How to Re-Architect without Breaking Stuff (too much)

Owen Garrett
March 2018
owen@nginx.com

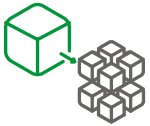
NGINX

All problems in computer science can be solved by another layer of indirection

--- David Wheeler, FRS


“This giant piece of software that made our company successful… is now a problem.”

The transition to a Modern Application Architecture

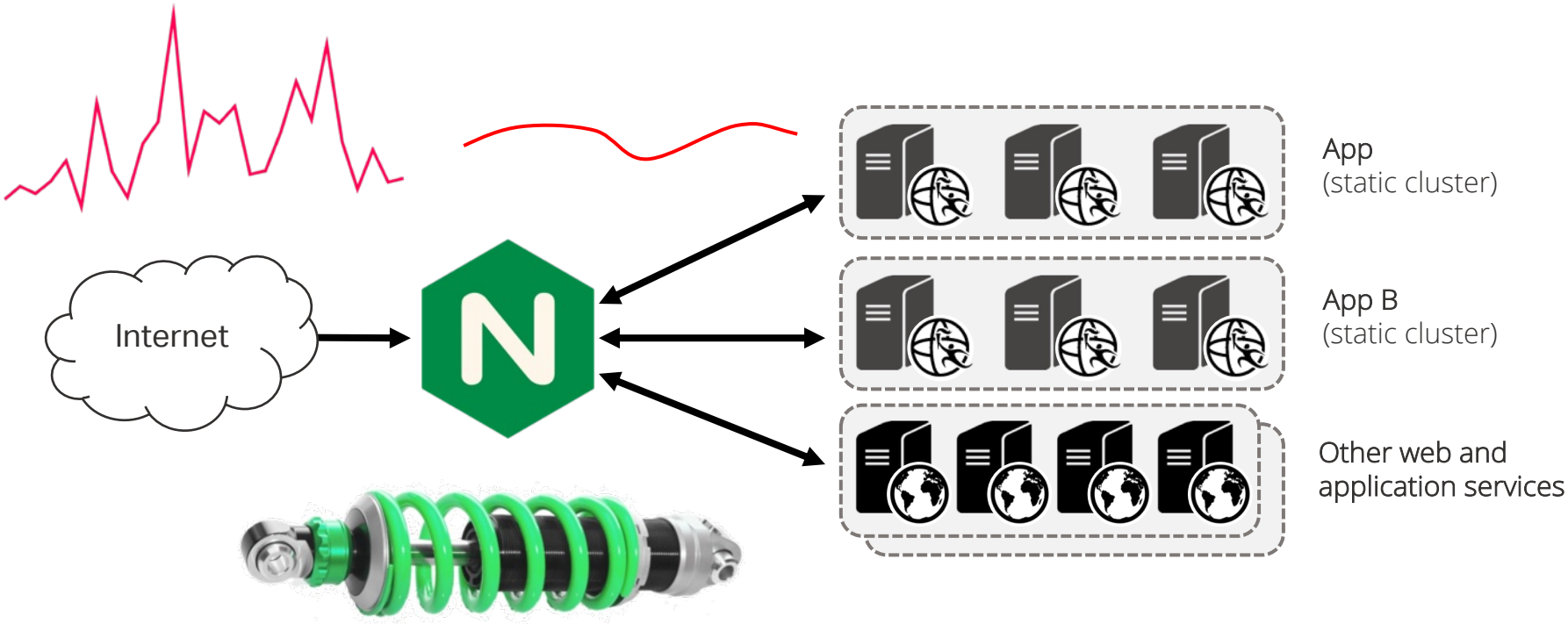


From Monolith …		… to Microservices
A giant piece of software	→	Small, loosely connected Services
Silo'ed teams (Dev, Test, Ops)	→	DevOps Culture
Big-bang releases	→	Continuous delivery
Persistent deployments	→	VMs, Containers, Functions
Fixed, static Infrastructure	→	Infrastructure as code
Complex protocols (HTML, SOAP)	→	Lightweight, Programmable (REST, JSON)

