Delivering Microservices Securely and at Scale with NGINX in Red Hat OpenShift

November, 2017

NGINX

Klaus Oxdal

Channel Director

NGINX

klaus@nginx.com

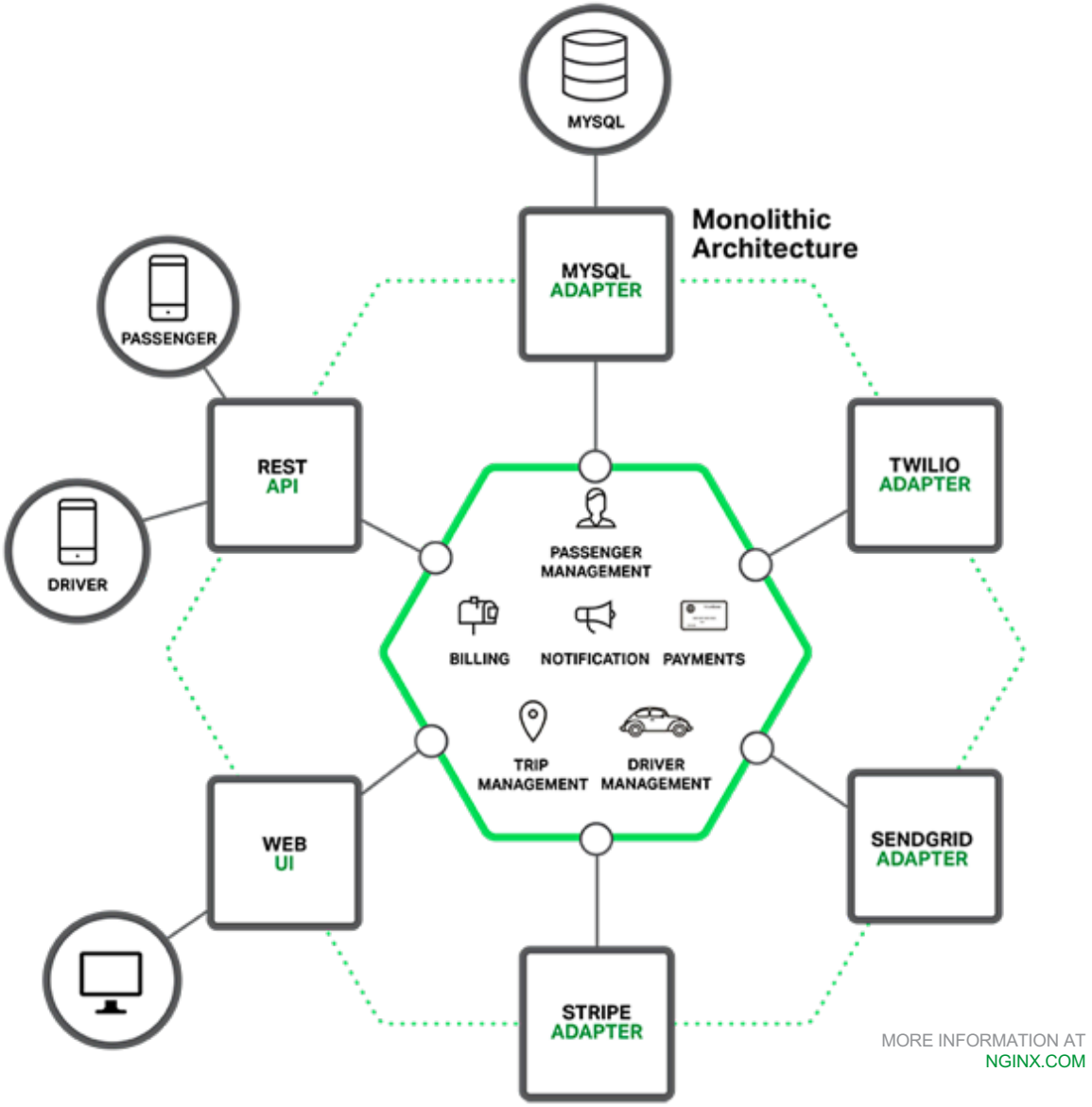
MORE INFORMATION AT
NGINX.COM

The Big Shift

Architectural Changes:

Monolith

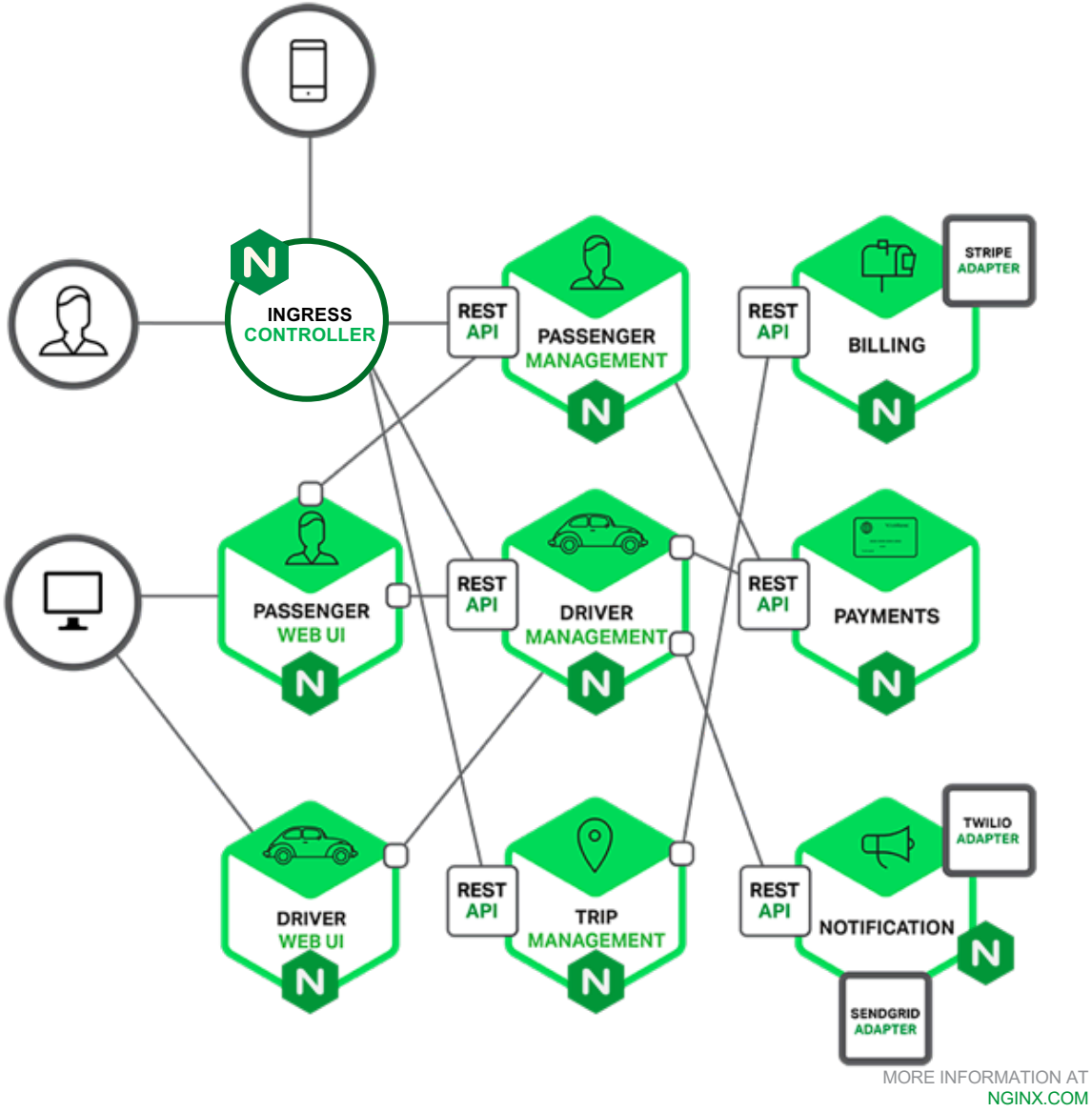
```
import myapp.Driver

require('myapp.Driver')

from myapp import Driver;
```



