RED HAT FORUM 2018 ZURICH

redhat.

PAAS: ARE YOU READY FOR PRODUCTION?

PHILIPPE BÜRGISSER | SENIOR TECHNICAL CONSULTANT

acceleris
cloudifying IT

About me

Philippe Bürgisser

Senior Technical Consultant at Acceleris
Red Hat Certified Architect level II

Focuses

RED HAT® OPENSHIFT Container Platform

ANSIBLE

openstack™

About Acceleris – Facts & Figures

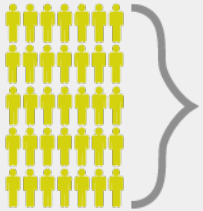
ACCELERIS AG

Bern
Renens
Zürich

ACCELERIS SRL

Bucarest

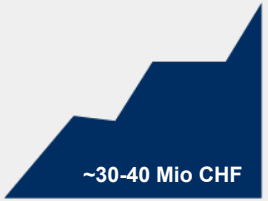
1 OWNER



60

SWITZERLAND & ROMANIA

EMPLOYEES



REVENUE

About Acceleris - Solutions & SERVICES

CONSULTING

INFRASTRUCTURE SERVICES

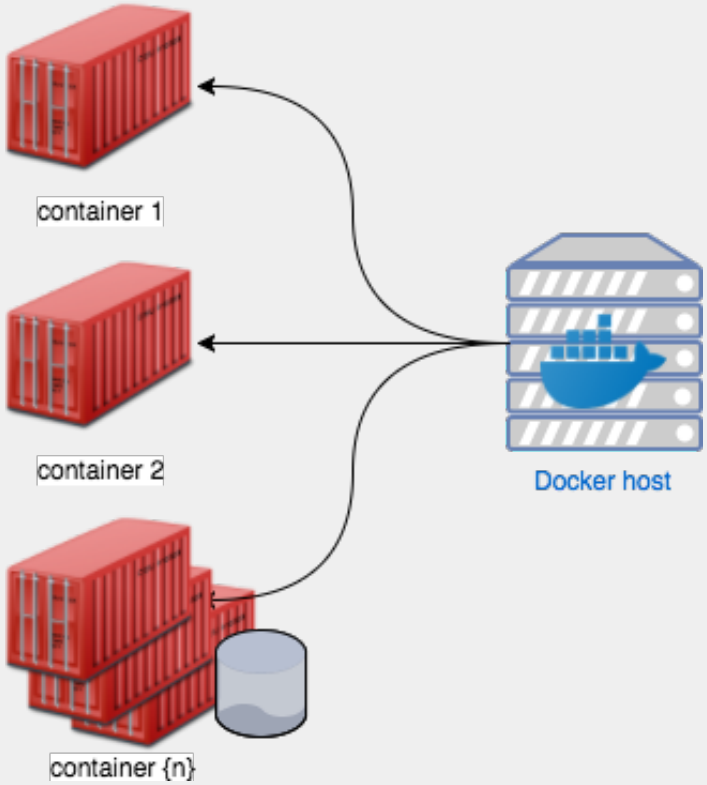
CLOUD & IT OPERATIONS

About Acceleris and Red Hat

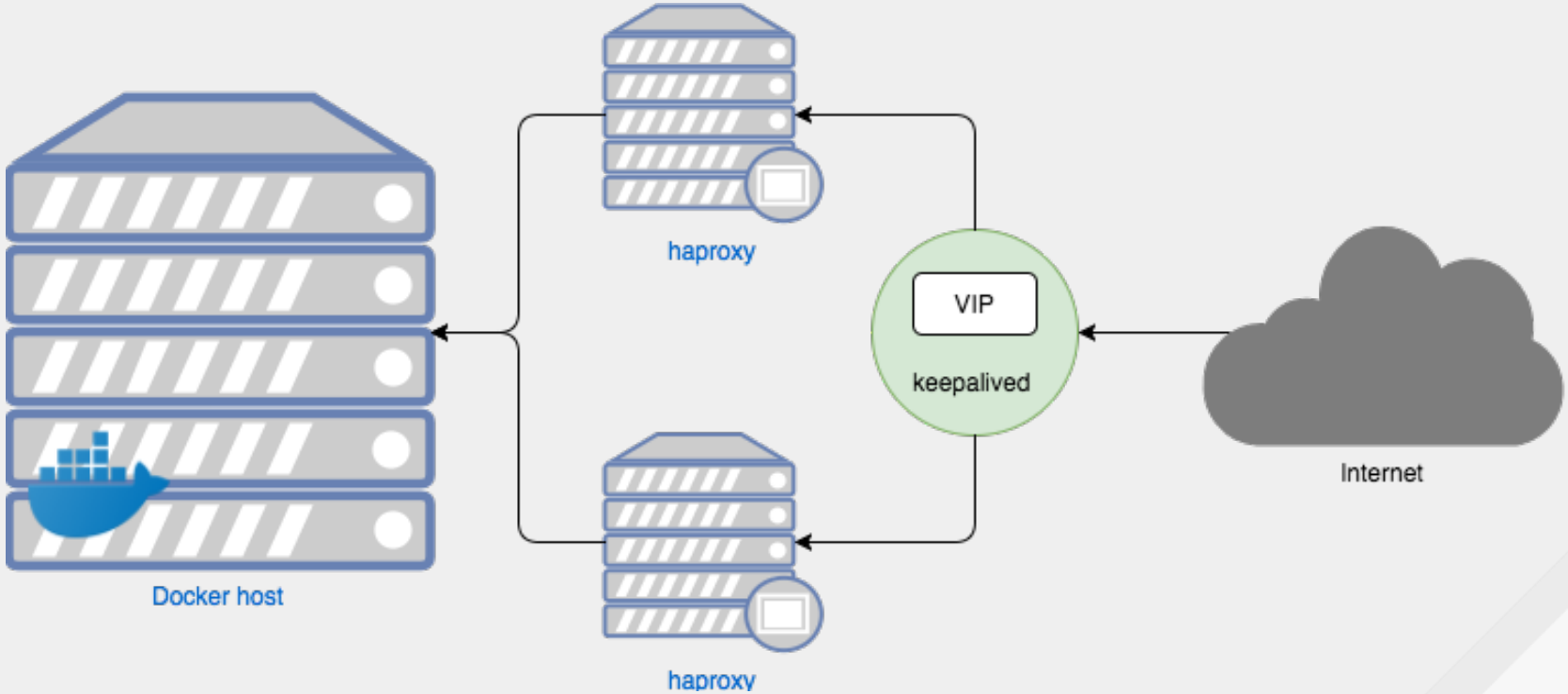
- Top-selling partner of Switzerland
- First Premier Partner of Switzerland