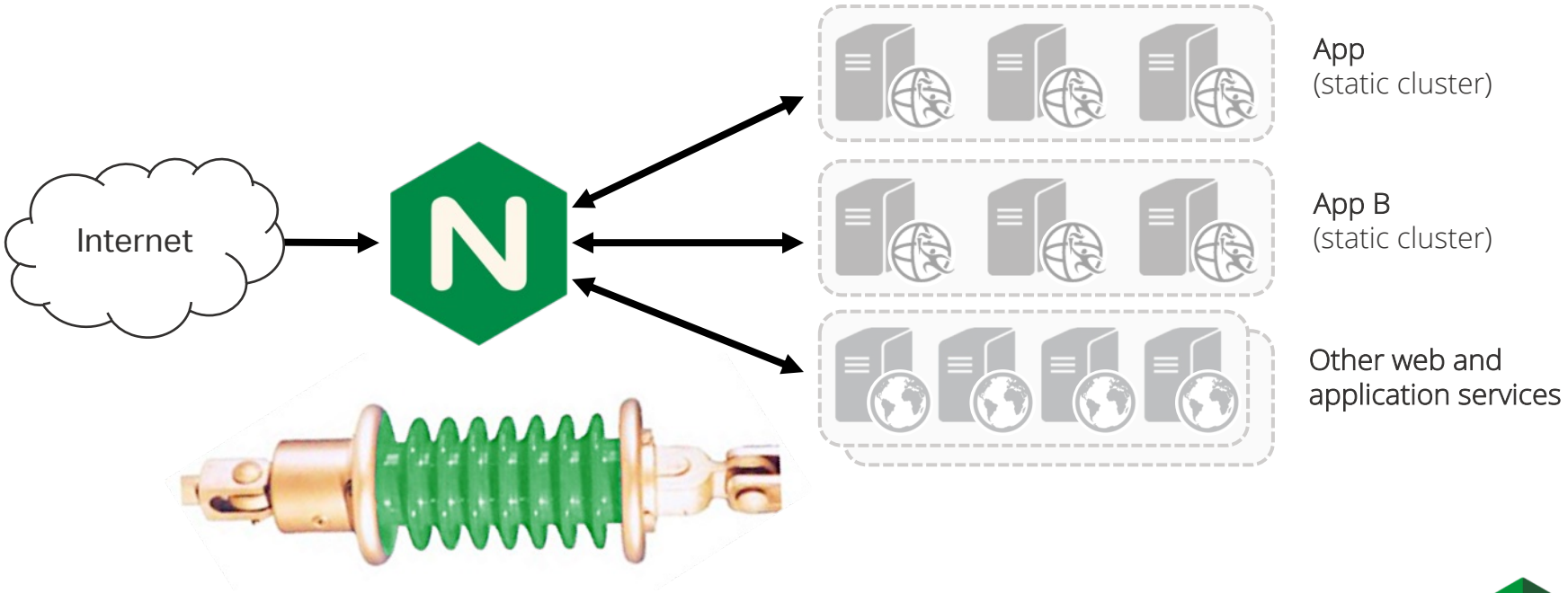

Disruption is happening
at the speed of software


NGINX

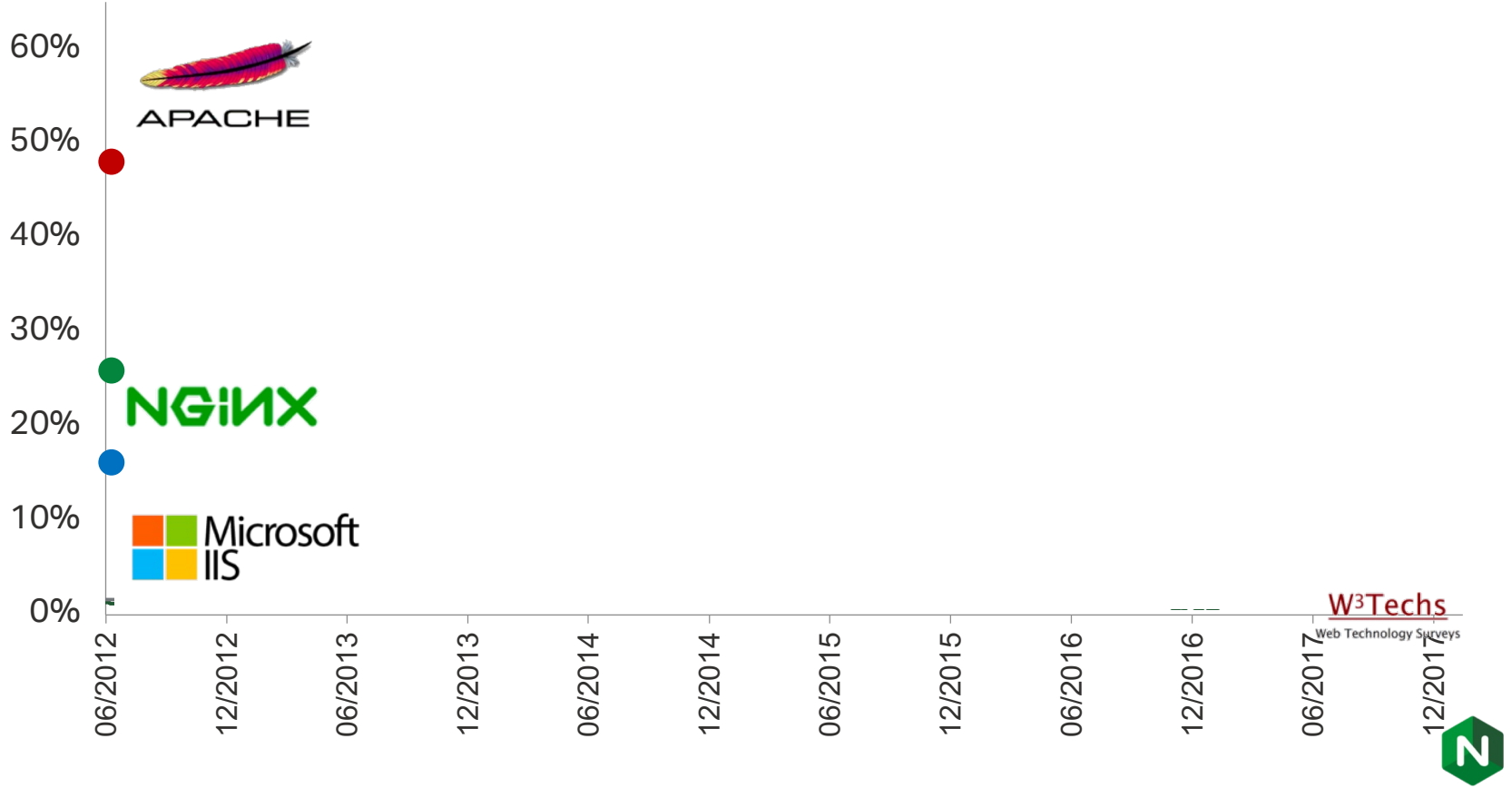
NGINX as a Shock Absorber



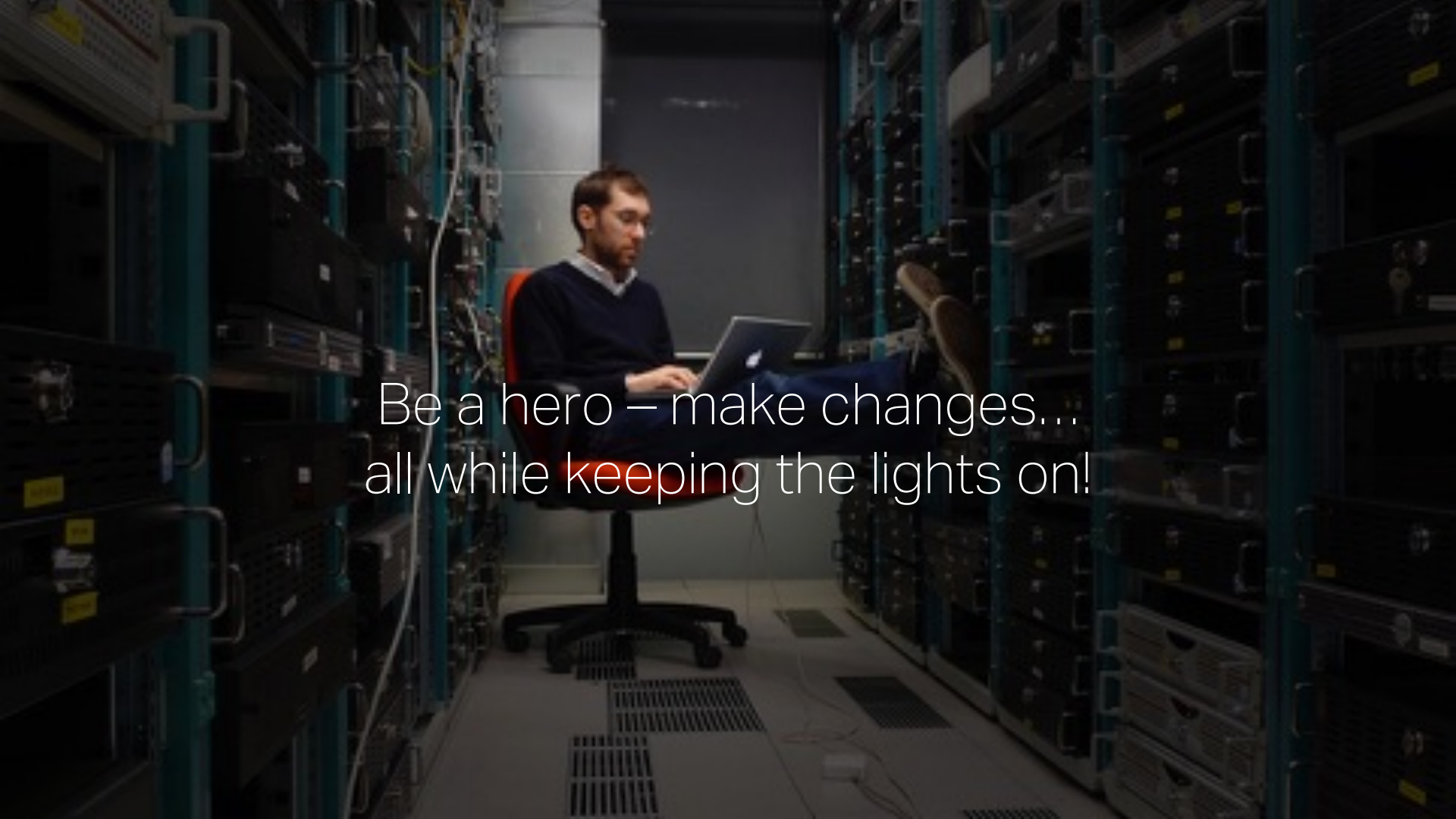
NGINX as an Insulator



The busiest sites in the world use NGINX open source



Four steps to non-disruptive rearchitecting

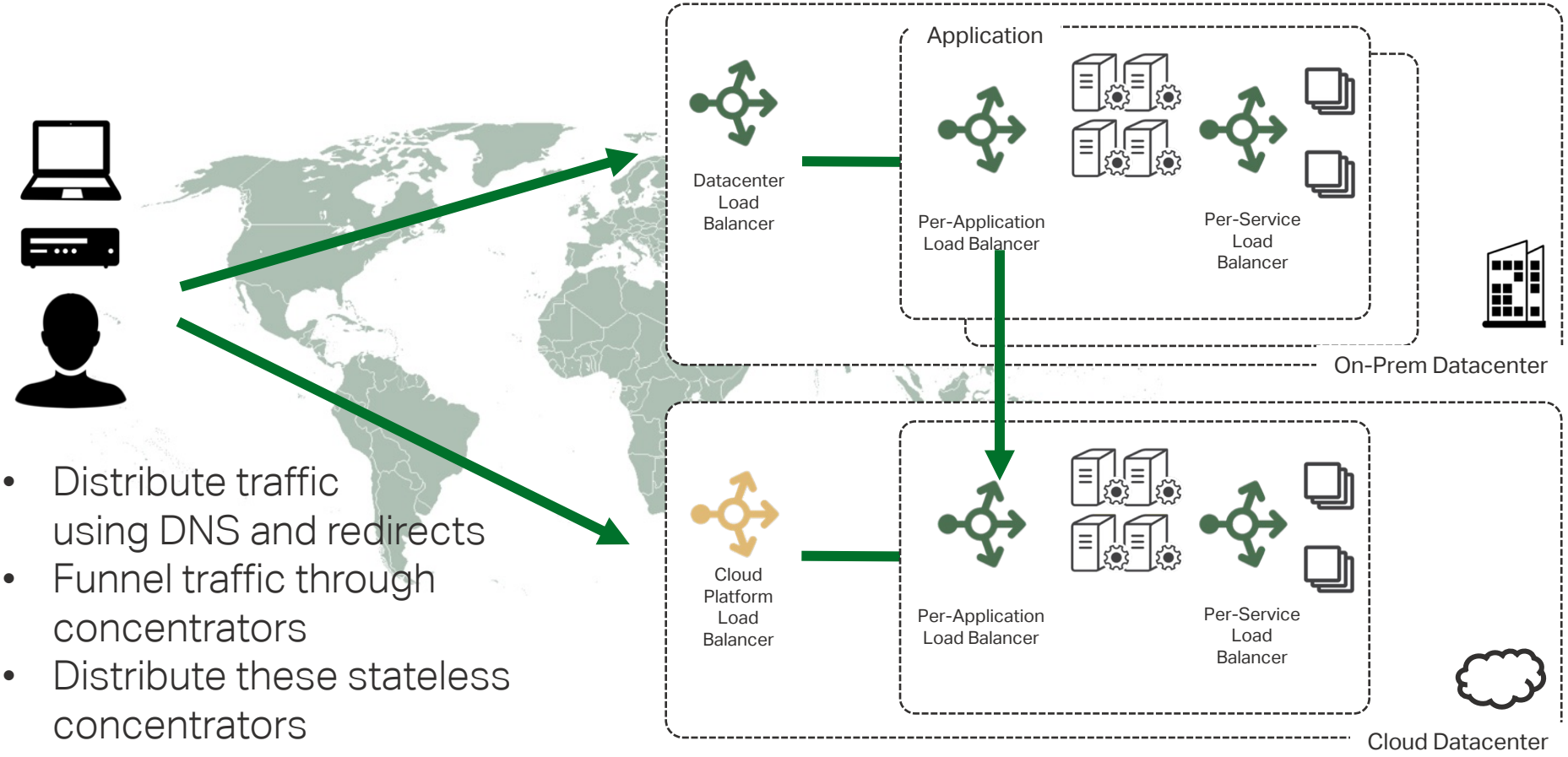

Be a hero – make changes…
all while keeping the lights on!