Architectural Changes:

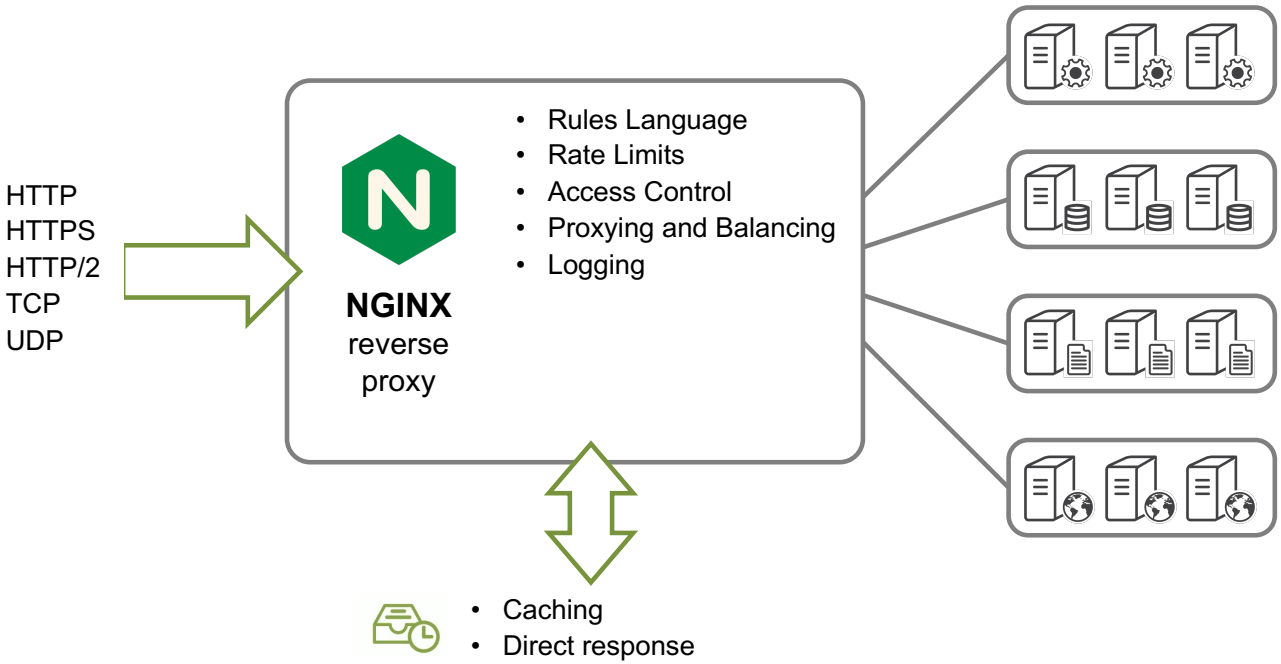
Microservices

```
http.request( opts,
  function(res) { } );

$s = curl_init( $uri );
$r = curl_exec( $s );

res = requests.get(uri)
```



What is NGINX?

High-performance web server, application gateway, and app. accelerator



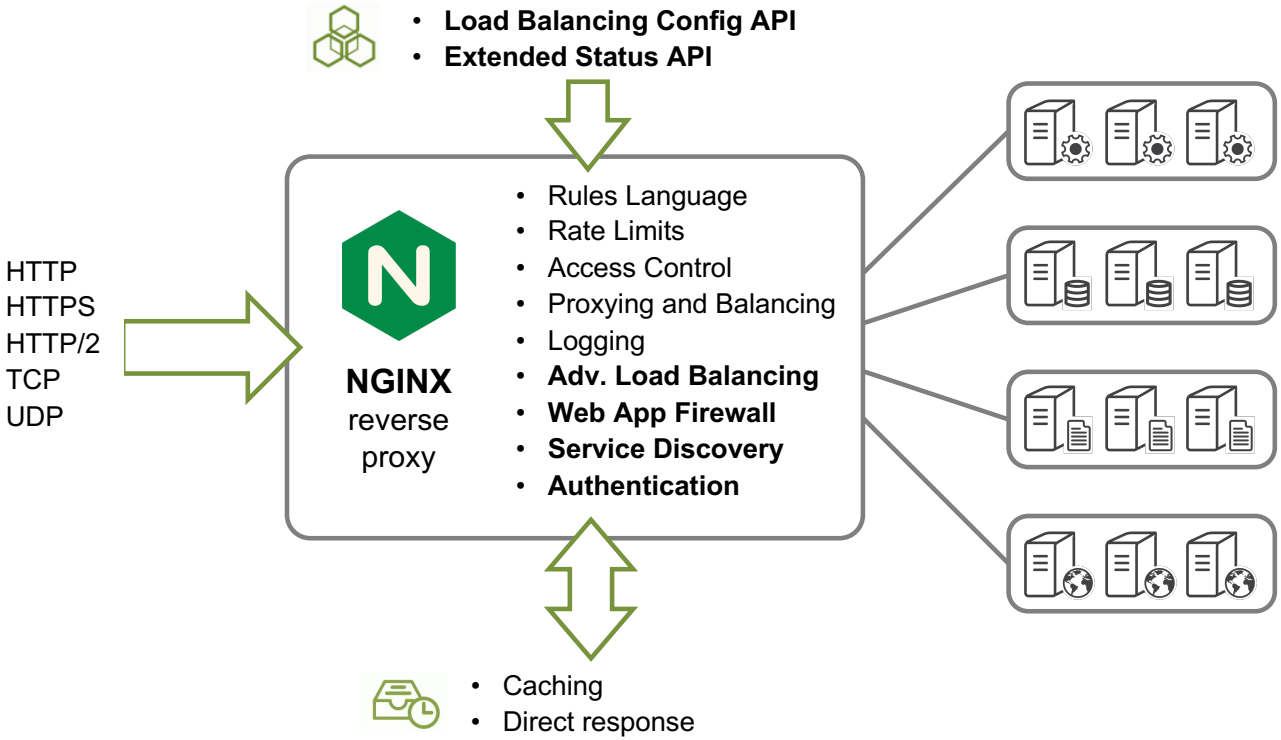
MORE INFORMATION AT
NGINX.COM

What is NGINX Plus?