- OpenShift
- Satellite
- RHEV
- OpenStack
- Ceph
- ...

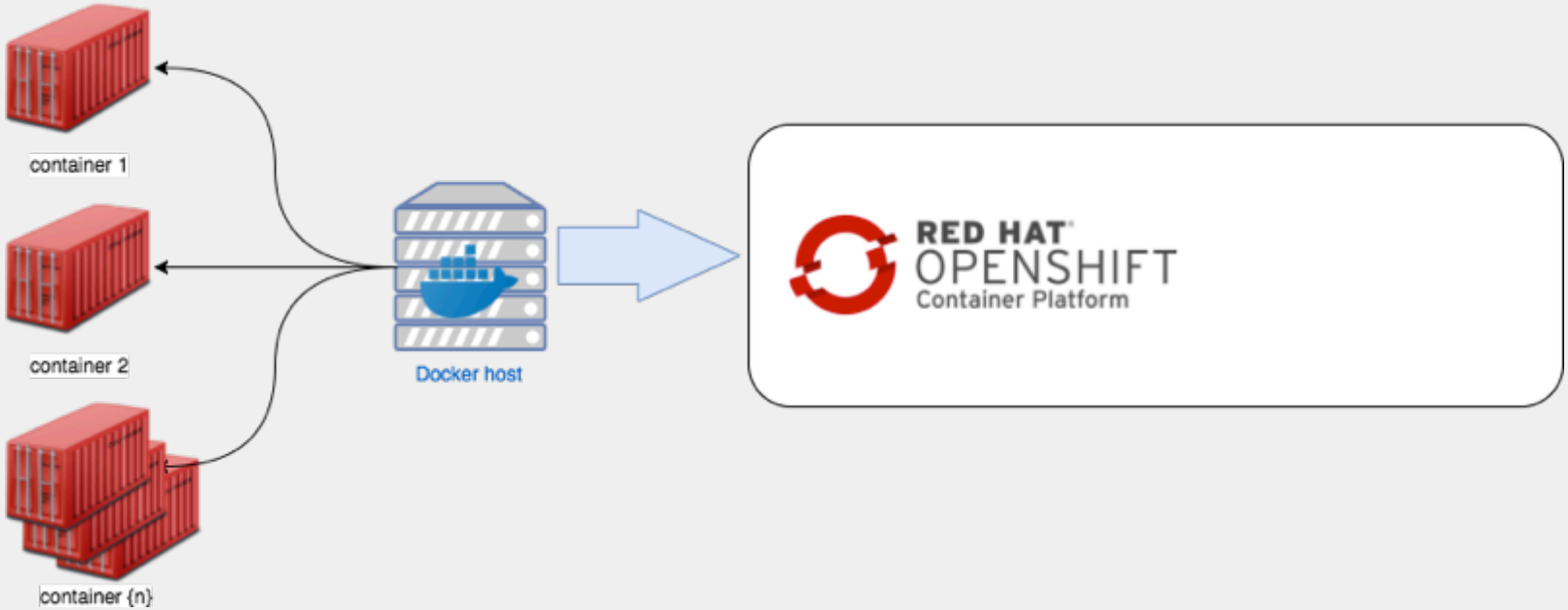
The smart move



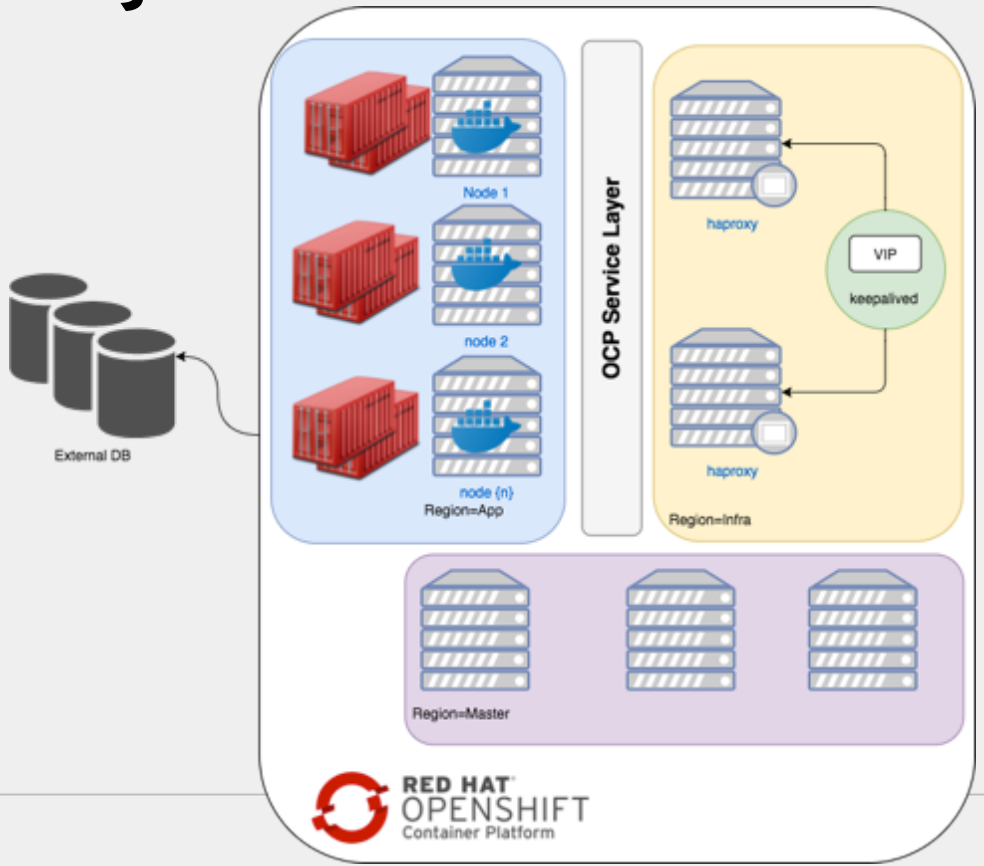
The smart move, the initial design



