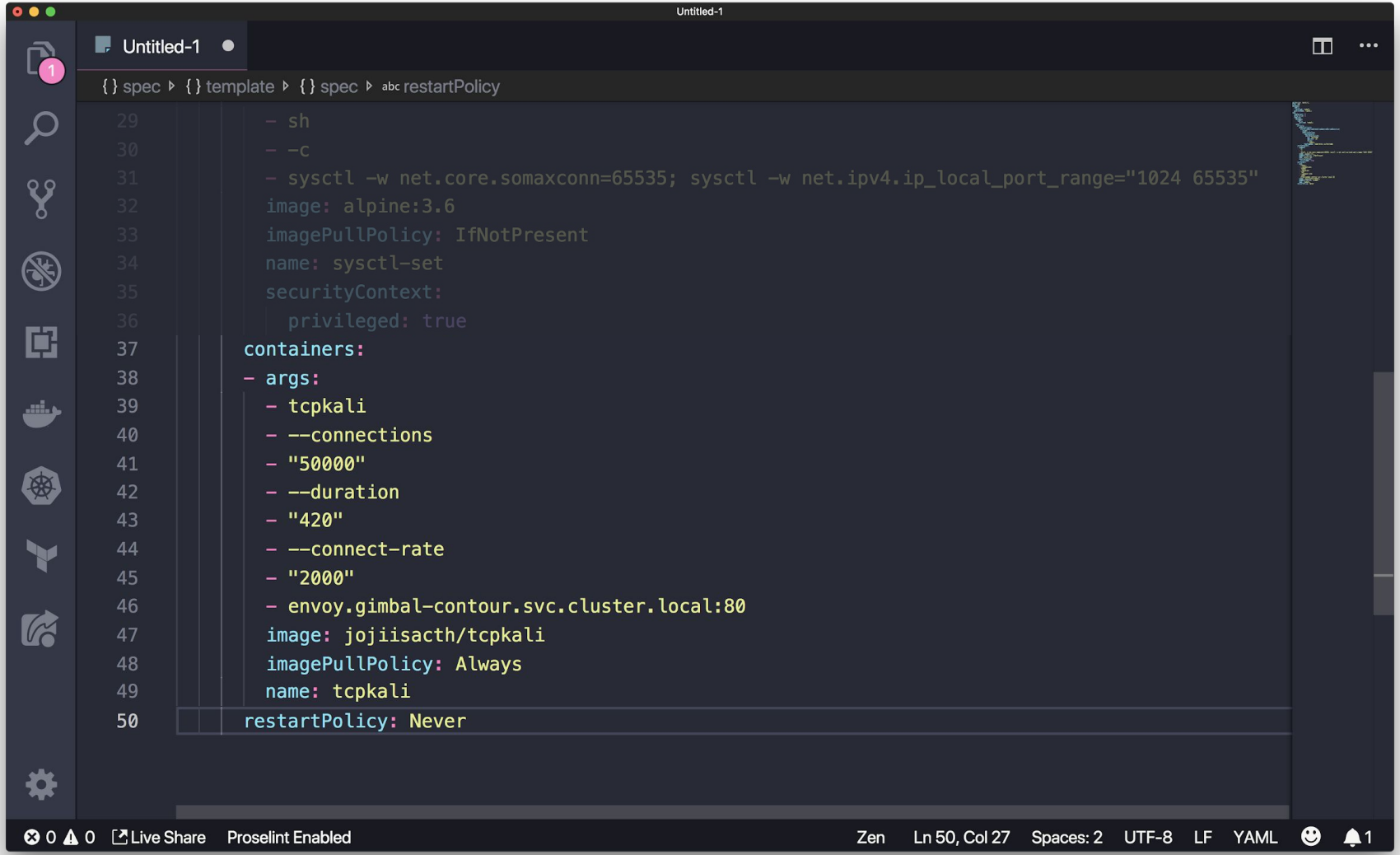

Untitled-1
Untitled-1
{} spec
containers:
- command:
- wrk
- --threads
- "4"
- --latency
- --duration
- "300"
- --connections
- "600"
- --rate
- "30000"
- --header
- "Host: example.com"
- http://envoy.gimbal-contour.svc.cluster.local
image: bootjp/wrk2
imagePullPolicy: Always
name: wrk2
nodeSelector:
workload: wrk
restartPolicy: OnFailure
0 0
Live Share
Proselint Enabled
Zen
Ln 7, Col 6
Spaces: 2
UTF-8
LF
YAML
1

tcpkali

heptio

- TCP load generator used to open thousands of connections
- Opens connections and keeps them open
- Much better at opening connections than wrk2
- Deployed as a Kubernetes Job
- https://github.com/satori-com/tcpkali