Load Balancer and App Delivery Controller

- Native Service Discovery
- Detailed monitoring
- Authentication
- API configuration

NGINX+

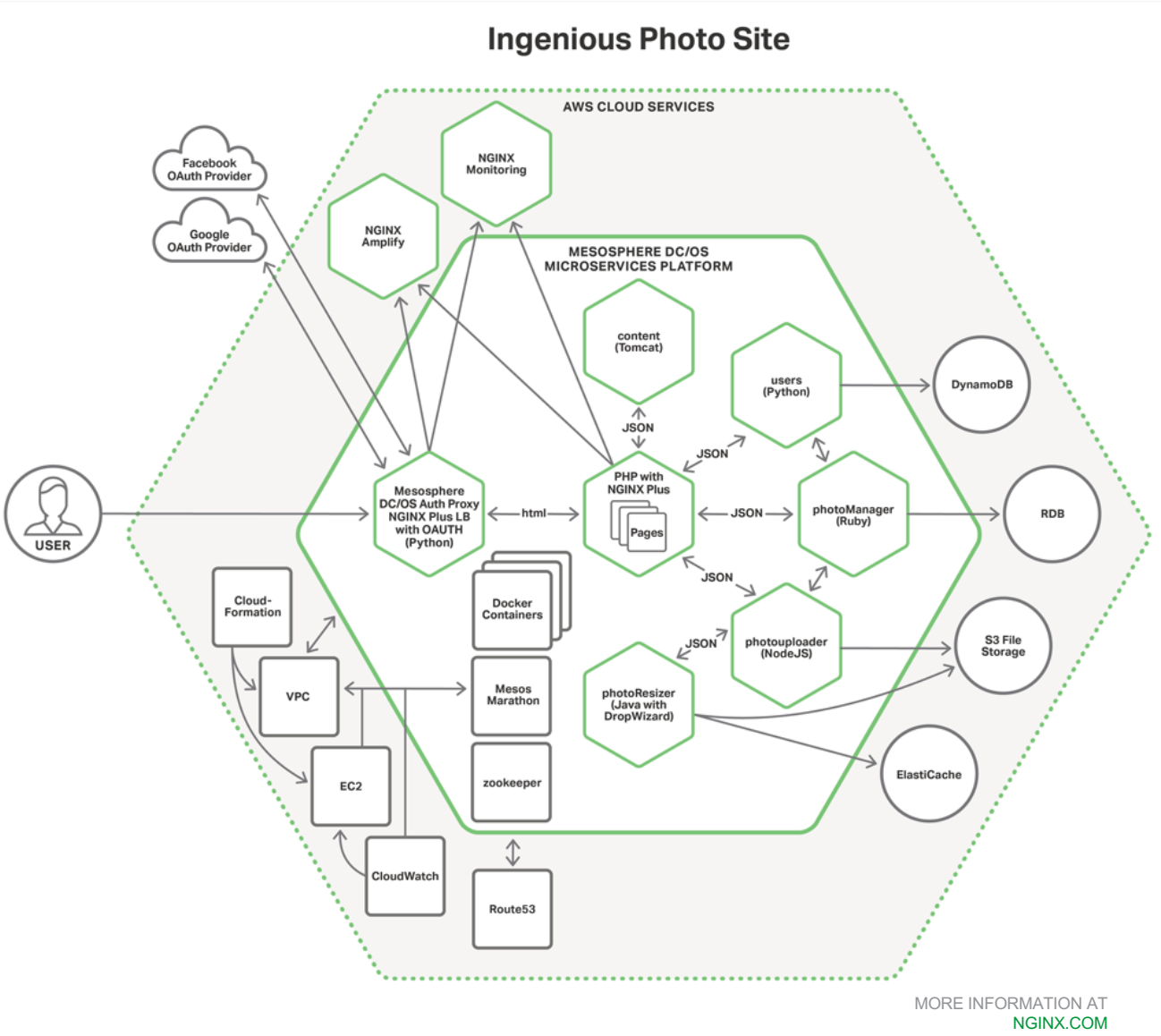


MORE INFORMATION AT
NGINX.COM

MORE INFORMATION AT
NGINX.COM

Microservices Reference Architecture

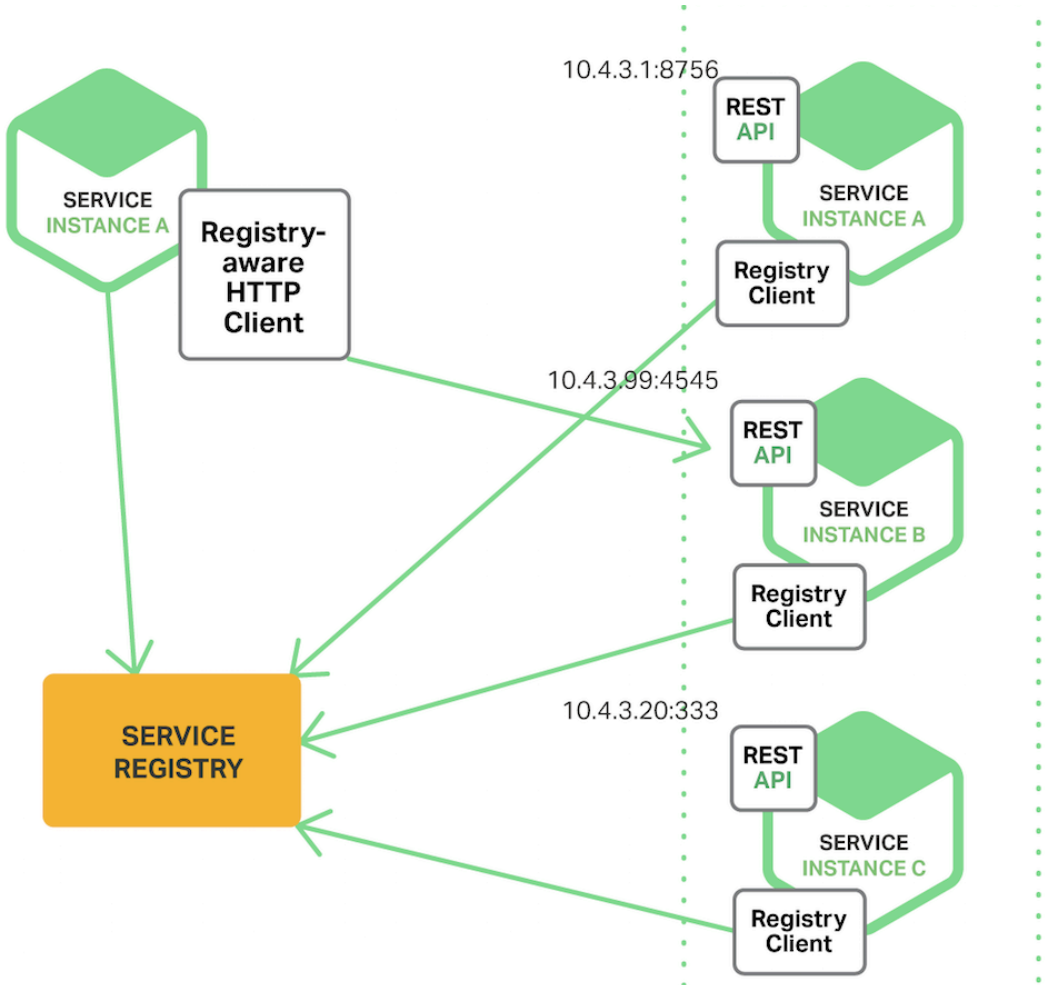
- Containers
- Polyglot services
- 12-Factor App design