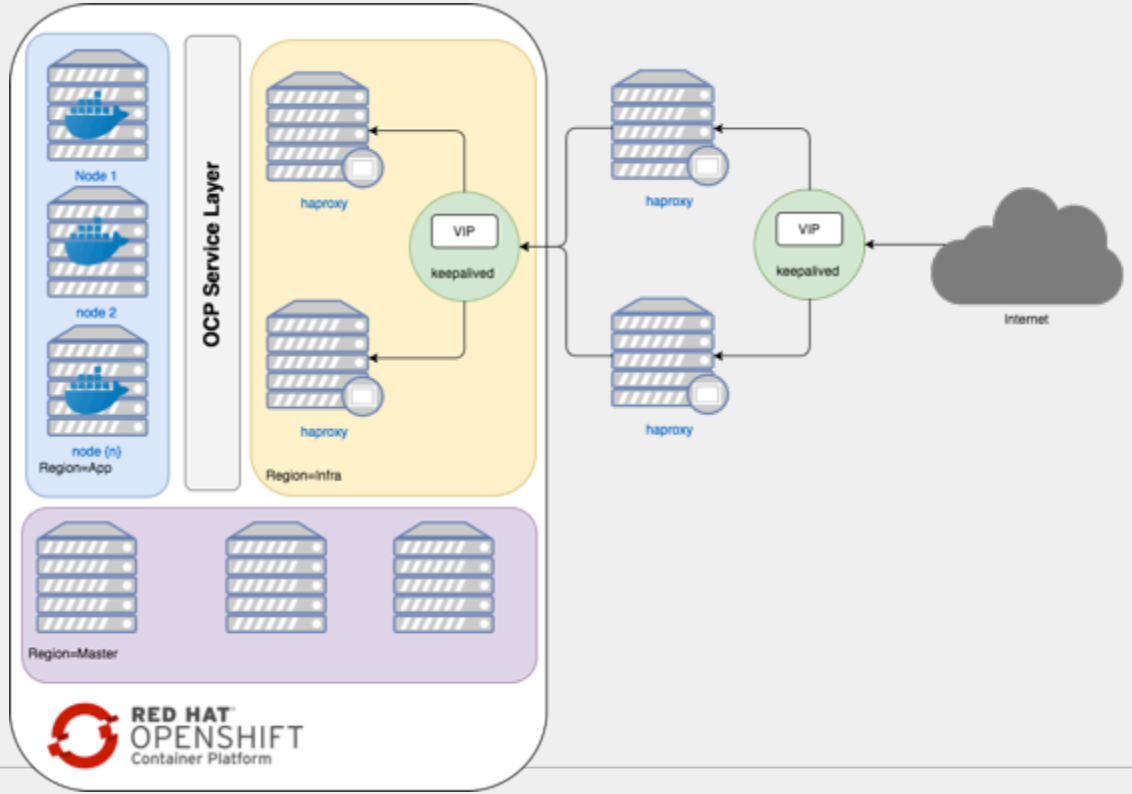
The smart move



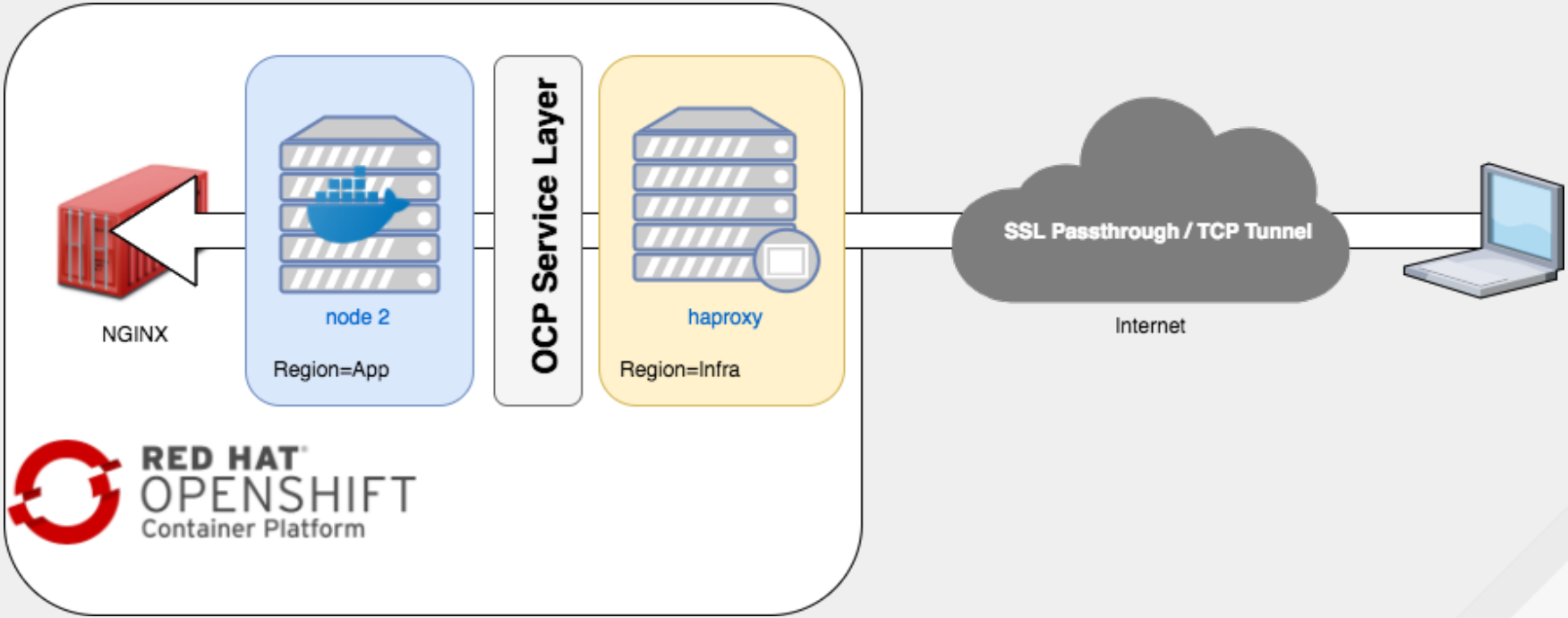
What design we recommended



The final look

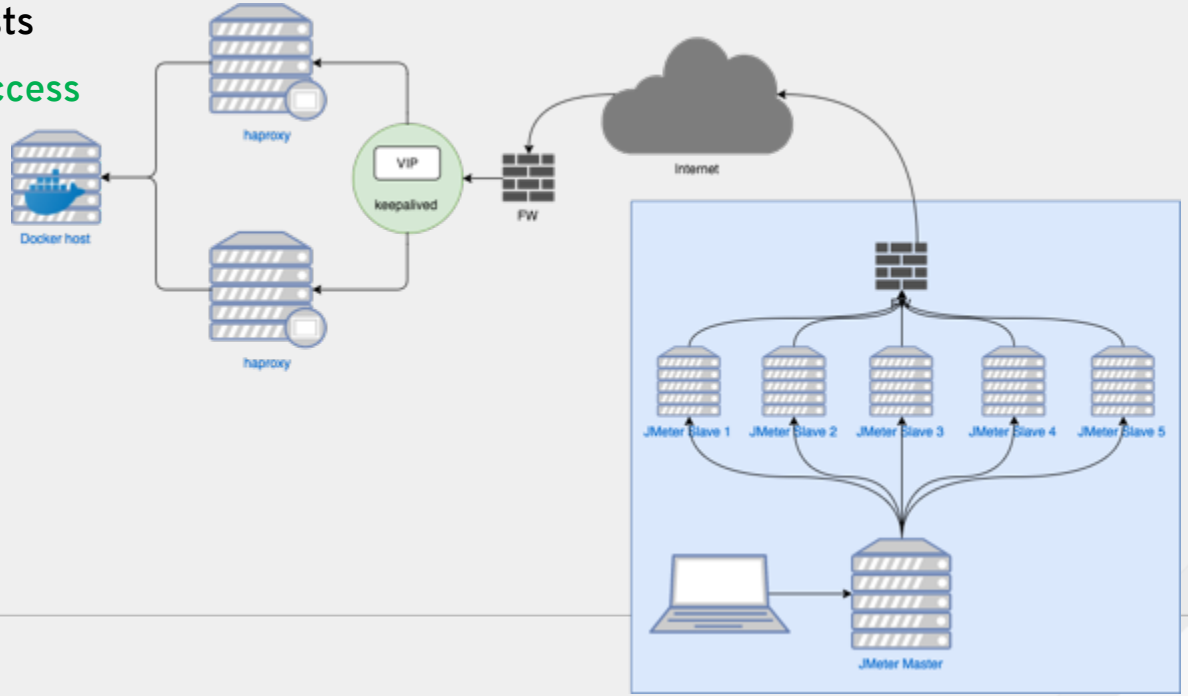


How communication works