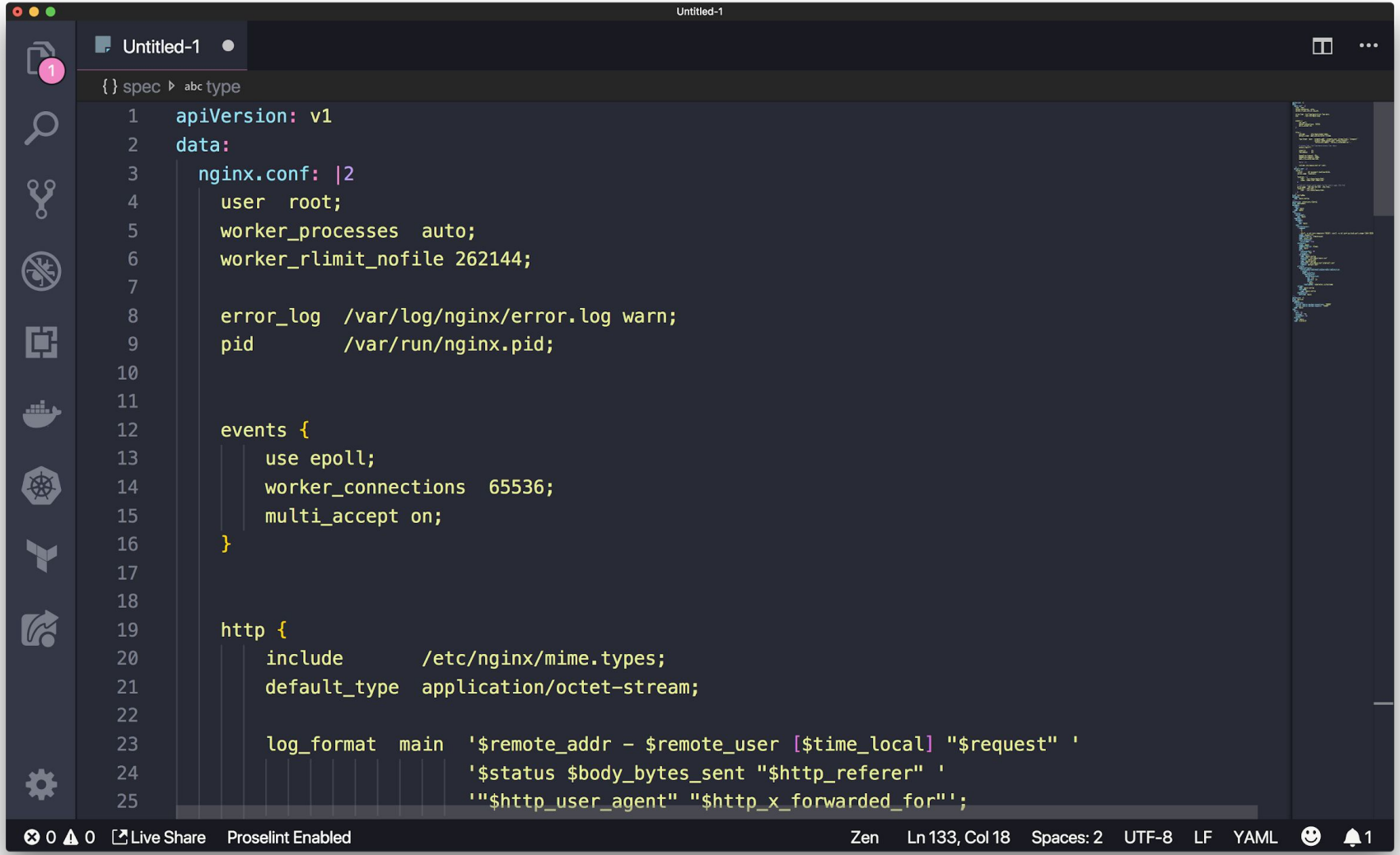

Untitled-1
Untitled-1
{ } spec ▸ { } template ▸ { } spec ▸ abc restartPolicy
1 apiVersion: batch/v1
2 kind: Job
3 metadata:
4 labels:
5 workload: tcpkali
6 generateName: tcpkali-
7 spec:
8 completions: 6
9 parallelism: 6
10 template:
11 metadata:
12 labels:
13 workload: tcpkali
14 spec:
15 affinity:
16 podAntiAffinity:
17 preferredDuringSchedulingIgnoredDuringExecution:
18 - weight: 1
19 podAffinityTerm:
20 labelSelector:
21 matchExpressions:
22 - key: workload
23 operator: In
24 values:
25 - tcpkali
0 0 Live Share Proselint Enabled
Zen Ln 50, Col 27 Spaces: 2 UTF-8 LF YAML 1


Untitled-1
Untitled-1
{ } spec ▸ { } template ▸ { } spec ▸ abc restartPolicy
- sh
- -c
- sysctl -w net.core.somaxconn=65535; sysctl -w net.ipv4.ip_local_port_range="1024 65535"
image: alpine:3.6
imagePullPolicy: IfNotPresent
name: sysctl-set
securityContext:
privileged: true
containers:
- args:
- tcpkali
- --connections
- "50000"
- --duration
- "420"
- --connect-rate
- "2000"
- envoy.gimbal-contour.svc.cluster.local:80
image: jojiisacth/tcpkali
imagePullPolicy: Always
name: tcpkali
restartPolicy: Never
0 0 Live Share Proselint Enabled
Zen Ln 50, Col 27 Spaces: 2 UTF-8 LF YAML 1

Nginx

heptio

- Used nginx as the target service
- Ran as a Kubernetes Deployment
- Ran tests against two variants: “vanilla” (600 bytes) and custom (22 kilobytes)
- Default configuration was inadequate for our load test