The Networking Problem

Service Discovery

One service needs to know where other services are

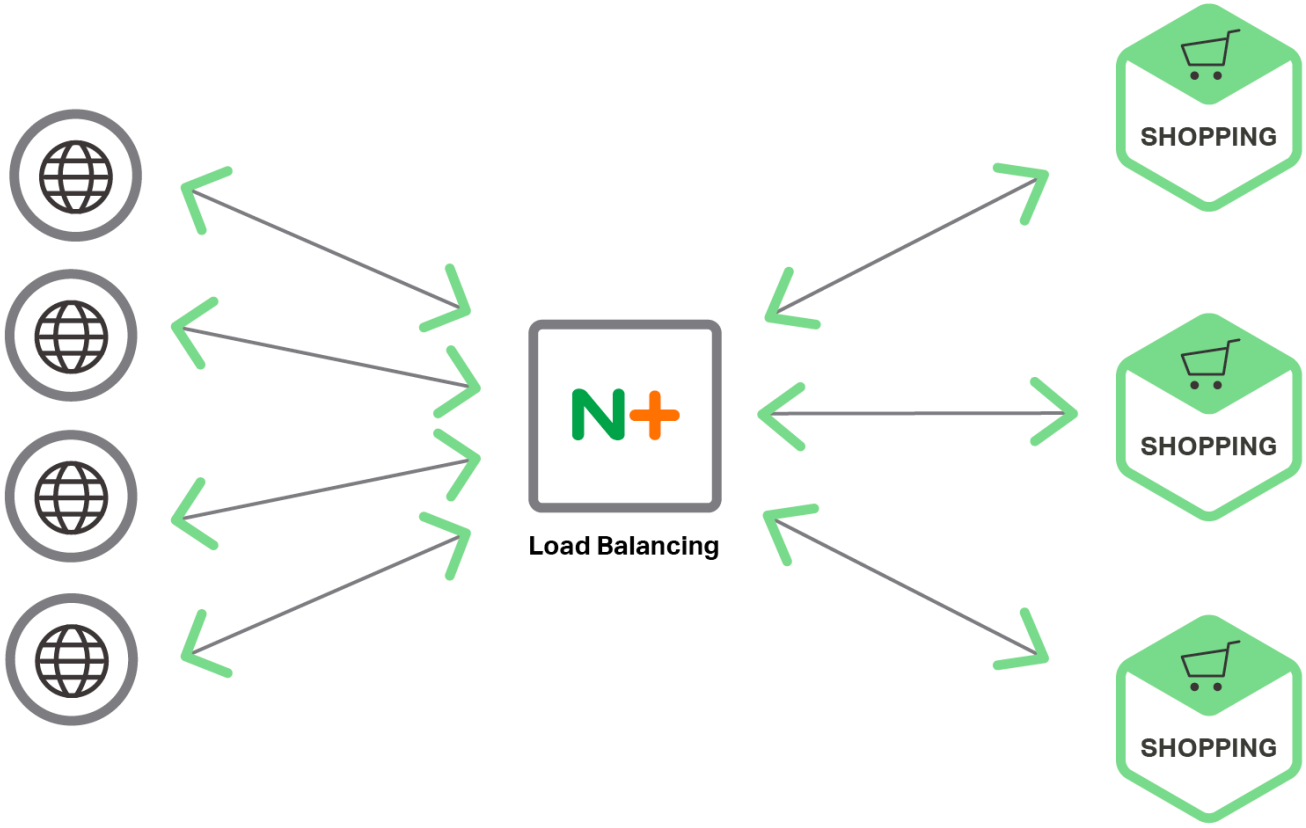


MORE INFORMATION AT
NGINX.COM

Load-balancing

Simple Load Balancing is not effective

Developers need control



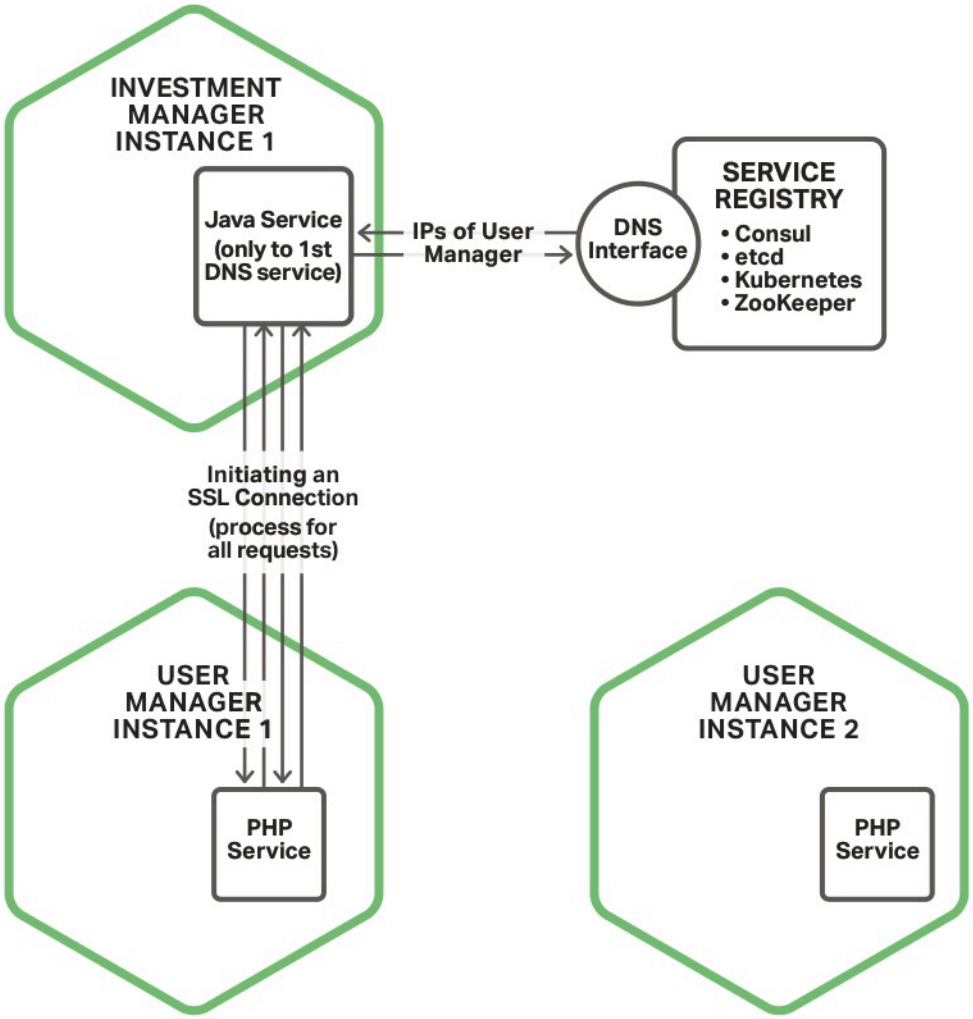
MORE INFORMATION AT
NGINX.COM

Secure & Fast Communication

Encryption at the transmission layer is becoming a must-have

SSL communication is slow and encryption is CPU intensive