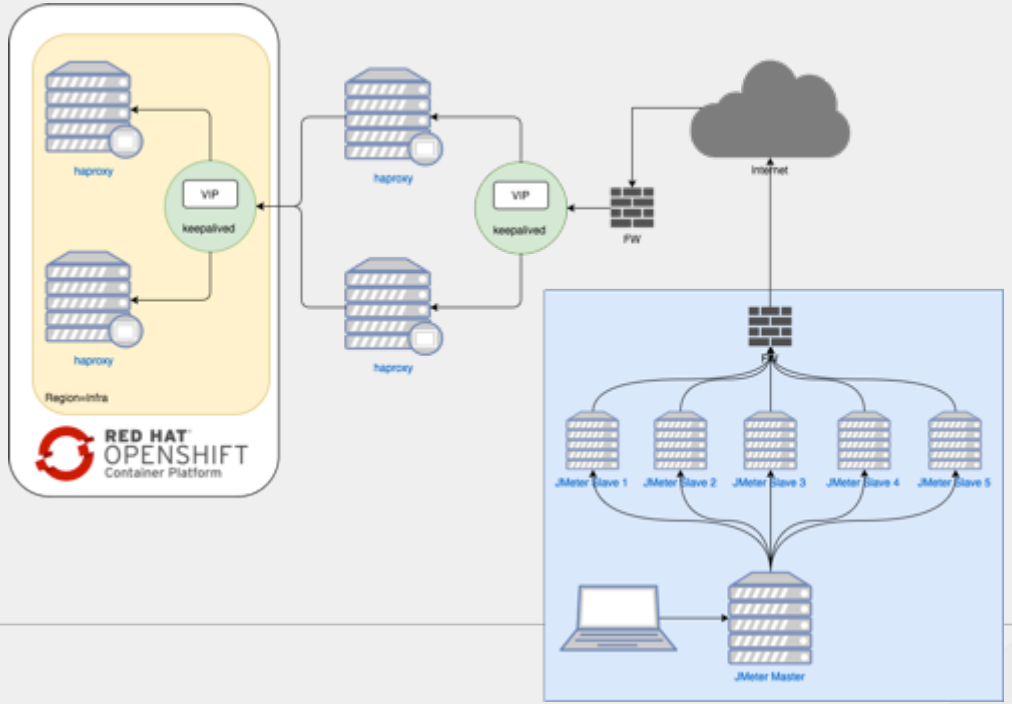
The First Tests on Standalone Docker Hosts

- HTTP load tests toward standalone Docker daemon from partner network
- 10K concurrent requests