Change the tyres
while the car is moving

Roadmap to rearchitecting

1. Plan
2. Prepare
3. Package
4. Proceed

1. Your global architecture will be fluid



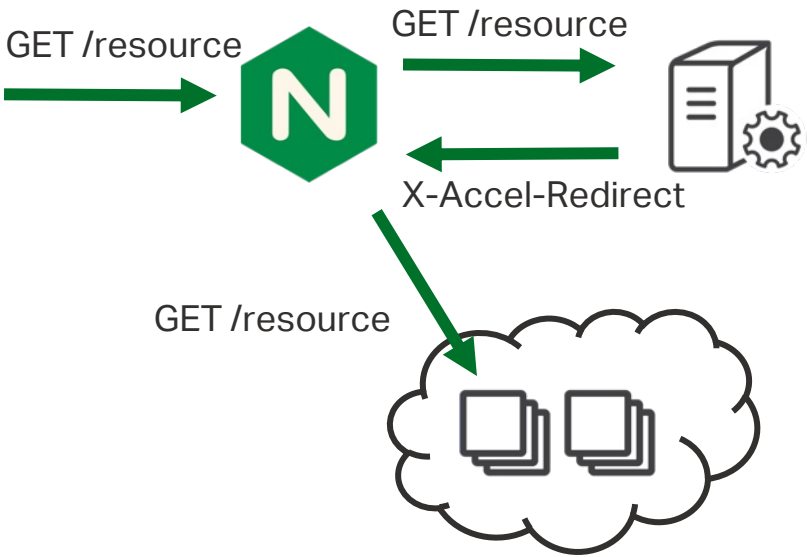
- Distribute traffic using DNS and redirects
- Funnel traffic through concentrators
- Distribute these stateless concentrators

Plan your global architecture for change

Plan how you route traffic to the correct datacenter:

1. Segment with DNS
2. Use External Redirects
   - Clients connect directly to the location of the service they are using
   - Use the proxy to push out redirects
3. Route traffic internally
4. Use X-Accel-Redirect
   - All traffic is handled through the same NGINX cluster, and internally routed to cloud

Get started with X-Accel-Redirect



- A more sophisticated alternative to a simple proxy_pass
- Request goes to local server
- Local server internally redirects to remote server

Ideal for moving content to cloud storage or serverless, while retaining NGINX-based authentication and logging .

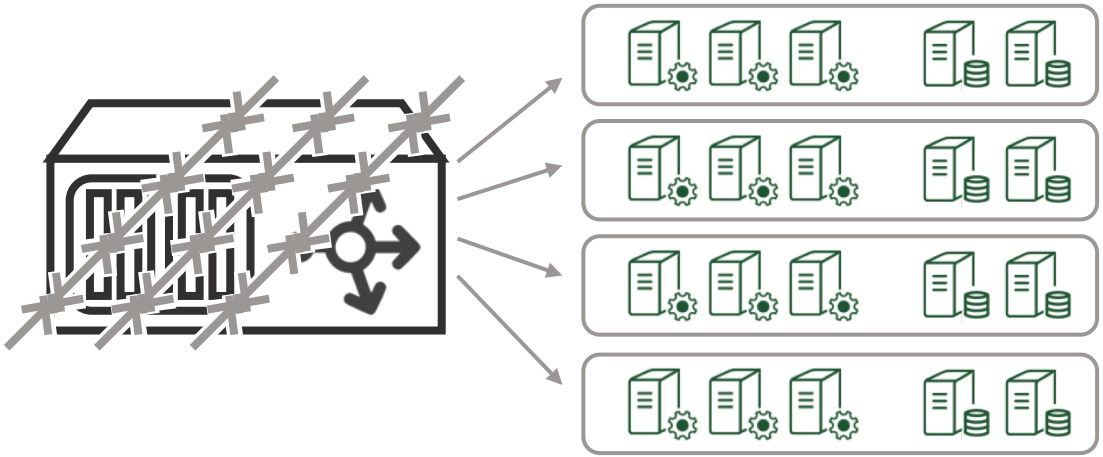
Client can never access remote server directly.

2. Prepare to execute the change

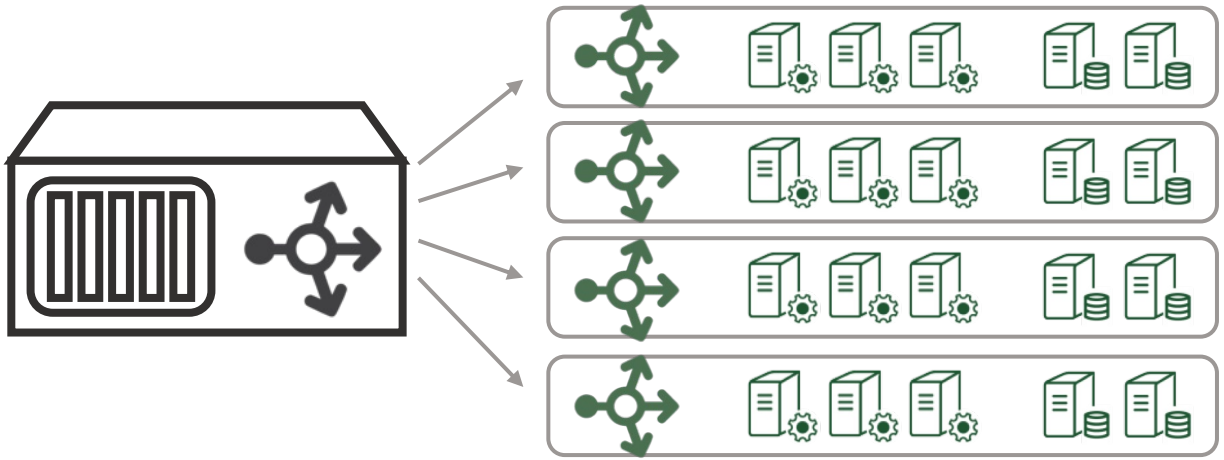
- Remove or streamline dependencies outside the core devops pipeline
  - Hardware replace
  - External business or technical processes

- Don’t underestimate the strength of “we’ve always done it this way”

Example – Hardware Replace



Example – Hardware Replace



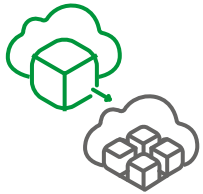
Datacenter Load Balancer

Application-specific Proxy

Example – Hardware Replace with NGINX

- Cost savings

Save more than 80% and run on commodity hardware



- Modernize

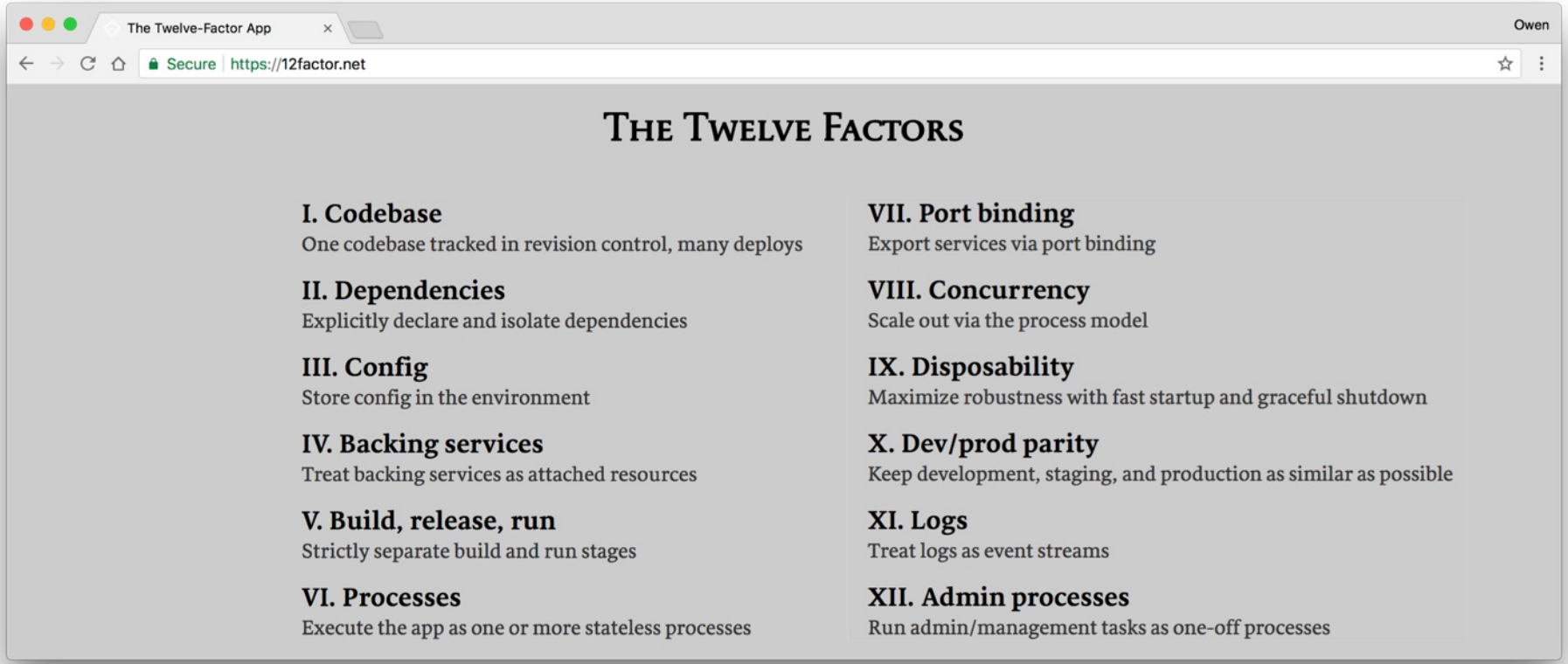
Get the flexibility to move to the cloud, microservices, Devops, and more