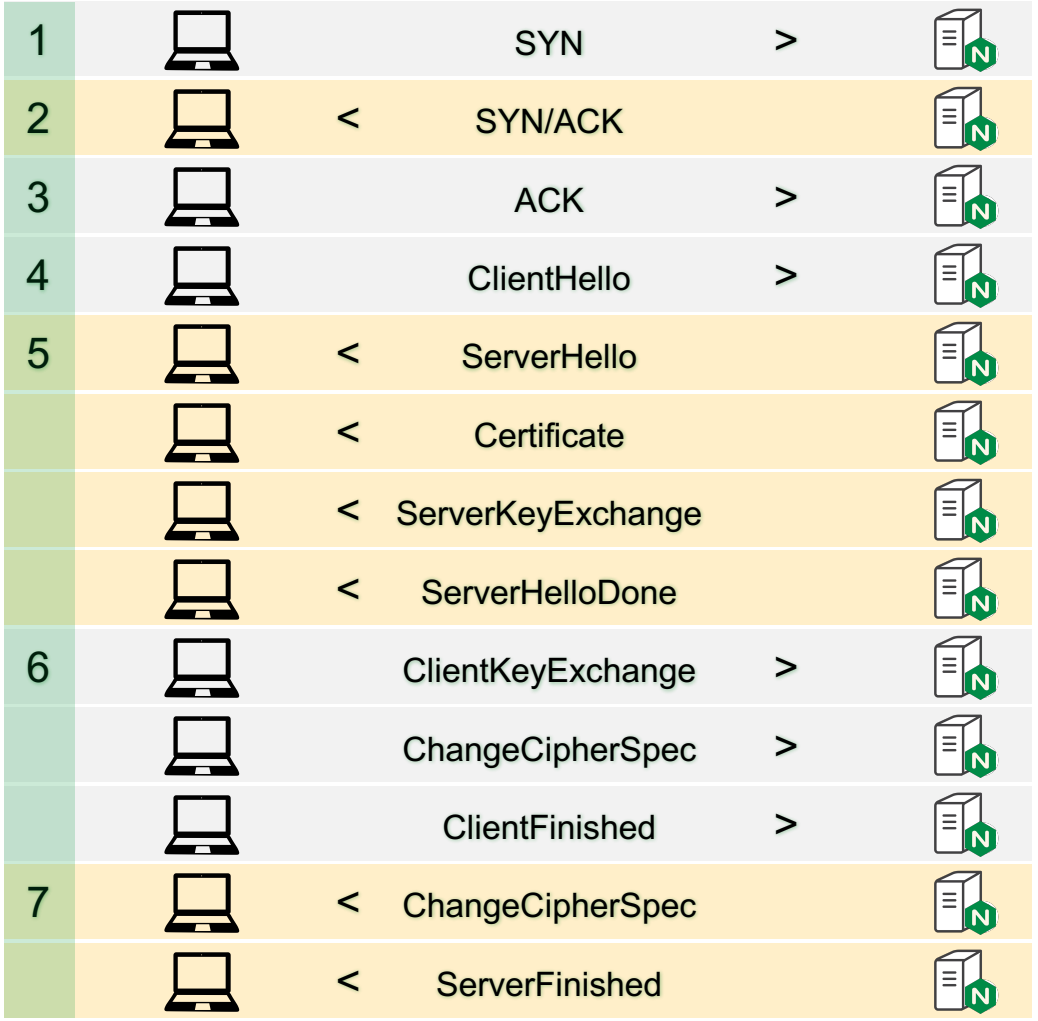
VPNs add complexity and are not a good fit



MORE INFORMATION AT
NGINX.COM

The SSL problem

A new SSL connection takes a minimum of 7 messages to establish.



MORE INFORMATION AT
NGINX.COM

The Networking Problem

- Service discovery
- Robust load balancing
- Persistent encryption

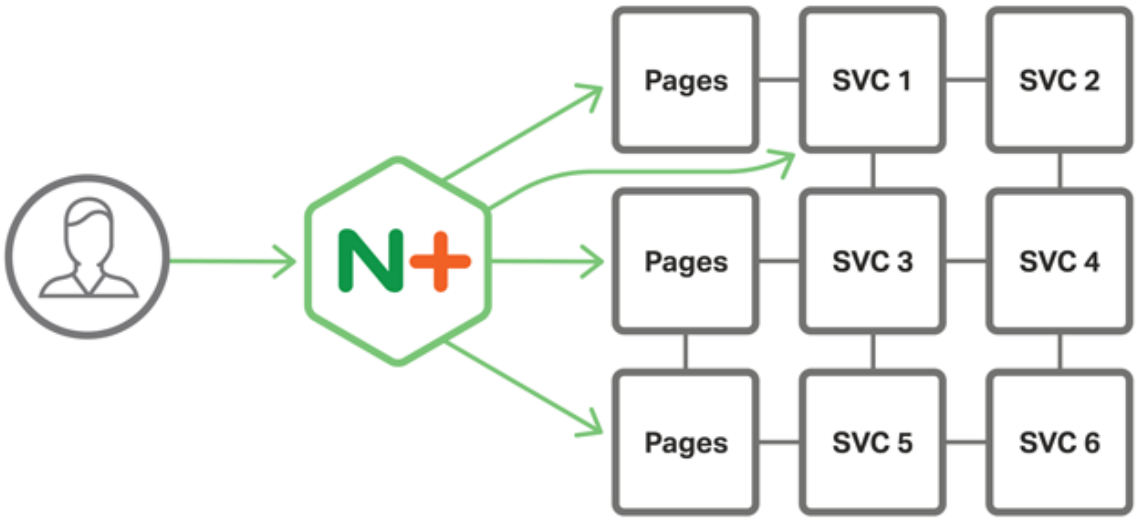
Three Network Architectures

Proxy Model

- Focus on internet traffic
- A shock absorber for your app
- Dynamic connectivity