- High percentage of success

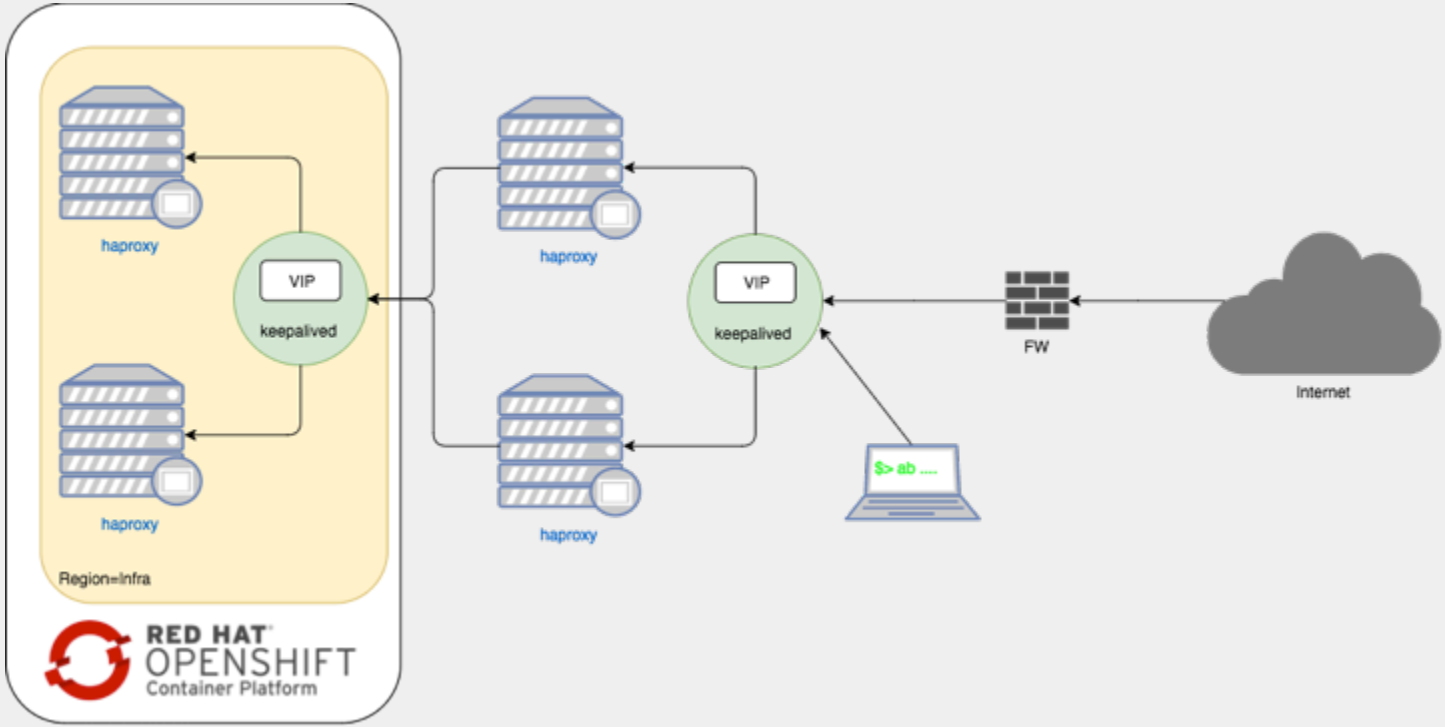


The First Tests on OCP

- HTTP load tests toward OpenShift from partner network
- 10K concurrent requests
- High percentage of failure