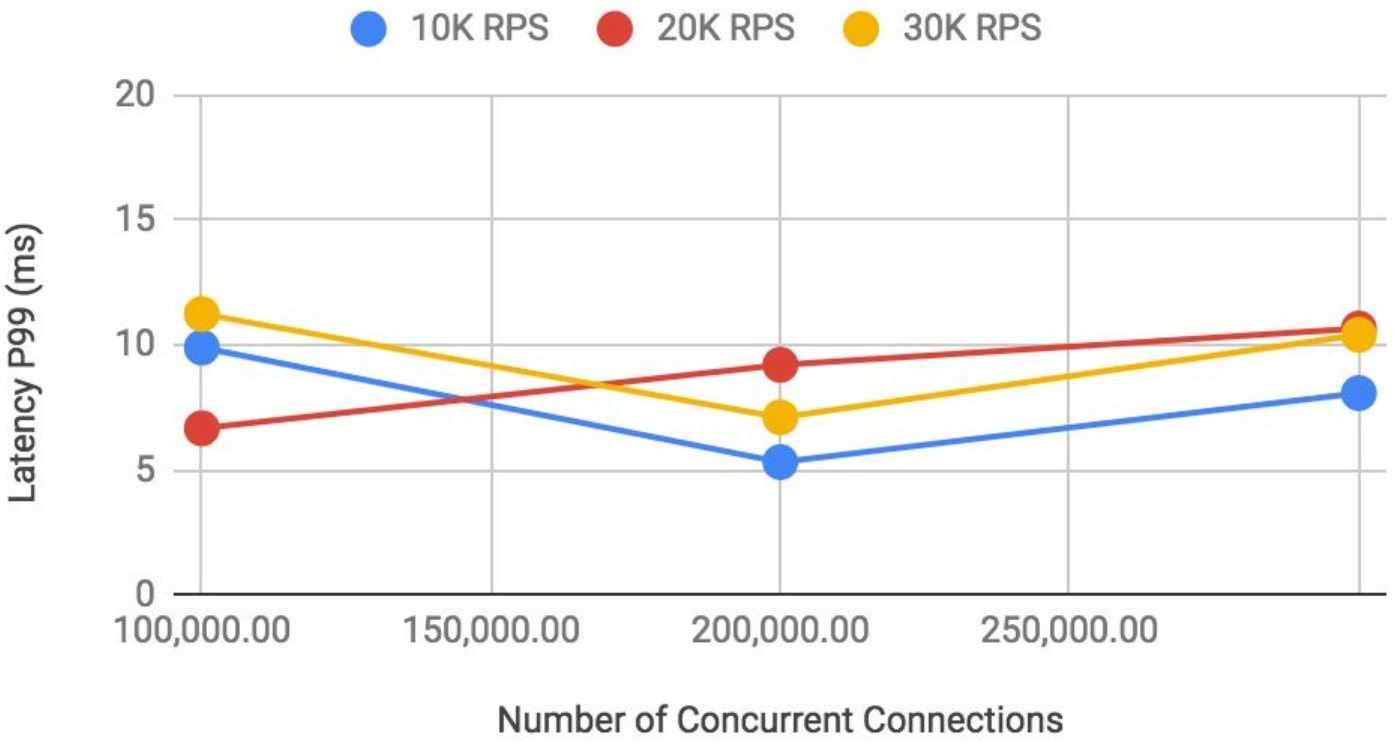

Untitled-1
Untitled-1
{ } spec ▸ abc type
apiVersion: v1
data:
nginx.conf: |2
user root;
worker_processes auto;
worker_rlimit_nofile 262144;
error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;
events {
use epoll;
worker_connections 65536;
multi_accept on;
}
http {
include /etc/nginx/mime.types;
default_type application/octet-stream;
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"';
0 0 Live Share Proselint Enabled
Zen Ln 133, Col 18 Spaces: 2 UTF-8 LF YAML 1