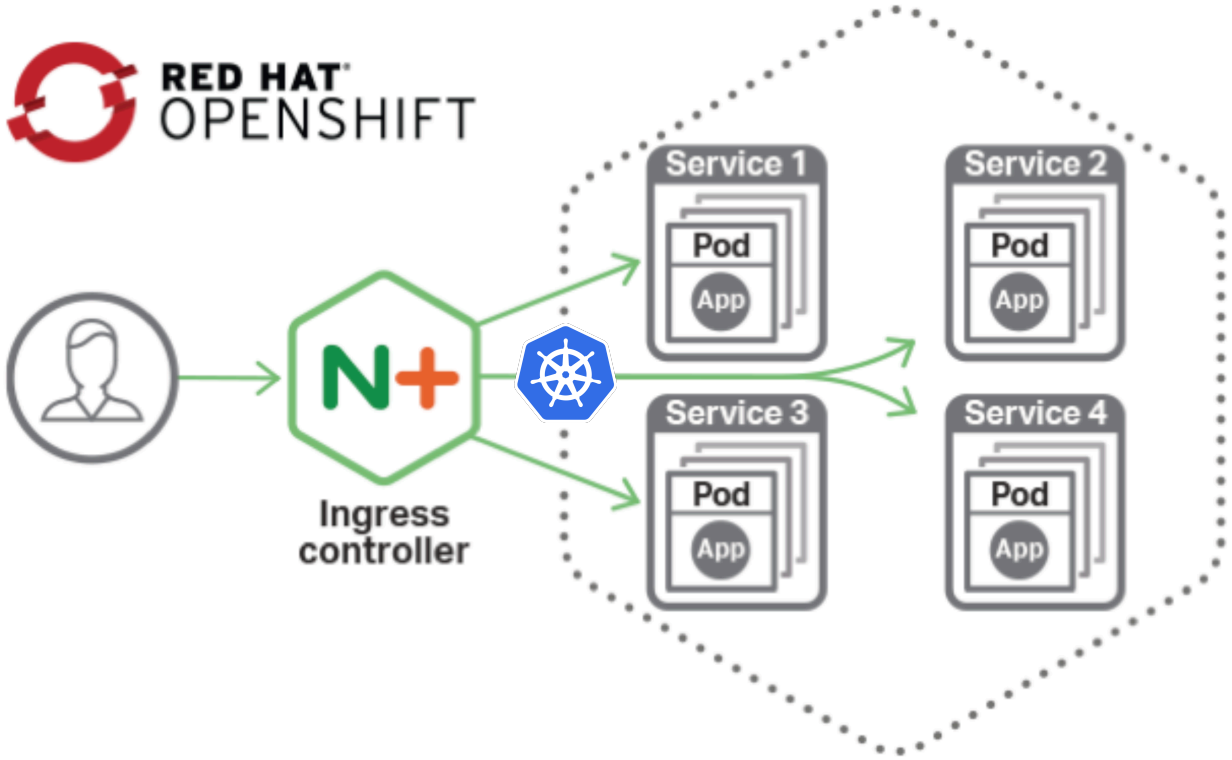
Proxy Model

- Inbound traffic is managed through a reverse proxy/load balancer
- Services are left to themselves to connect to each other.



MORE INFORMATION AT
NGINX.COM

NGINX Kubernetes Ingress Controller for Red Hat OpenShift



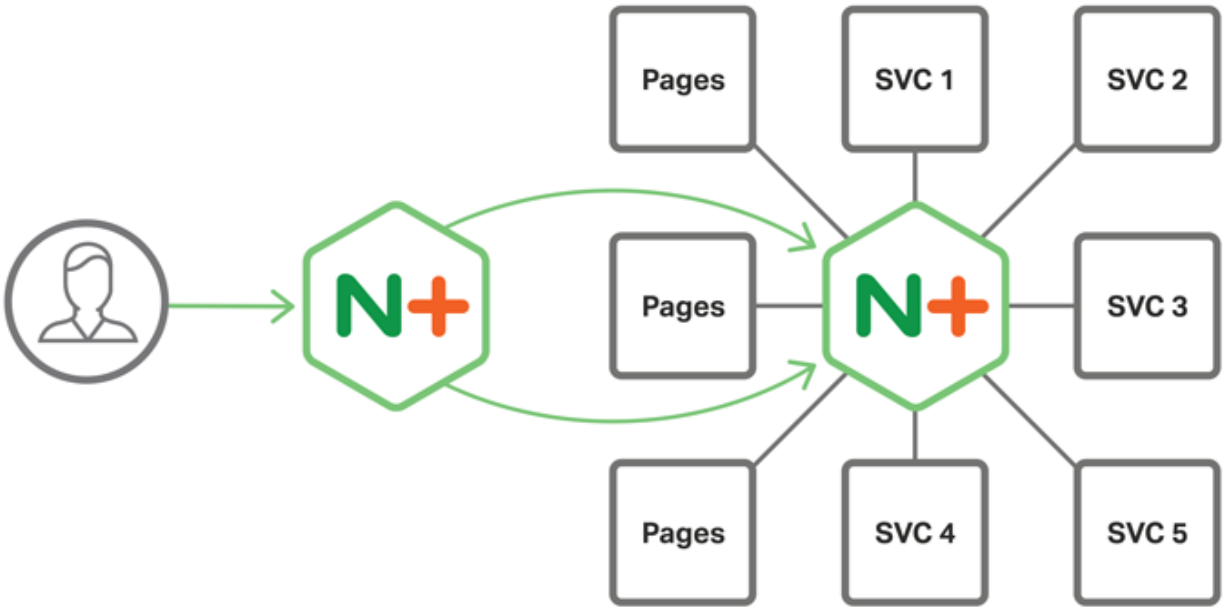
https://www.nginx.com/partners/red-hat/

Router Mesh

- Robust service discovery
- Advanced load balancing
- Circuit breaker pattern

Router Mesh Model

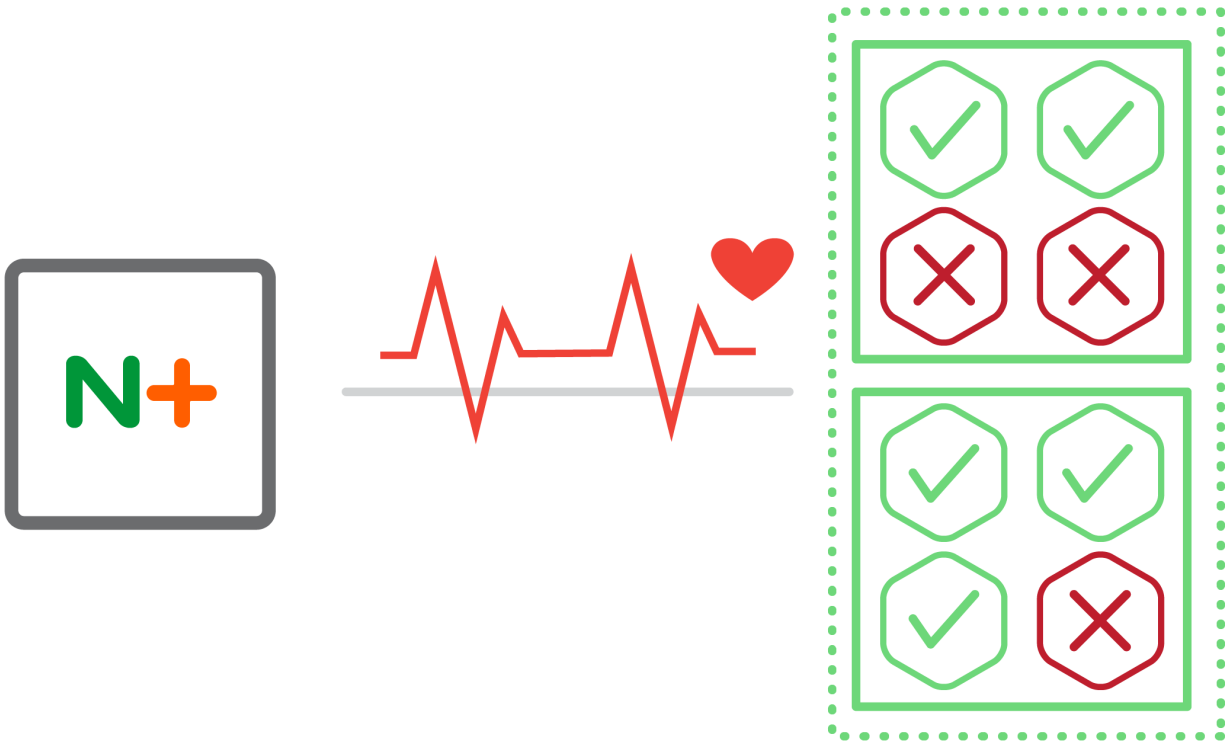
- In-bound routing through reverse proxy
- Centralized load balancing through a separate load balancing service
- Deis Router works like this