#1 Investigation



#1 Investigation

- Running HTTPS stress tests with **ab** command against the frontal HAProxies toward OCP

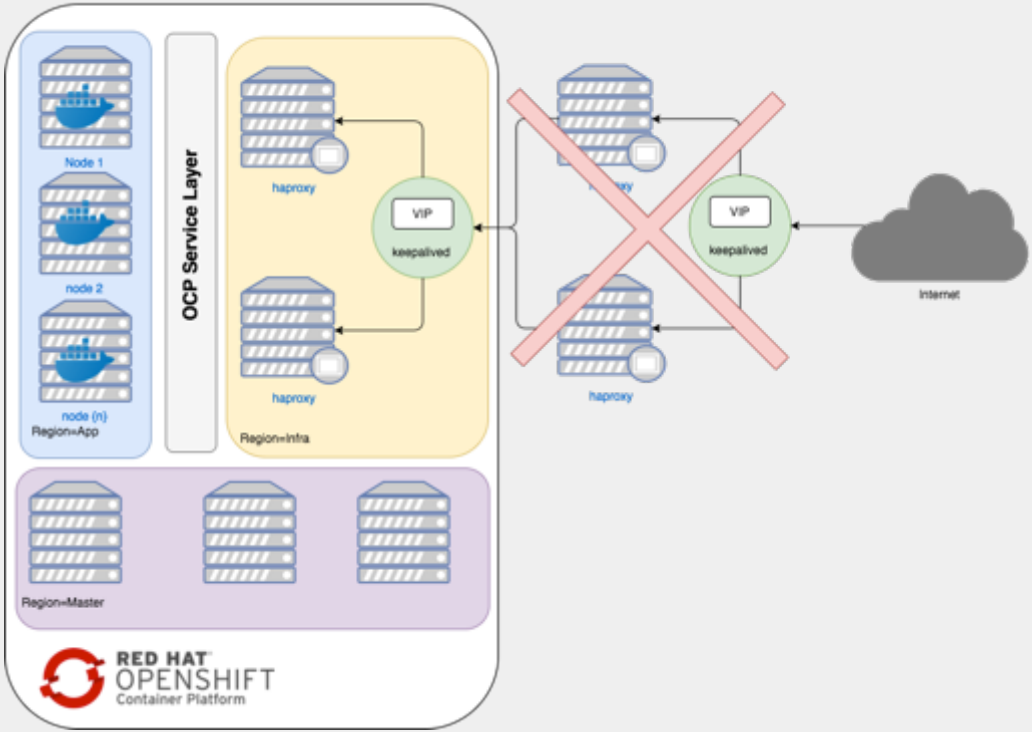
#1 Results

1. 70% of failures , starting to fail at 3K requests
2. Results with ab different from partner's JMeter cluster
3. OpenShift HAProxies (OCP routers) crashing
4. MAX_CONNECTIONS set to 20'000
5. Bypassing internet and firewall biased the results
6. Partner's firewall dropping some traffic
7. Frontal HAProxies underperforming and adding unnecessary complexity