- No limits

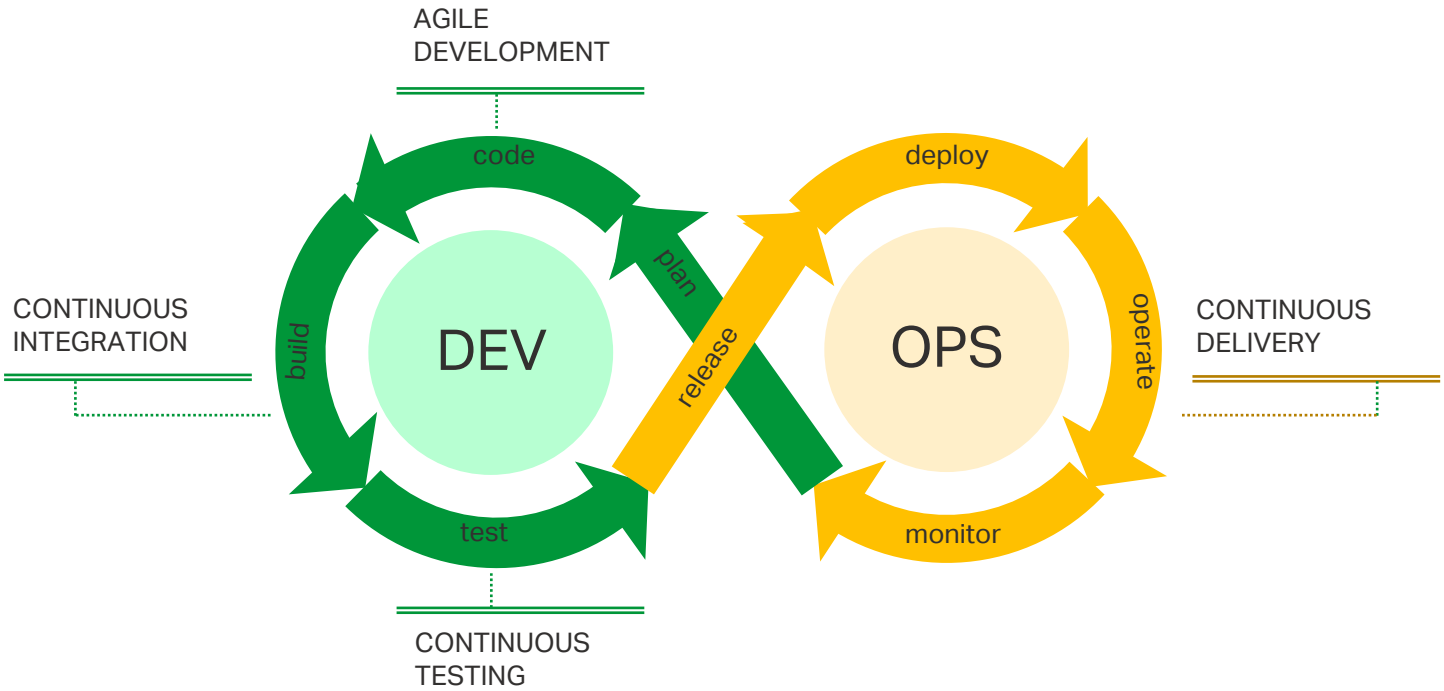
No artificial bandwidth or throughput caps to slow you down

3. Package your Applications

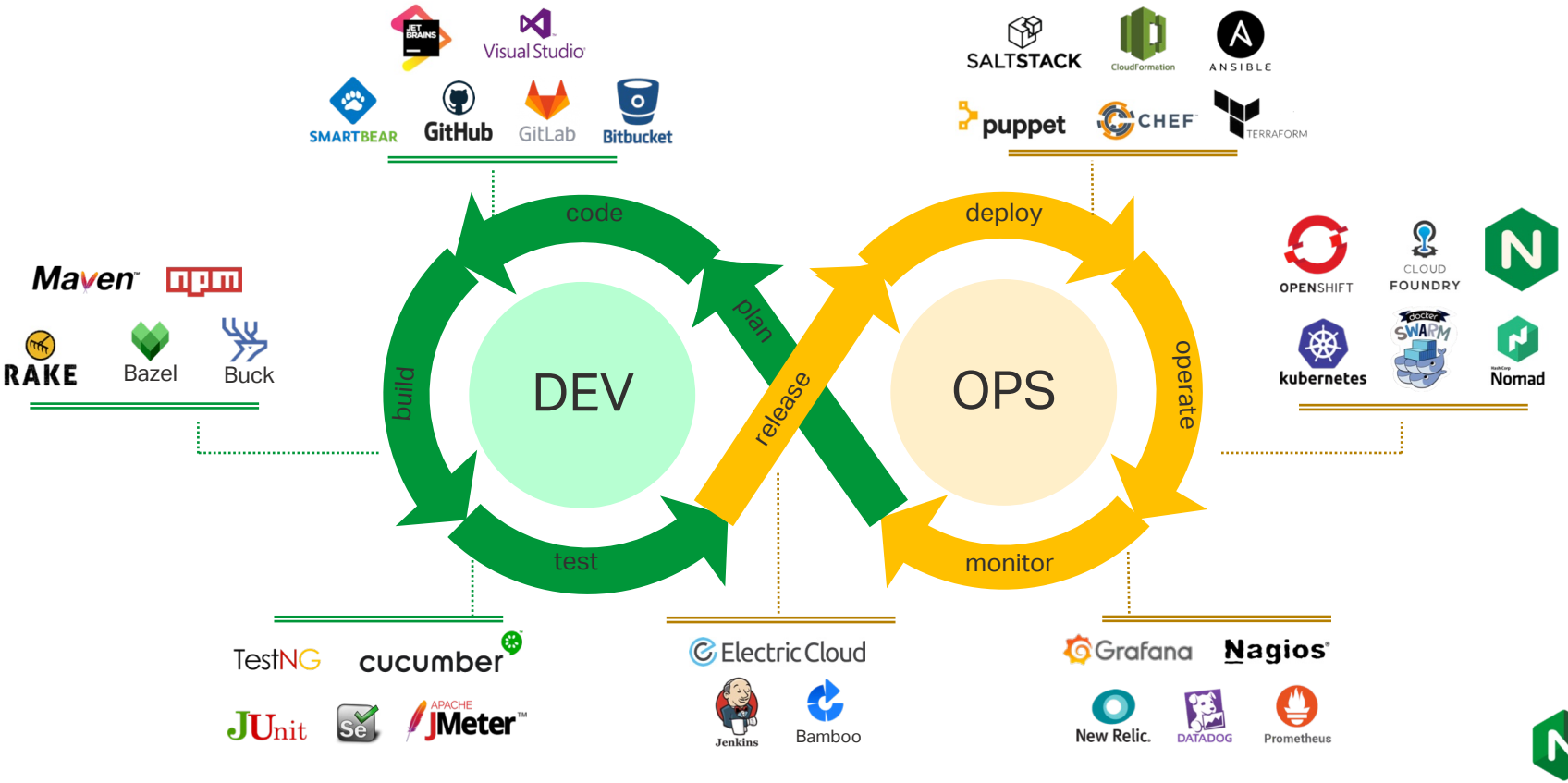
- Package as VMs or Containers; full-stack CI & CD should be your goal