heptio

Results

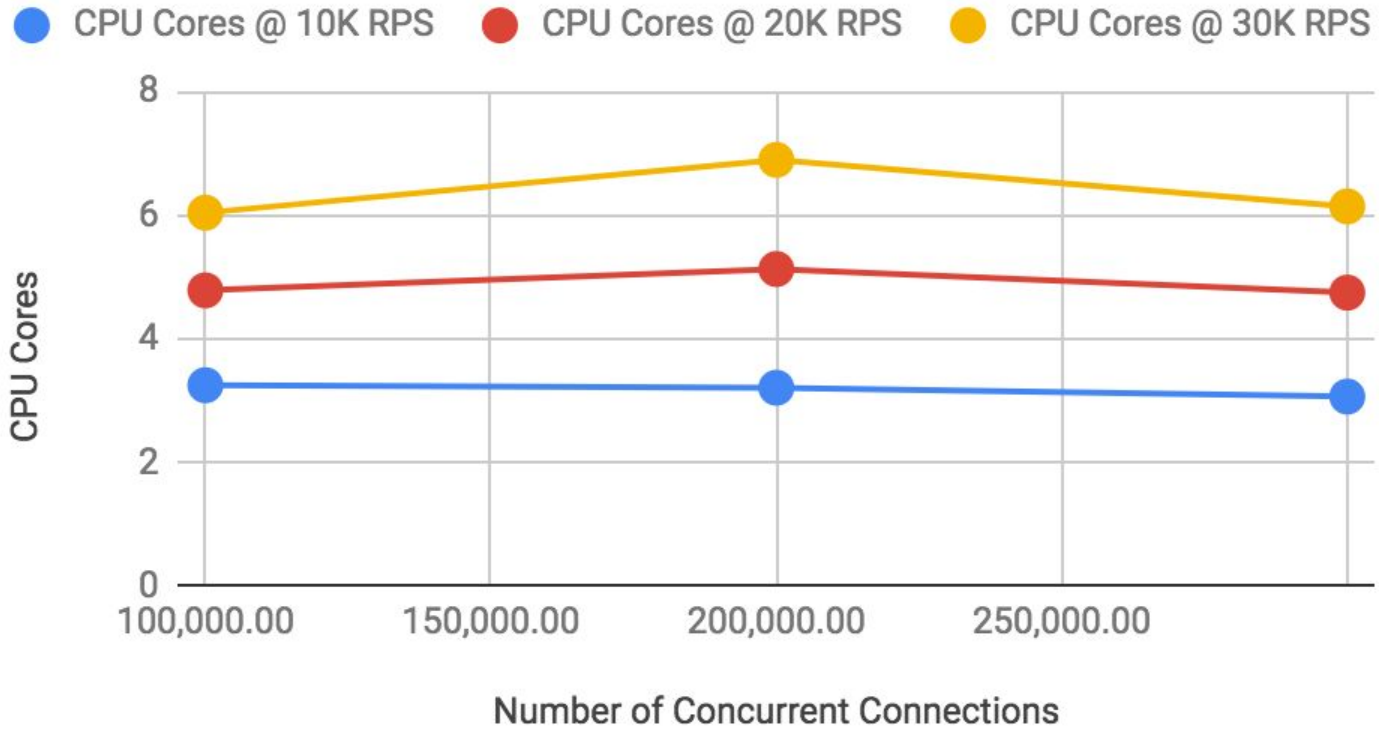
heptio

Gimbal: Impact of Concurrency & RPS on Latency



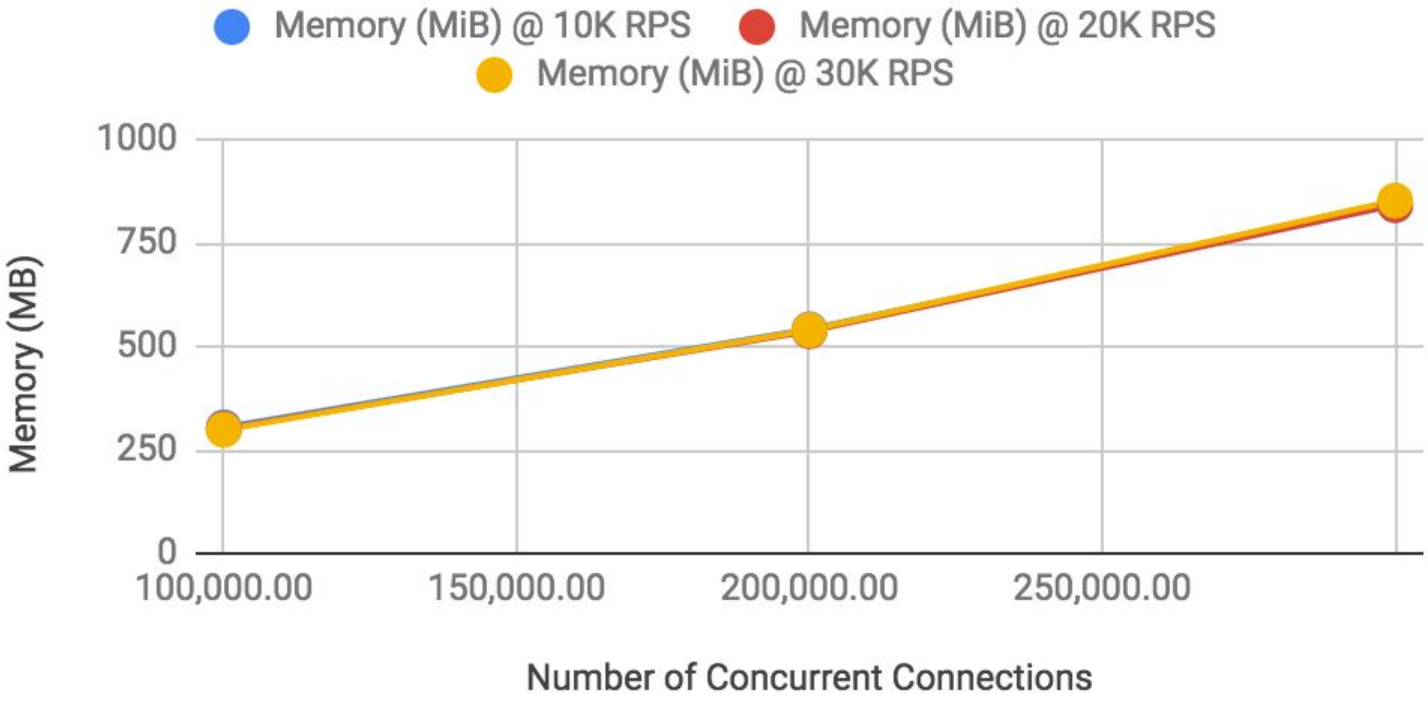
heptio

Gimbal: Impact of Concurrency & RPS on CPU



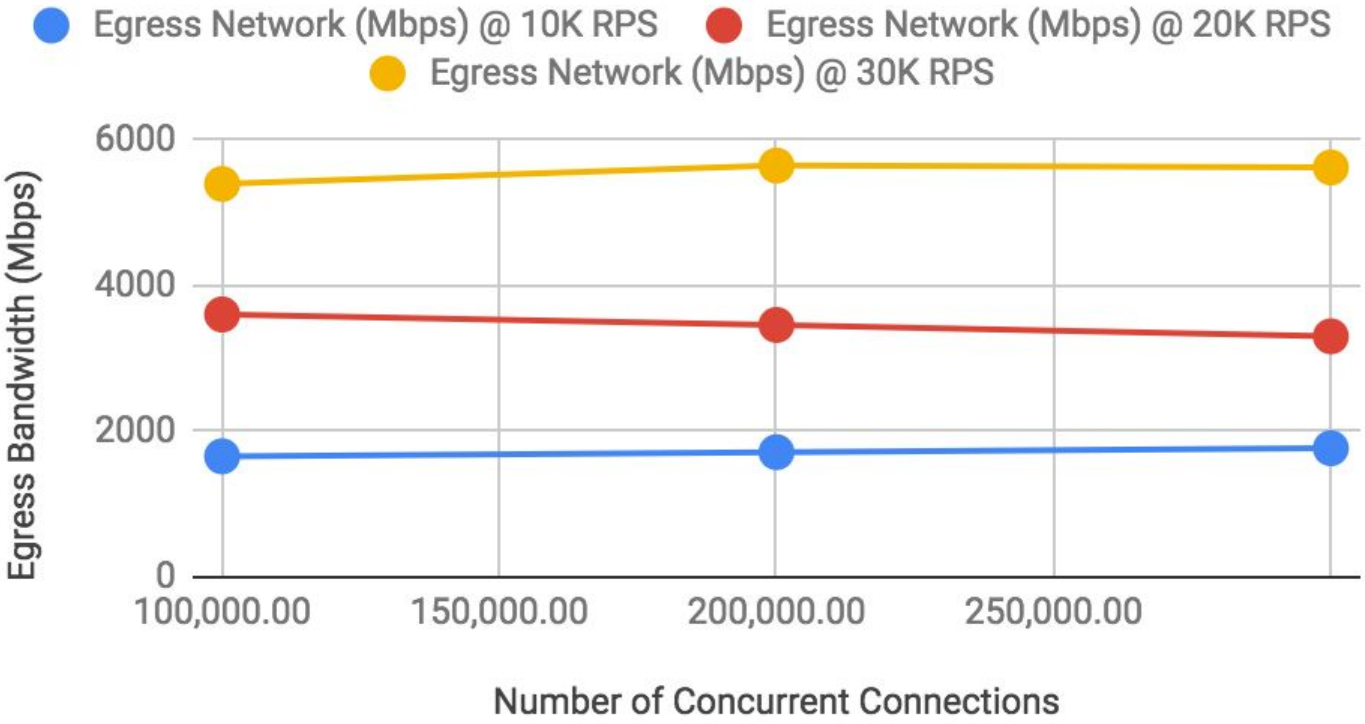
heptio

Gimbal: Impact of Concurrency & RPS on Memory