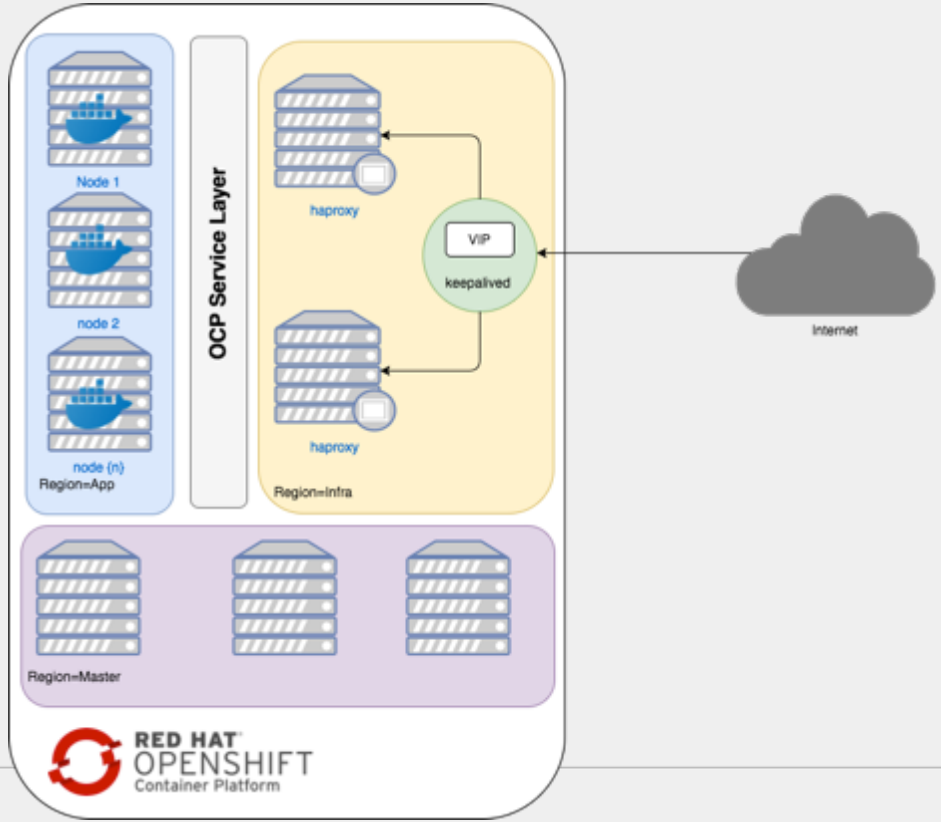
#1 Mitigation

- OpenShift HAProxies (OCP routers)
  - Crashing router: Bug opened at Red Hat [1], solved in OCP 3.7.54
  - MAX_CONNECTION=50000
- Bypassing internet and firewall
- Frontal HAProxies underperforming: Extended memory and CPU allocation, tuning HAProxy: MAX_CONNECTION=50000
- Frontal HAProxies removed
- Deployed JMeter Cluster in front of OCP