MORE INFORMATION AT
NGINX.COM

Circuit Breakers

- Active health checks
- Retry behaviour
- Caching
- Slowstart



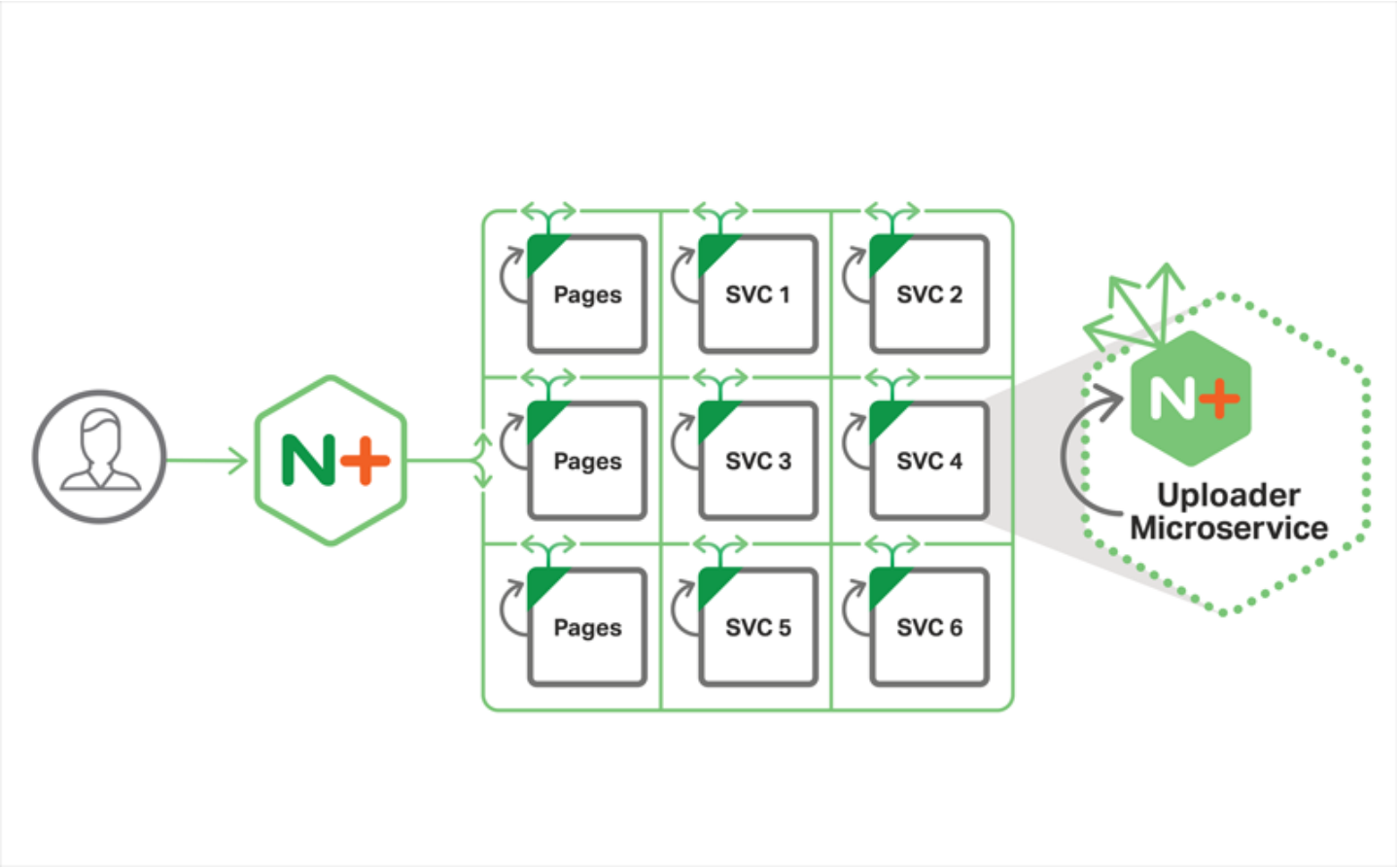
MORE INFORMATION AT
NGINX.COM

Fabric Model

- Robust service discovery
- Advanced load balancing
- Circuit breaker pattern
- Persistent SSL network