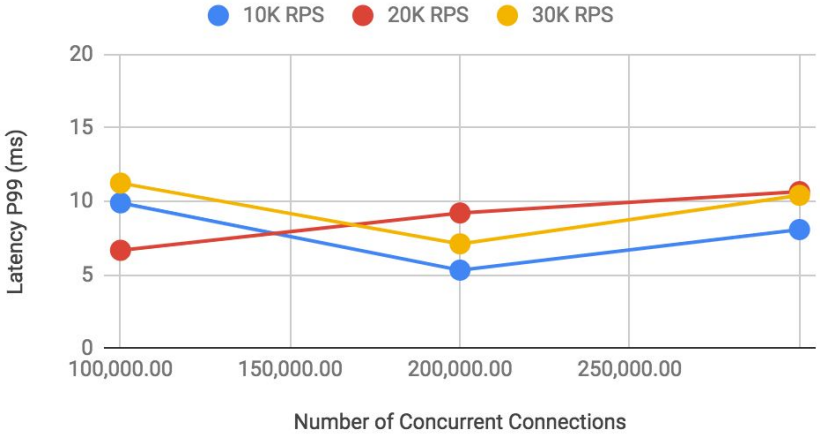


heptio

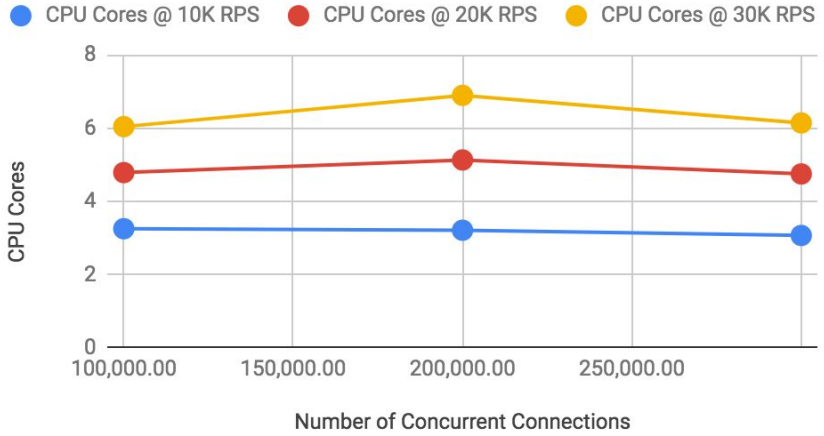
Gimbal: Impact of Concurrency & RPS on Network



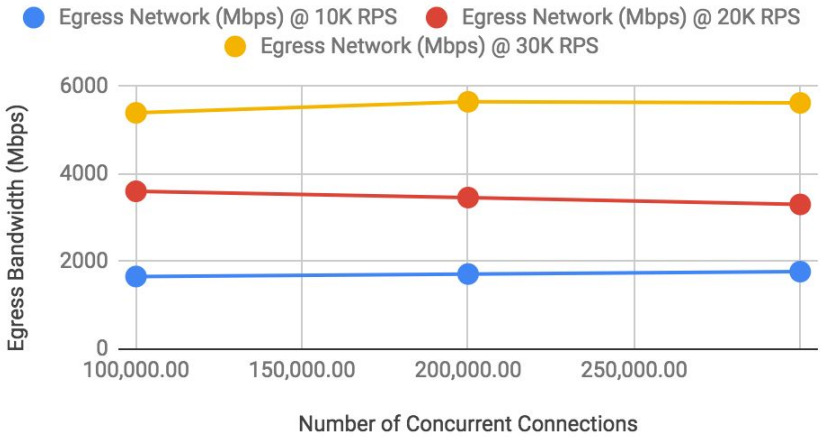
Gimbal: Impact of Concurrency & RPS on Latency



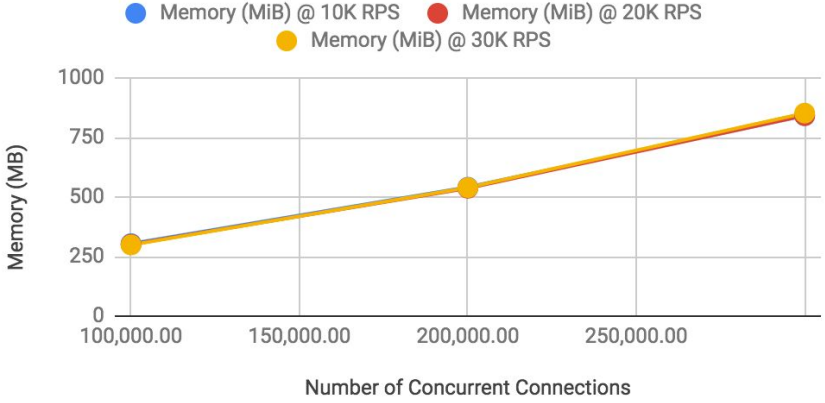
Gimbal: Impact of Concurrency & RPS on CPU