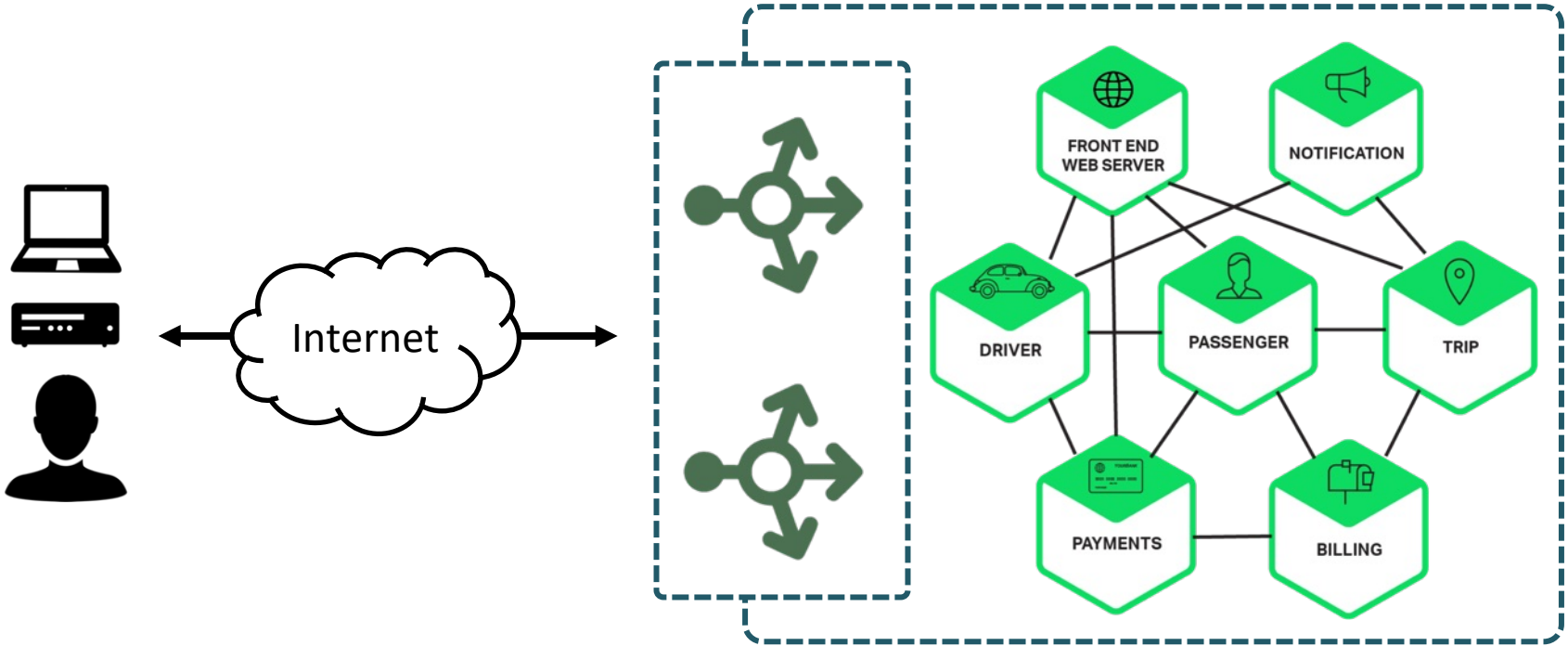
Agile Methodology



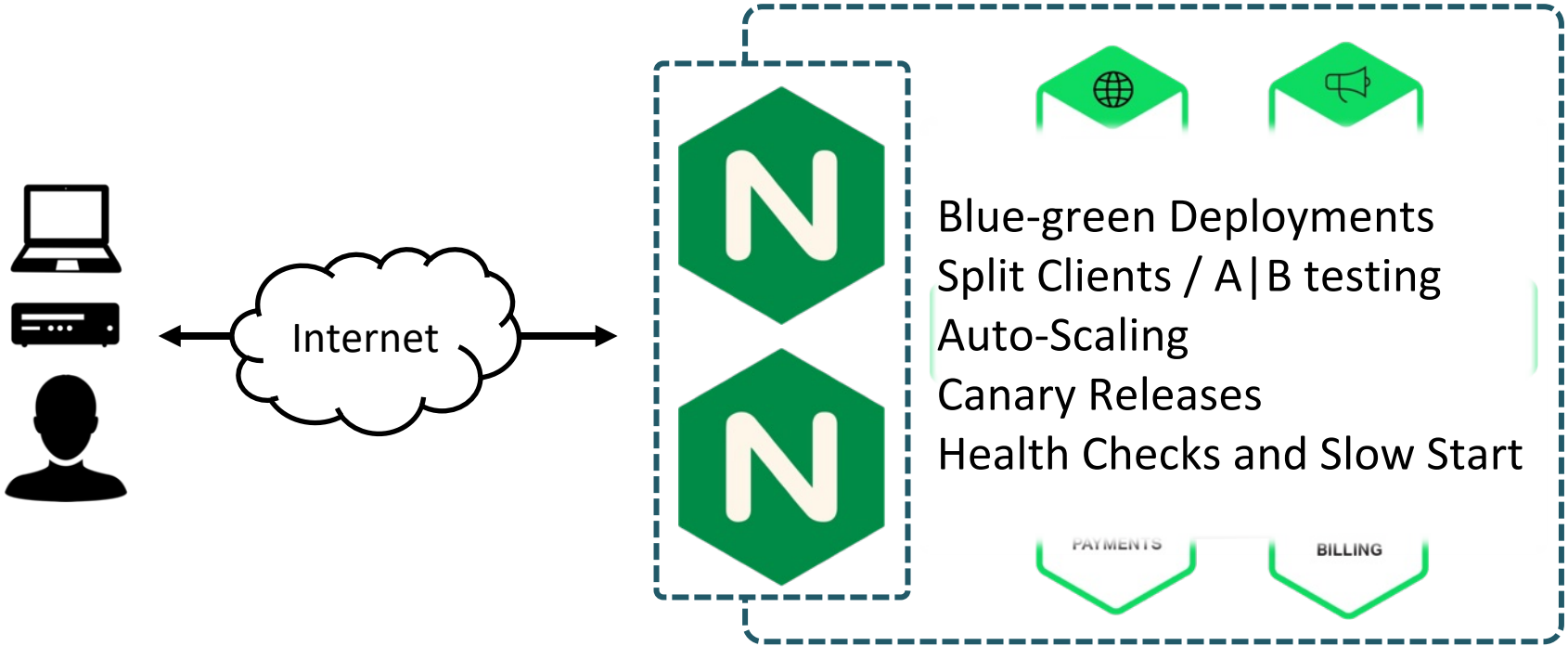
Automation Tools



4. Proceed to operate the deployment



4. Proceed to operate the deployment



Split Clients configuration

```
http {
    upstream blue_servers {
        server 10.0.0.100:3001;
        server 10.0.0.101:3001;
    }

    upstream green_servers {
        server 10.0.0.104:6002;
        server 10.0.0.105:6002;
    }

    split_clients "${remote_addr}" $appversion {
        5%      green_servers;
        *       blue_servers;
    }

    server {
        listen 80;
        location / {
            proxy_pass http://$appversion;
        }
    }
}
```

- Split traffic to multiple servers based on, for example, source IP address
- Just one example of the many ways to route traffic in NGINX:
  - By user cookie or authentication token
  - By source geography
- Forms the basis of blue-green deployments
- Monitor NGINX access logs or extended status to measure health of new, green server

Service Discovery with Consul

https://github.com/nginxinc/NGINX-Demos/tree/master/consul-template-demo

- NGINX open source can be configured using an agent that is triggered by changes to the service database

```
resolver consul:53 valid=10s;

upstream service1 {
    zone service1 64k;
    server service1.service.consul service=http
        resolve;
}
```

- NGINX Plus will look up consul in /etc/hosts/ file if using links or using Docker embedded DNS server.
- By default Consul uses this format for services: `[tag.]<service>.service[.datacenter].<domain>`

Active Health Checks

```
upstream my_upstream {
    zone my_upstream 64k;
    server server1.example.com slow_start=30s;
}
server {
    # ...
    location /health {
        internal;
        health_check interval=5s uri=/test.php
            match=statusok mandatory;
        proxy_set_header HOST www.example.com;
        proxy_pass http://my_upstream;
    }
}
match statusok {
    # Used for /test.php health check
    status 200;
    header Content-Type = text/html;
    body ~ "Server[0-9]+ is alive";
}
```

NGINX open source passively detects application failures

NGINX Plus provides “Active Health Checks”

- Polls `/URI` every 5 seconds
- If response is not `200`, server marked as failed
- If response body does not contain “**`ServerN is alive`**”, server marked as failed
- Recovered/new servers will slowly ramp up traffic over 30 seconds


BuzzFeed
Move to Microservices
"As we moved to microservices we realized that we needed a much smarter way of routing pages to our applications. The big benefits of NGINX Plus were firstly the support, the DNS configuration which allowed us to use sophisticated services in AWS, and the metrics told us which servers were failing."
- John Cleveley, Senior Engineering Manager

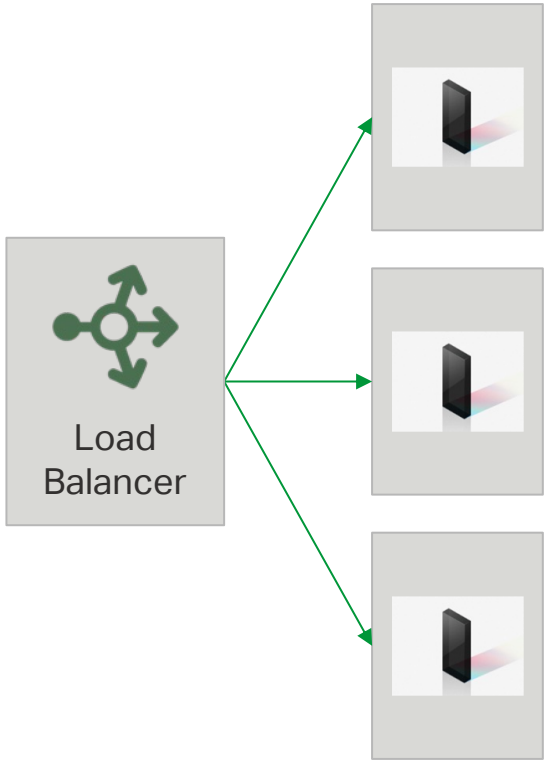
Two proven microservices delivery patterns

1.

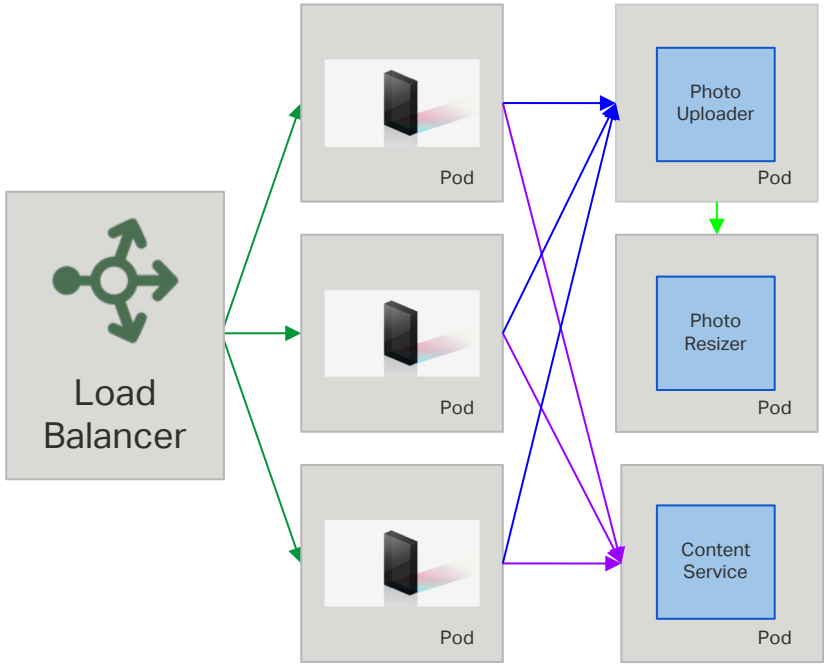
Managing north-south traffic with an Ingress Controller