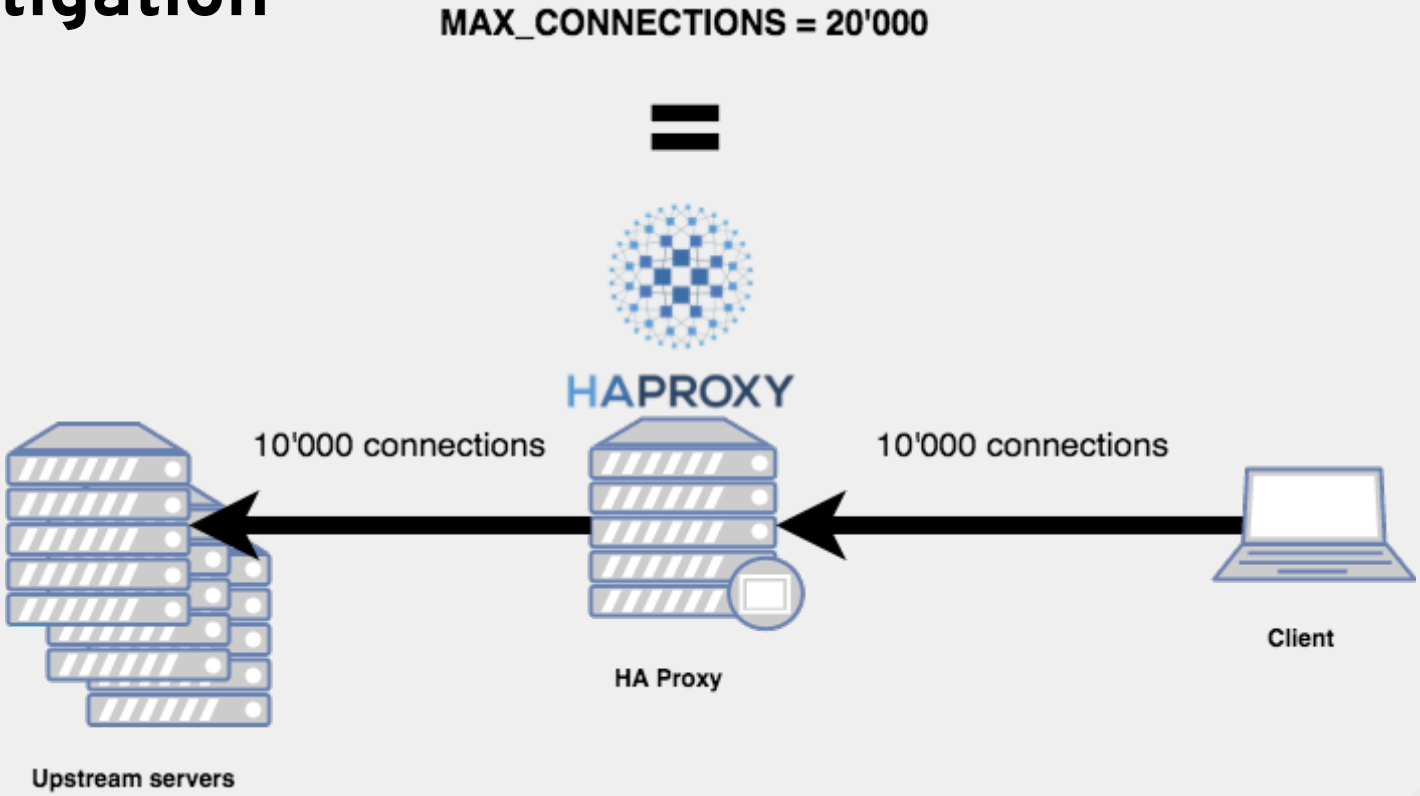
#1 Mitigation



#1 Mitigation



#1 Mitigation



JMeter vs ab

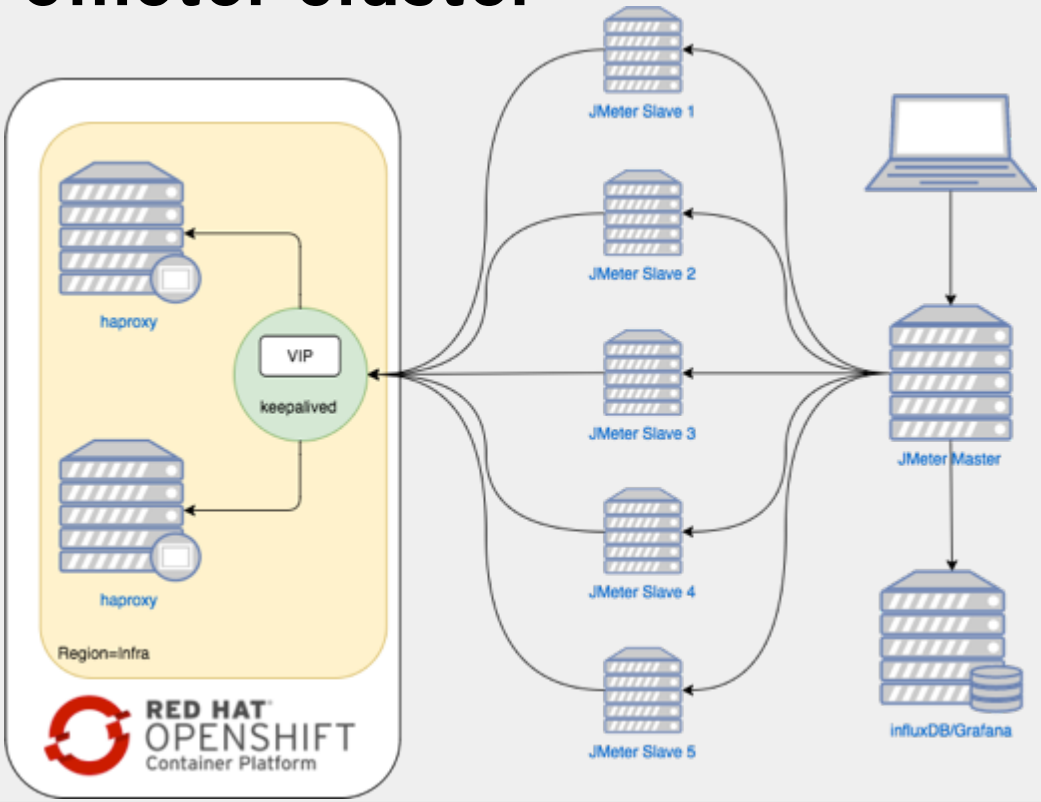
ab

- Real name: Apache Benchmark
- Available in RHEL base repo
- Simple command to use

Jmeter

- Distributed testing
- Complex test recipes
- GUI
- Java based

Jmeter cluster



- 1 x Master to control the slaves with a GUI or CLI
- {n} x Slave to run concurent requests
- InfluxDB: time series DB to collect and store data
- Grafana: live charts based on influxDB data
- Fully deployed and configured with

ANSIBLE

#2 investigation

- Running 10K concurrent HTTPS requests against OCP

#2 results

- 70% of requests failed
- OCP Routers consuming the full CPU of the pod
- SSL Termination pods (Nginx) consuming the full CPU of the node

#2 mitigation

- OCP Routers
  - Configured support of multiple CPU
    - nbproc 4
      cpu-map 1 0
      cpu-map 2 1
      cpu-map 3 2
      cpu-map 4 3
  - Extended number of maximum connection
    - ROUTER_MAX_CONNECTIONS=10000
  - Enabled logging in debug mode
    - ROUTER_SYSLOG_ADDRESS = <syslog IP>
      ROUTER_LOG_LEVEL = debug
  - Set higher CPU limit to the pod handling the SSL endpoint

#2 mitigation

- Extended OCP node running routers to match expected results given by RH

On an public cloud instance of size 4 vCPU/16GB RAM, a single HAProxy router is able to handle between 7000-32000 HTTP keep-alive requests depending on encryption, page size, and the number of connections used. For example, when using TLS edge or re-encryption terminations with large page sizes and a high numbers of connections, expect to see results in the lower range. With HTTP keep-alive, a single HAProxy router is capable of saturating 1 Gbit NIC at page sizes as small as 8 kB.

The table below shows HTTP keep-alive performance on such a public cloud instance with a single HAProxy and 100 routes:

Encryption	Page size	HTTP(s) requests per second
none	1kB	15435
none	4kB	11947
edge	1kB	7467
edge	4kB	7678
passthrough	1kB	25789
passthrough	4kB	17876
re-encrypt	1kB	7611
re-encrypt	4kB	7395

#3 investigation

- Running 10K concurrent HTTPS requests against OCP

#3 results

- 30% of requests failed
- CPU Consumption reduced and distributed among all CPUs of the OCP Routers
- Router logging:
  haproxy[144]: Connect() failed for backend be_tcp:cut-dev1:webserver: no free ports.

#3 mitigation

- Kernel tuning:
  - net.ipv4.ip_local_port_range="1025 65000”
  - net.ipv4.tcp_tw_reuse = 1
  - fs.nr_open = 100000
  - fs.file-max = 100000
- Router tuning:
  - DROP_SYN_DURING_RESTART = false

#4 investigation

- Running 10K concurrent HTTPS requests against OCP

#4 results

- < 1% of requests failed

#4 investigation

- Running 10K concurrent HTTPS requests against OCP

#4 results

- < 1% of requests failed