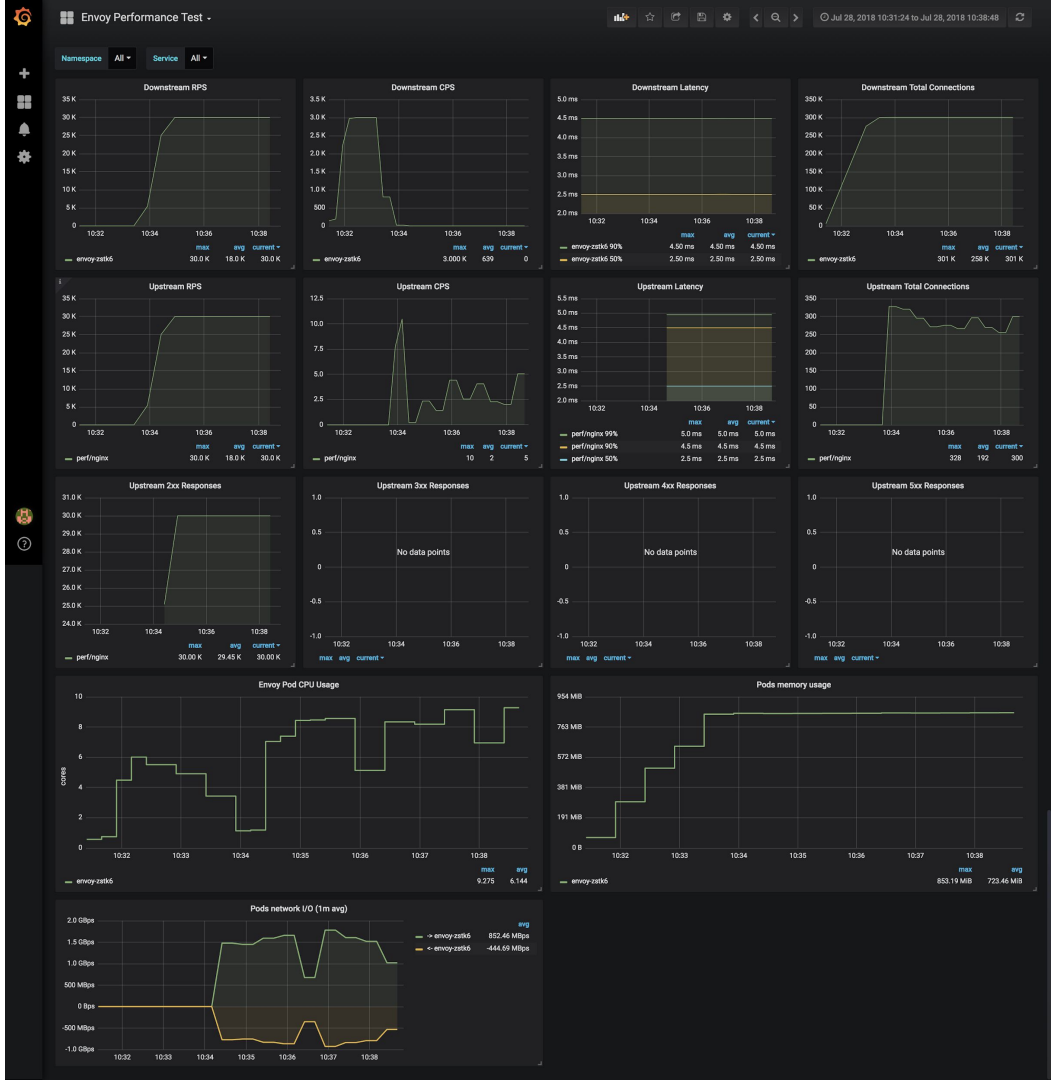


Gimbal: Impact of Concurrency & RPS on Network



Gimbal: Impact of Concurrency & RPS on Memory




Envoy Performance Test
Jul 28, 2018 10:31:24 to Jul 28, 2018 10:38:48
Namespace All
Service All
Downstream RPS
Downstream CPS
Downstream Latency
Downstream Total Connections
Upstream RPS
Upstream CPS
Upstream Latency
Upstream Total Connections
Upstream 2xx Responses
Upstream 3xx Responses
Upstream 4xx Responses
Upstream 5xx Responses
No data points
Envoy Pod CPU Usage
Pods memory usage
Pods network I/O (1m avg)

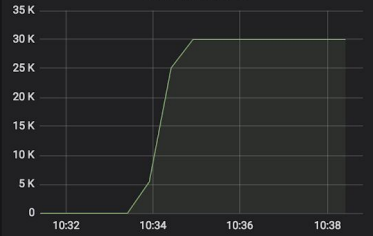
heptio

Envoy Performance Test

Jul 28, 2018 10:31:24 to Jul 28, 2018 10:38:48

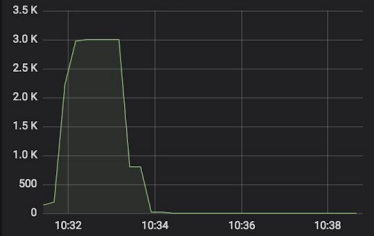
Namespace: All | Service: All

Downstream RPS



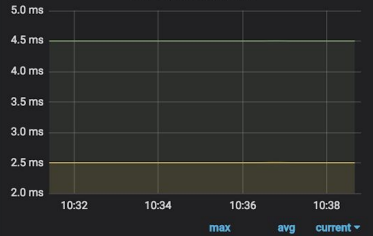
	max	avg	current
envoy-zstk6	30.0 K	18.0 K	30.0 K