Starting from your Monolith…

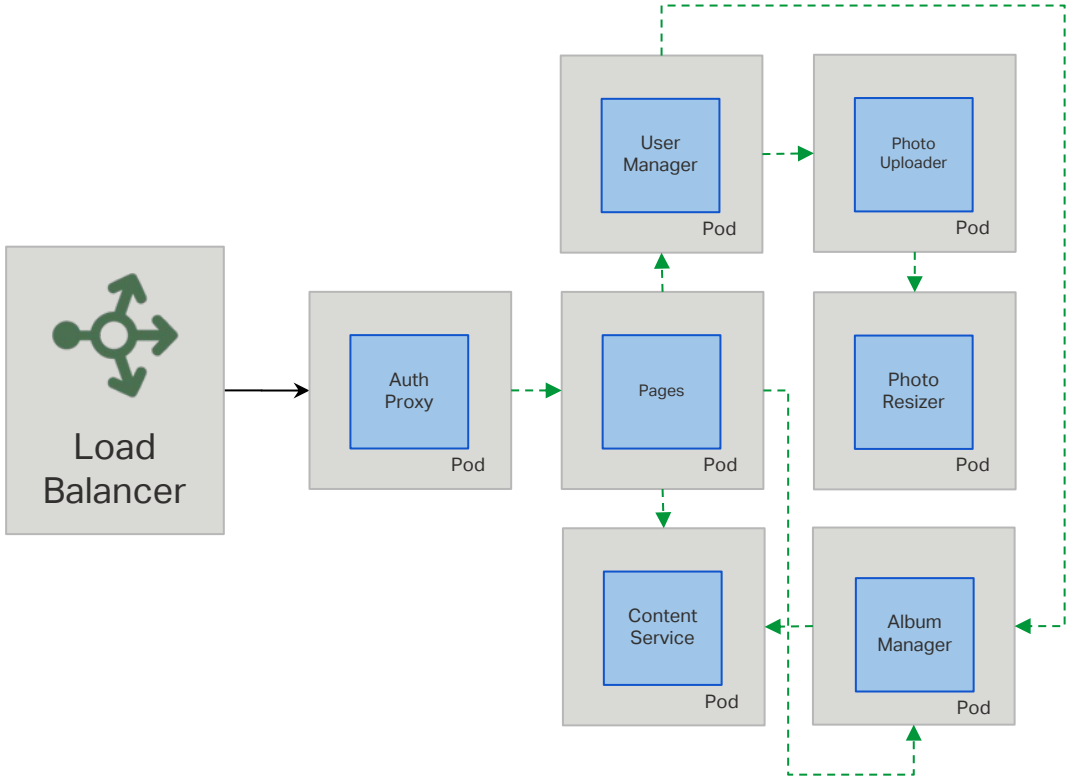
1. Containerise your Monolith



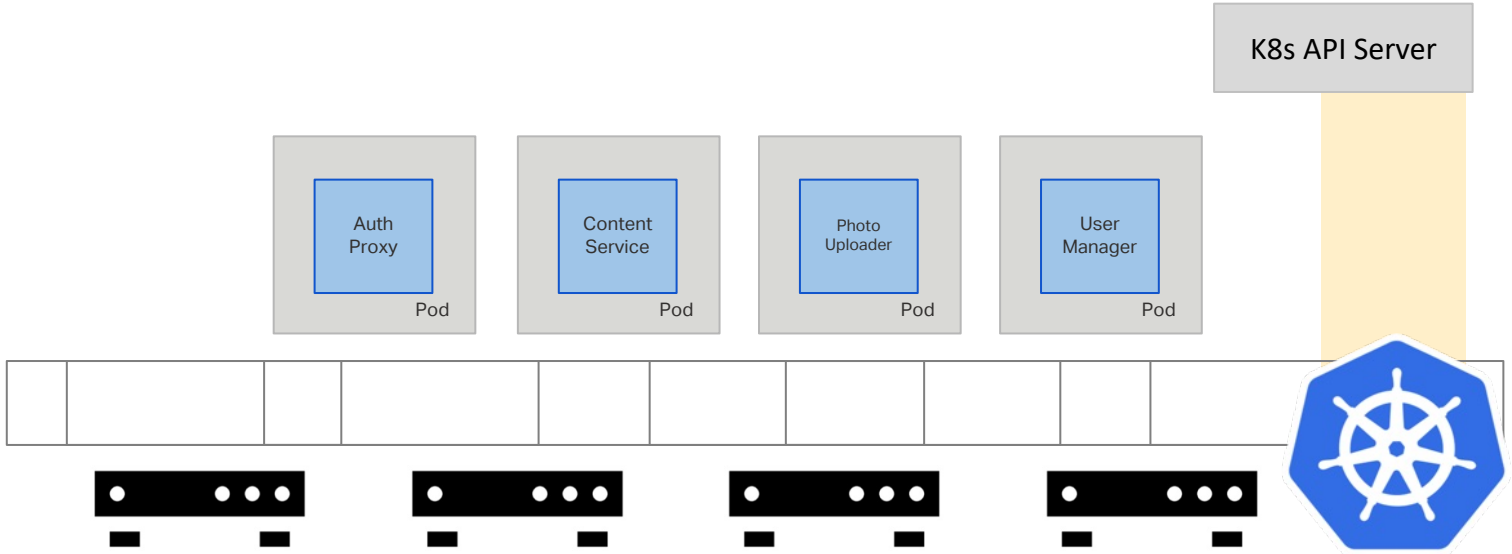
2. Decompose your Monolith



3. Rearchitect your Monolith



Deploy on, for example, Kubernetes



Kubernetes Ingress Resource

```
1. apiVersion: extensions/v1beta1
2. kind: Ingress
3. metadata:
4.   name: hello-ingress
5. spec:
6.   tls:
7.   - hosts:
8.     - hello.example.com
9.     secretName: hello-secret
10.  rules:
11.  - host: hello.example.com
12.    http:
13.      paths:
14.      - path: /
15.        backend:
16.          serviceName: hello-svc
17.          servicePort: 80
```

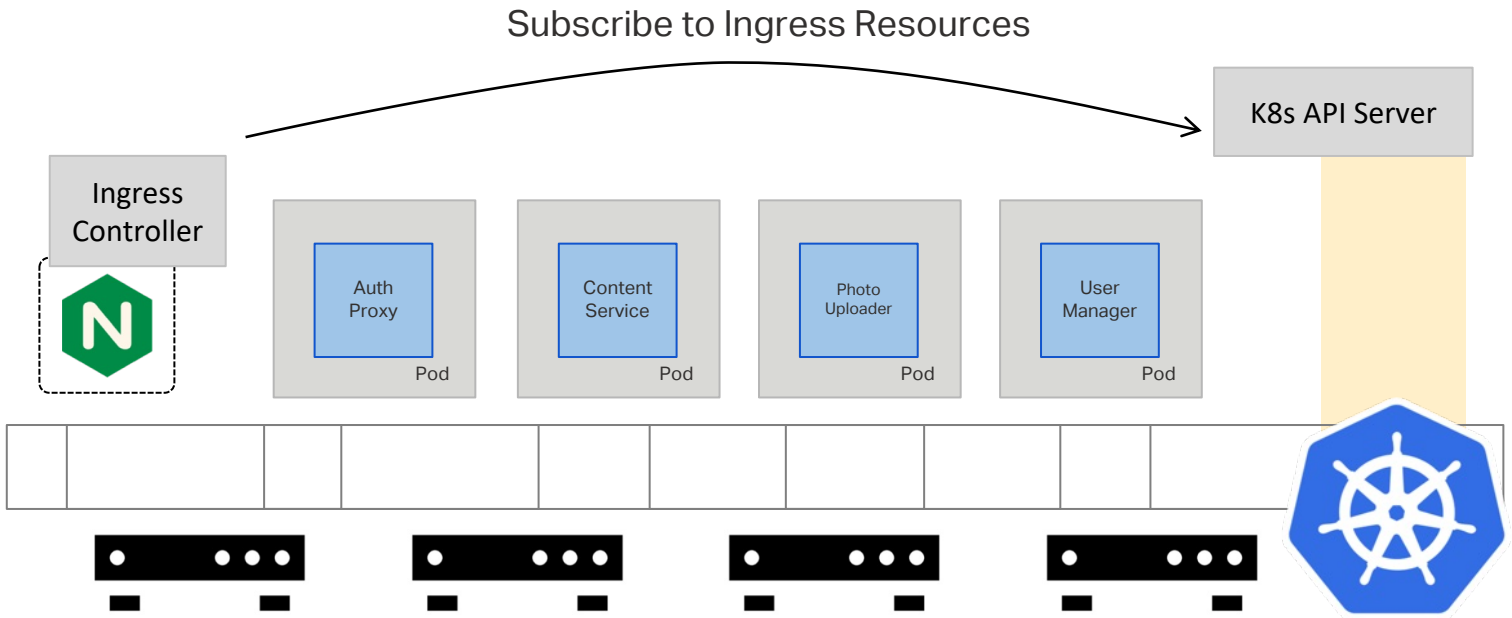
Ingress:

- Built-in Kubernetes resource
- Automates Configuration for an edge load balancer (or ADC)

Ingress features:

- L7 routing based on the host header and URL
- TLS termination

Application Delivery on Kubernetes



Limitations of the Kubernetes Ingress Resource

```
1. kind: Ingress
2. metadata:
3.   name: hello-ingress
4. spec:
5.   tls:
6.   - hosts:
7.     - hello.example.com
8.     secretName: hello-secret
9.   rules:
10.  - host: hello.example.com
11.    http:
12.      paths:
13.      - path: /
14.        backend:
15.          serviceName: hello-svc
16.          servicePort: 80
```

Only does:

- Routing on the host header and URL
- TLS termination

What about:

- Session persistence
- JWT validation
- Rewriting the URL of a request
- Enabling HTTP/2
- Choosing a load balancing method
- Changing the SSL parameters
- …

Extending the Kubernetes Ingress Resource

Annotations

- Vendor-specific configuration settings

```
1. apiVersion: extensions/v1beta1
2. kind: Ingress
3. metadata:
4.   name: hello-ingress
5.   annotations:
6.     nginx.org/lb-method: "ip_hash"
```