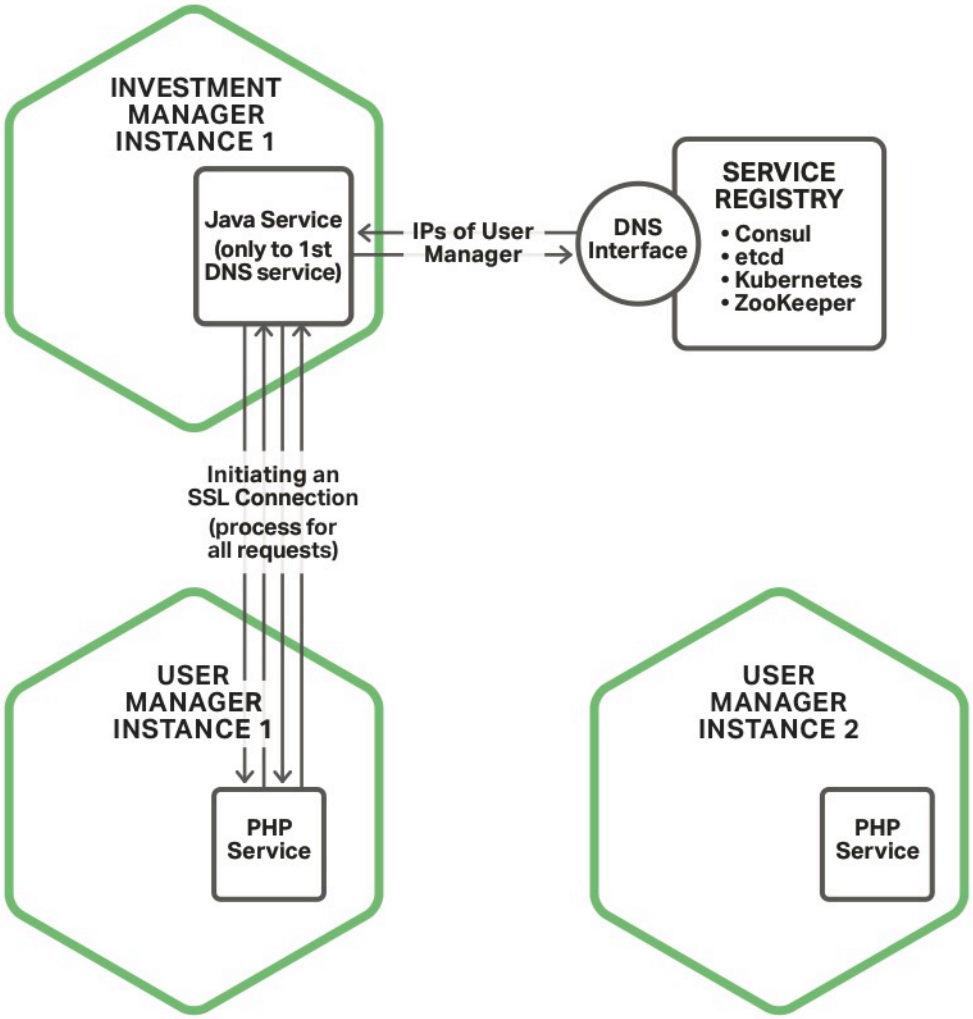
Inter-Process Communication

- Routing is done at the container level
- Services connect to each other as needed
- NGINX Plus acts as the forward and reverse proxy for all requests



Without the Fabric Model

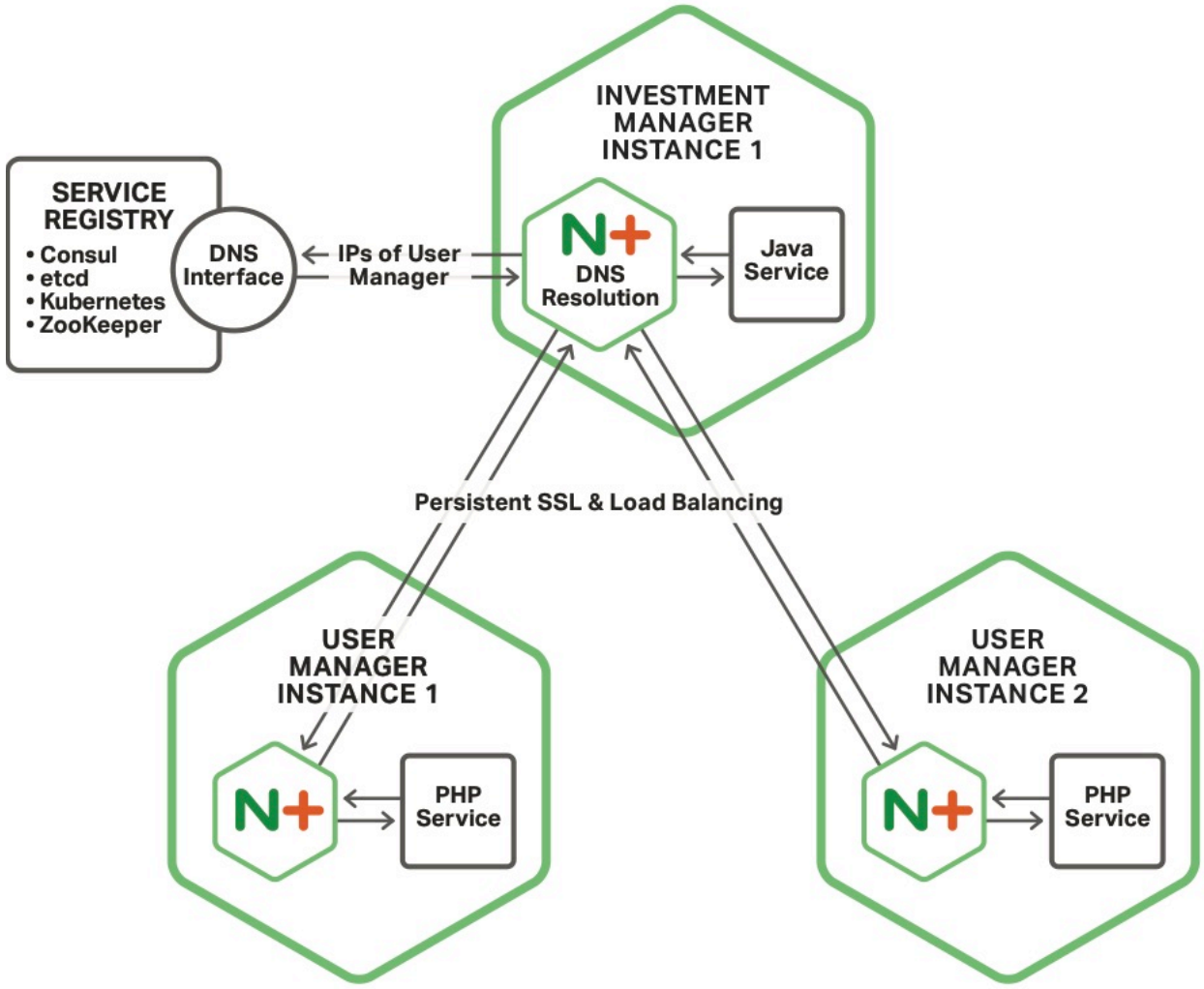
- DNS service discovery
- Relies on round robin DNS
- Each request creates a new SSL connection which fully implemented in 7+ requests



MORE INFORMATION AT
NGINX.COM

With the Fabric Model

- NGINX Plus runs in each container
- Application code talks to NGINX locally
- NGINX talks to NGINX
- NGINX queries the service registry



Closing Thoughts

NGINX

Microservices

Getting the **L7 network right** is critical to the success of a production microservices deployment.

NGINX has a supported **Ingress Controller** implementation that works with NGINX Plus.

We can help you to architect a scalable, secure, reliable **internal load balancing** solution.

All supported by **Red Hat** and **NGINX**

MORE INFORMATION AT
NGINX.COM

Reference links

- Running Ingress controller on OpenShift -- https://github.com/nginxinc/kubernetes-ingress/tree/master/examples/openshift
- Router Mesh -- https://github.com/nginxinc/router-mesh-architecture
- Fabric Model -- https://github.com/nginxinc/fabric-model-architecture
- Nginmesh -- https://github.com/nginmesh/nginmesh
- Connect, Secure and Scale Microservices Red Hat and NGINX-- https://www.redhat.com/cms/managed-files/cl-pa-nginx-openshift-solution-brief-f9016-20171101-en.pdf