Downstream CPS



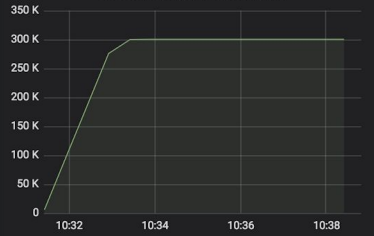
	max	avg	current
envoy-zstk6	3.000 K	639	0

Downstream Latency



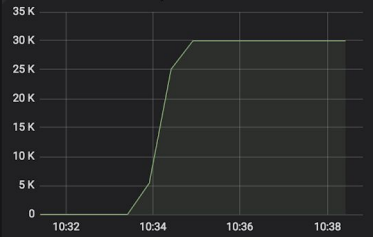
	max	avg	current
envoy-zstk6 90%	4.50 ms	4.50 ms	4.50 ms
envoy-zstk6 50%	2.50 ms	2.50 ms	2.50 ms

Downstream Total Connections



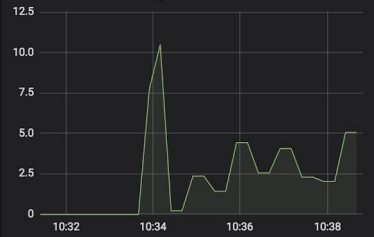
	max	avg	current
envoy-zstk6	301 K	258 K	301 K

Upstream RPS



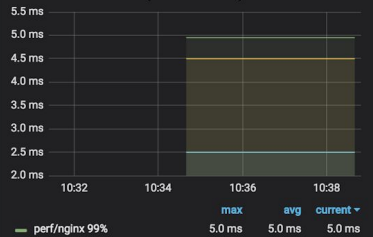
	max	avg	current
perf/nginx	30.0 K	18.0 K	30.0 K

Upstream CPS



	max	avg	current
perf/nginx	10	2	5

Upstream Latency

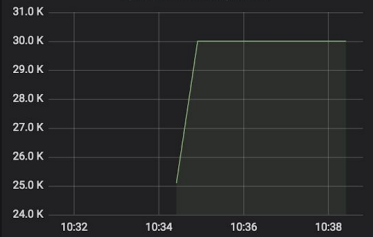


	max	avg	current
perf/nginx 99%	5.0 ms	5.0 ms	5.0 ms
perf/nginx 90%	4.5 ms	4.5 ms	4.5 ms
perf/nginx 50%	2.5 ms	2.5 ms	2.5 ms

Upstream Total Connections

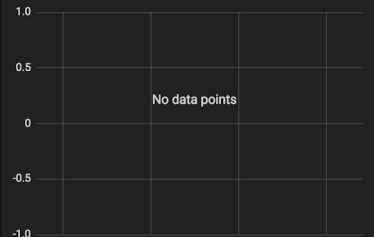
	max	avg	current
perf/nginx	328	192	300

Upstream 2xx Responses



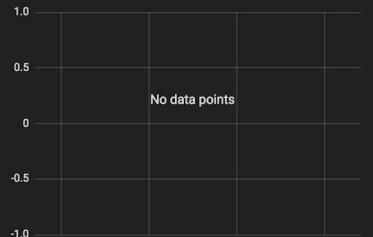
	max	avg	current
perf/nginx	30.00 K	29.45 K	30.00 K

Upstream 3xx Responses



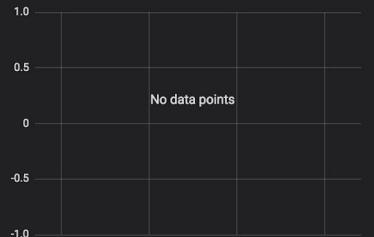
max avg current

Upstream 4xx Responses

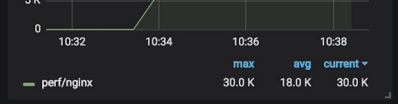


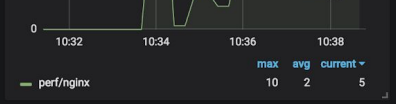
max avg current

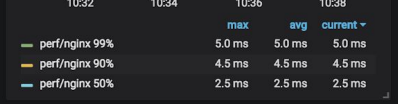
Upstream 5xx Responses

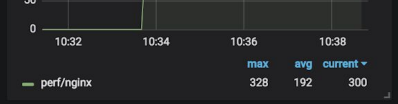


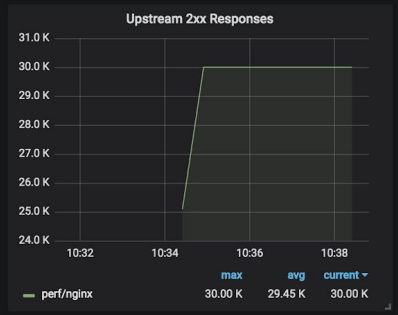
max avg current


10:32
10:34
10:36
10:38
max avg current
perf/nginx 30.0 K 18.0 K 30.0 K


10:32
10:34
10:36
10:38
max avg current
perf/nginx 10 2 5

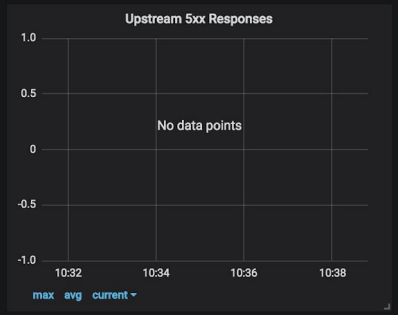

max avg current
perf/nginx 99% 5.0 ms 5.0 ms 5.0 ms
perf/nginx 90% 4.5 ms 4.5 ms 4.5 ms
perf/nginx 50% 2.5 ms 2.5 ms 2.5 ms