Configuration Snippets

- Embed NGINX configuration directives directly into config contexts

```
1. apiVersion: extensions/v1beta1
2. kind: Ingress
3. metadata:
4.   name: hello-ingress
5.   annotations:
6.     nginx.org/location-snippets: |
7.       proxy_set_header X-Custom-Header-1 foo;
8.       proxy_set_header X-Custom-Header-2 bar;
```

or… Edit Ingress Controller template directly

Ingress Controller: where to find out more

- GitHub: https://github.com/nginxinc/kubernetes-ingress

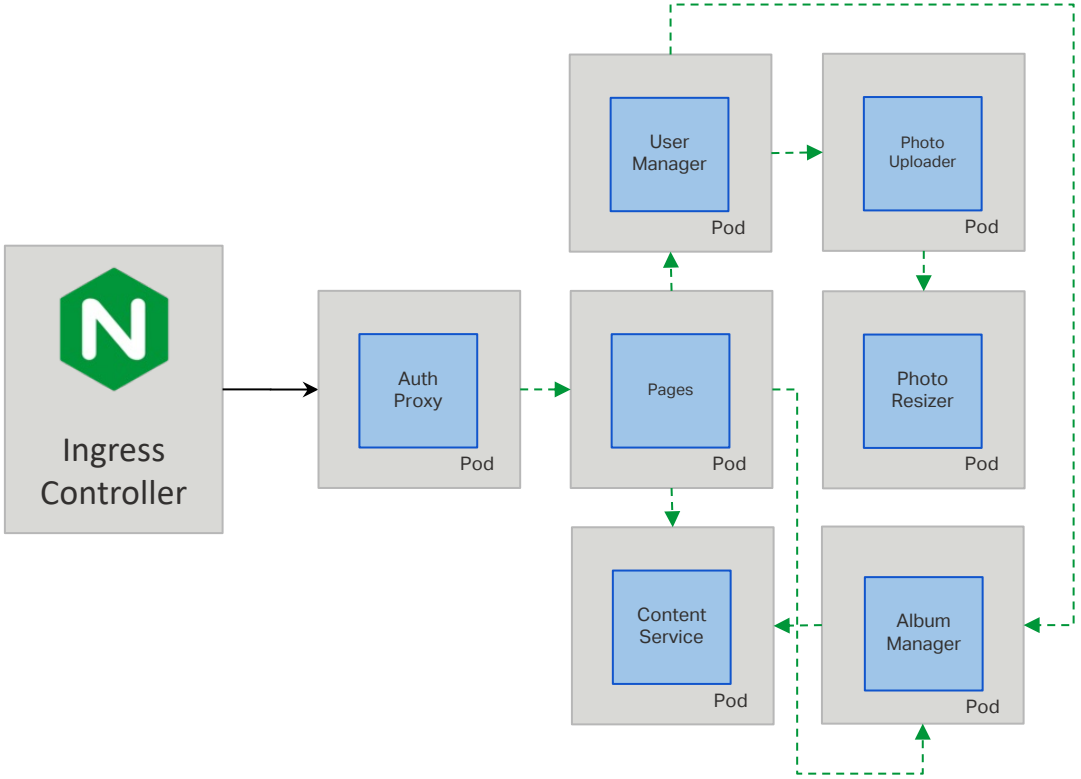
- NGINX Documentation:
  - https://www.nginx.com/blog/introducing-nginx-kubernetes-ingress-controller/

- NGINX offers a fully-supported Ingress Controller implementation based on NGINX Plus and the Open Source IC
  - Detailed metrics
  - Faster, more reliable reloads
  - Full support

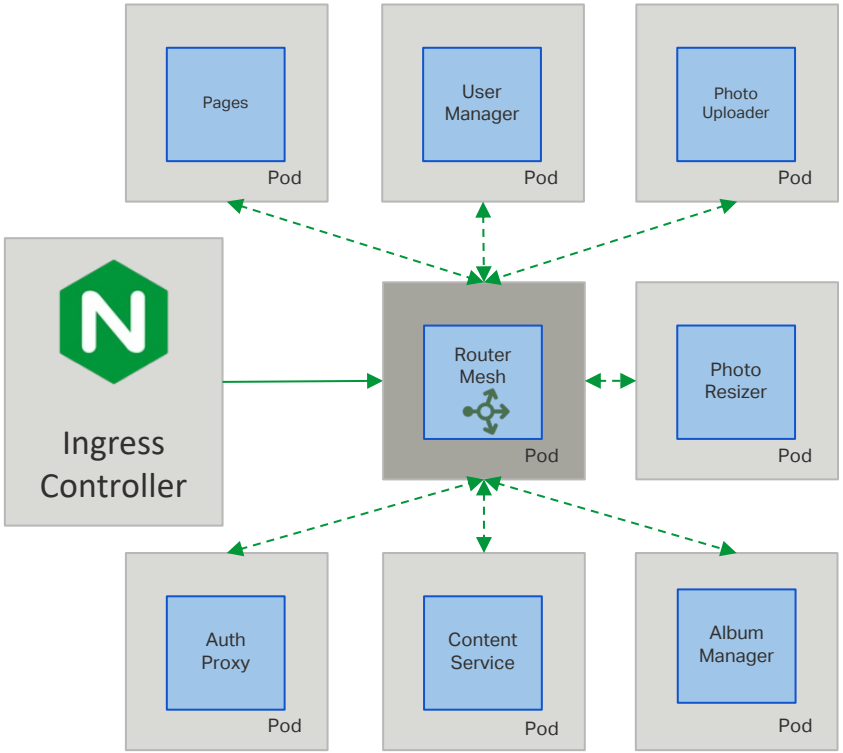
2.

Managing east-west traffic with an internal router

Limitations of the Ingress Controller alone

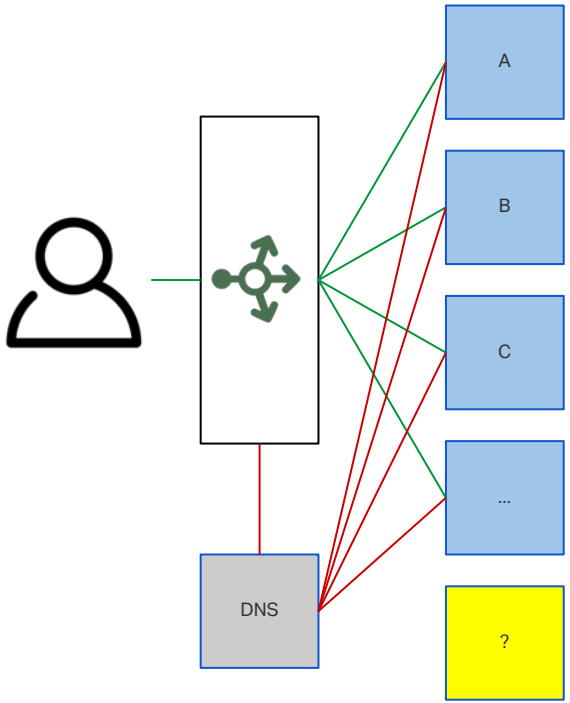


Introducing the Router Mesh model



Router Mesh relies on Service Discovery

- Required when:
  - New Services are added
  - Instances of existing services are added
- Proxies are configured using triggers:
  - Ansible Roles
  - Consul templates



Router Mesh relies on Service Discovery

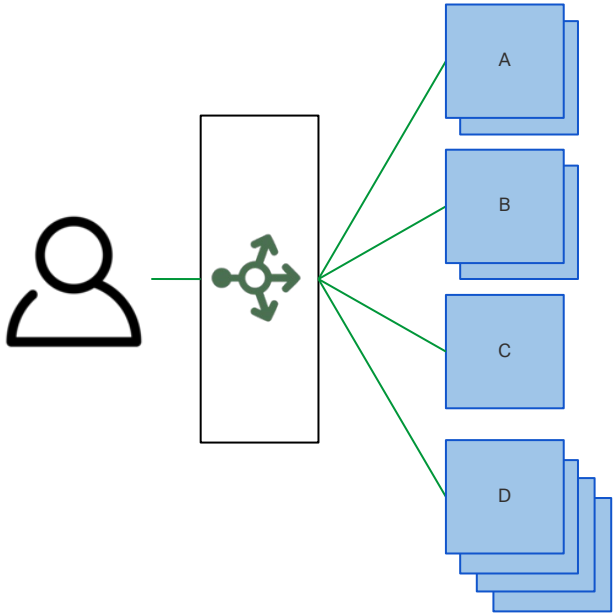
- Required when:
  - New Services are added
  - Instances of existing services are added
- Proxies are configured using triggers
  - Ansible Roles
  - Consul templates
- NGINX Plus uses DNS
  - Vanilla DNS server
  - Consul, kube-dns, Mesos-dns

```
resolver consul:53 valid=10s;

upstream service1 {
    zone service1 64k;
    server service1.service.consul
        service=http resolve;
}
```