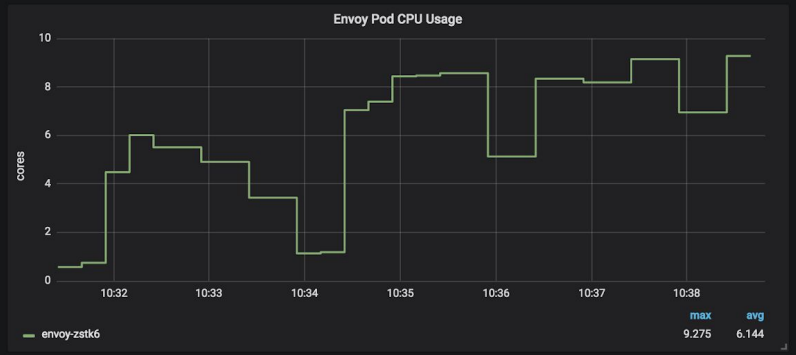

10:32
10:34
10:36
10:38
max avg current
perf/nginx 328 192 300

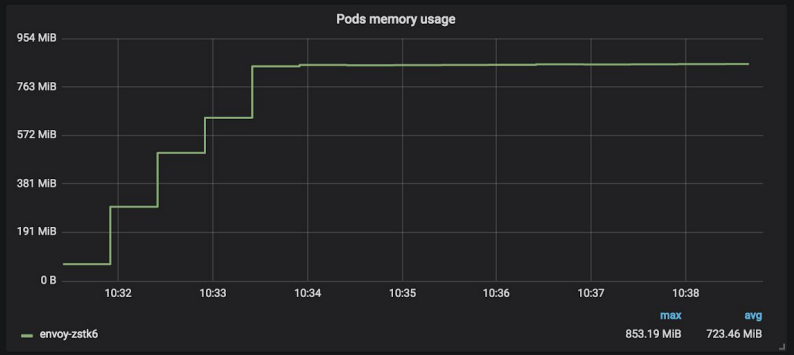

Upstream 2xx Responses
31.0 K
30.0 K
29.0 K
28.0 K
27.0 K
26.0 K
25.0 K
24.0 K
10:32
10:34
10:36
10:38
max avg current
perf/nginx 30.00 K 29.45 K 30.00 K

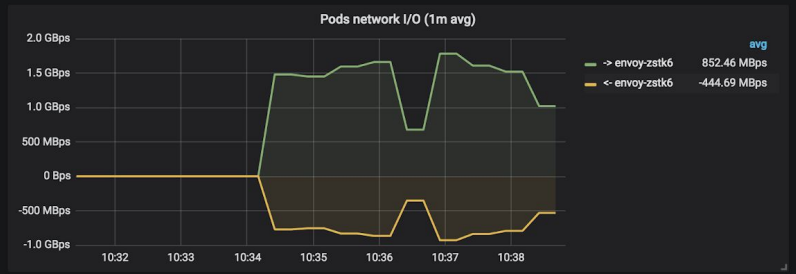

Upstream 3xx Responses
1.0
0.5
0
-0.5
-1.0
No data points
10:32
10:34
10:36
10:38
max avg current


Upstream 4xx Responses
1.0
0.5
0
-0.5
-1.0
No data points
10:32
10:34
10:36
10:38
max avg current


Upstream 5xx Responses
1.0
0.5
0
-0.5
-1.0
No data points
10:32
10:34
10:36
10:38
max avg current


Envoy Pod CPU Usage
cores
10
8
6
4
2
0
10:32
10:33
10:34
10:35
10:36
10:37
10:38
max avg
envoy-zstk6 9.275 6.144


Pods memory usage
954 MiB
763 MiB
572 MiB
381 MiB
191 MiB
0 B
10:32
10:33
10:34
10:35
10:36
10:37
10:38
max avg
envoy-zstk6 853.19 MiB 723.46 MiB


Pods network I/O (1m avg)
2.0 GBps
1.5 GBps
1.0 GBps
500 MBps
0 Bps
-500 MBps
-1.0 GBps
10:32
10:33
10:34
10:35
10:36
10:37
10:38
avg
-> envoy-zstk6 852.46 MBps
<- envoy-zstk6 -444.69 MBps

Lessons Learned

Document everything

heptio

- Create a plan that outlines what and how you are going to test
- Create a results table (or document) to capture the numbers you care about
- Document the environment's characteristics and specifications
- Keep a journal or scratchpad while you are running tests

Observability is paramount

heptio

- Prometheus and Grafana proved to be indispensable
- Envoy, Contour, Gimbal discoverers all produce useful metrics
- Node-level visibility via Prometheus node_exporter
- Create test-specific dashboards
- Metrics are key to understand the system under test
- Don’t fly blind

Short tests can be deceptive

heptio

- Test should be in the order of minutes instead of seconds
- Allows all components in the test path to warm up
- Reduces network jitter over a long measurement period
- Prometheus can obtain a larger set of data points

Check the network

heptio

- Understand your network's capacity before running any tests
- Keep the capacity in mind when designing test cases
- The network will limit your testing if the pipes are overfilled
- For example, we observed very different performance when running on AWS vs bare-metal lab
- Use iperf3 to measure network bandwidth

Tweak the kernel

heptio

- The kernel can get in your way
- System and kernel logs can be helpful
- Can use init containers in Kubernetes

Timebox rabbit holes

heptio

- Weird things will happen at scale
- Some might be one-offs, some might be actual issues
- Take a note of what happened
- Investigate, but make sure to set a timebox on it

Understand first, automate later

heptio

- Resist temptation to automate everything from the get-go
- The test plan or strategy might change along the way
- Once the strategy is solid, document it
- Creating end-to-end automation might not be worth it

Happy testing!

heptio/gimbal | heptio/contour

@alexbrand