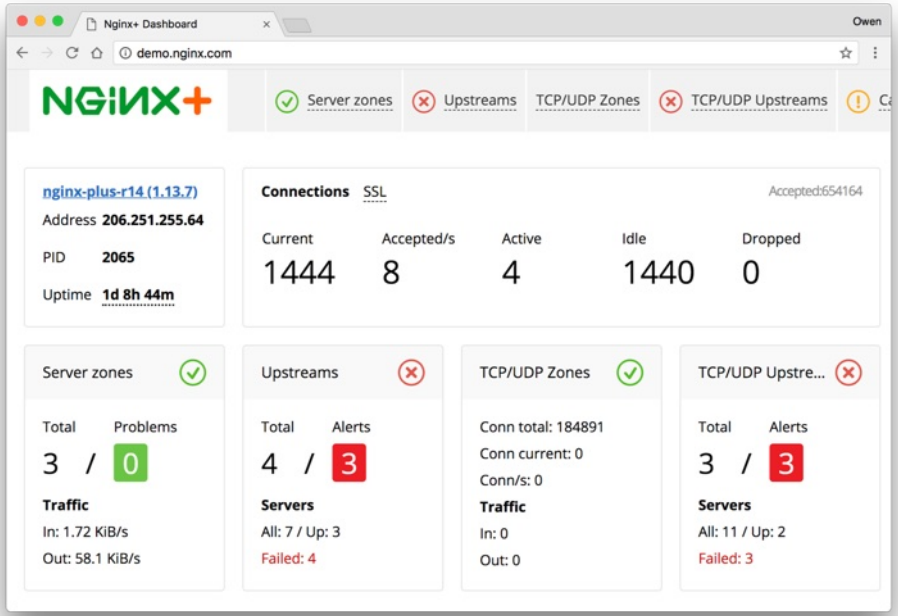
Router Mesh provides internal Load Balancing

- Provides:
  - Scalability
  - High-Availability
  - Circuit-breaker pattern



Router Mesh provides internal Load Balancing

- Provides:
  - Scalability
  - High-Availability
  - Circuit-breaker pattern
- NGINX Plus adds:
  - Application-level health checks
  - Slow-start on new server
  - Extended Status telemetry

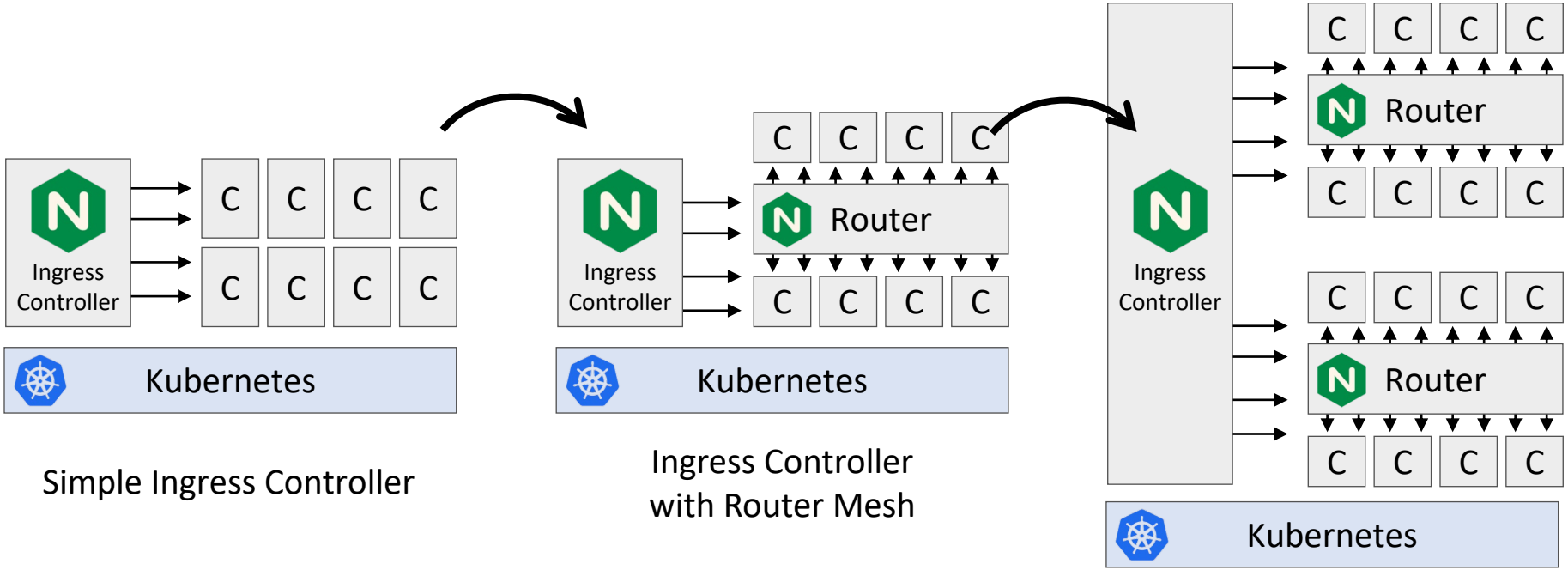


Roadmap to rearchitecting

- **Plan** – for parallelism
- **Prepare** – for the process of change
- **Package** – positioning yourself for CI/CD agility
- **Proceed** – using a proxy to orchestrate and insulate changes

- **Solutions**: NGINX Ingress Controller, internal Router Mesh

Non-disruptive Microservices Adoption Roadmap



“All problems in computer science can be solved by another layer of indirection

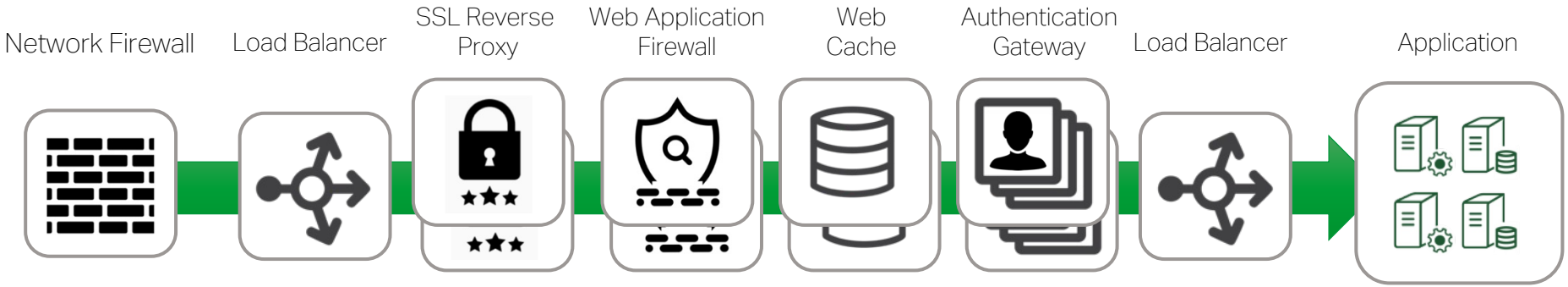
--- David Wheeler, FRS

“All problems in computer science can be solved by another layer of indirection

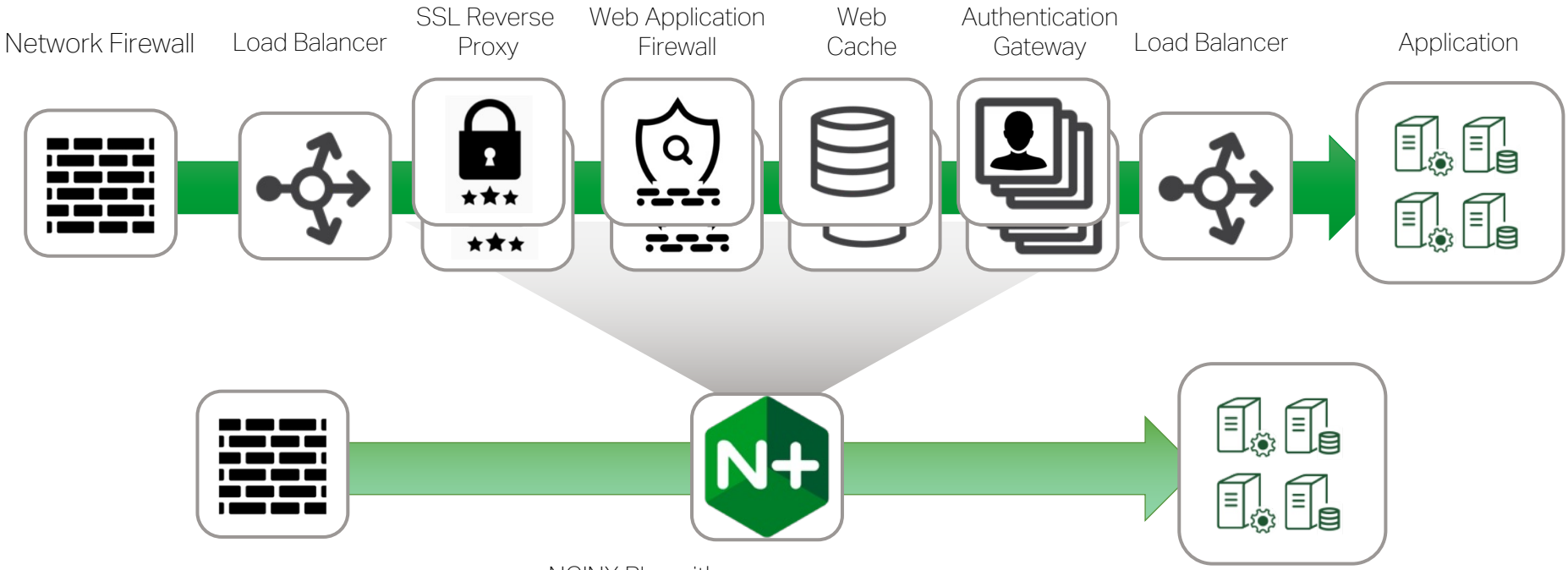
...except too many levels of indirection”

--- David Wheeler, FRS

Very complex application-delivery pipelines



NGINX Plus minimizes the amount of indirection



NGINX Plus with:

- ModSecurity Web Application Firewall
- OAuth2 and JWT validation
- Third-party Certified Authentication Modules

Where to next?

- You can get NGINX from nginx.org, your OS vendor or favourite PPA
- Find us on floor 3, near the keynote theatre
- Interested in getting more from NGINX:
  - NGINX Plus developer subscription
- Thank you!

Thank you

owen@nginx.com