イチから体験!
NGINX ハンズオントレーニング
～ 応用編 ～

東京エレクトロン デバイス株式会社

東京エレクトロン　デバイスについて

- F5の日本法人が出来る前からの代理店
- 幅広い取り扱いラインナップ
  - NGINX Plus
  - NGINX App Protect WAF
  - NGINX Management Suite
  - NGINX Ingress Controller
  - NGINX Service Mesh
- F5国内販売額7年連続No.1の一次代理店

お気軽にご相談ください（お問合せフォームより）

https://cn.teldevice.co.jp/product/f5-nginx/

Agenda

- NGINX Plus について
- ハンズオン環境の準備
- 流量制御 (Rate Limit/Connection Limit)
- ロードバランシングメソッド
- アクティブヘルスチェック ★
- セッションパーシステンス ★
- サービスディスカバリ ★

NGINX Plus でのみ利用可能

OSS版だけではありません

NGINX Plus

NGINX のエンタープライズ向け製品

- OSS を拡張した機能
  - 負荷分散アルゴリズム、セッション維持、アクティブヘルスチェック、DNS ディスカバリ
  - 冗長構成、OpenIDやJWTによる認証
- 弊社 および メーカーによる高品質なサポート
  - 認定サードパーティモジュールのサポート

NGINX ソリューション

NGINX App Protect

アプリケーションに特化したWAFとDDoS防御
NGINX Plus に統合可能でどこでも利用可能な新しいWAF/DDoS

低いレイテンシと高いスループット

NGINX Ingress Controller

Kubernetes 環境のIngress トラフィック管理
KubernetesのIngress リソースとしてNGINX Plusを使用

App Protect をアドオンすることで、Ingress にWAF/DDoSを追加

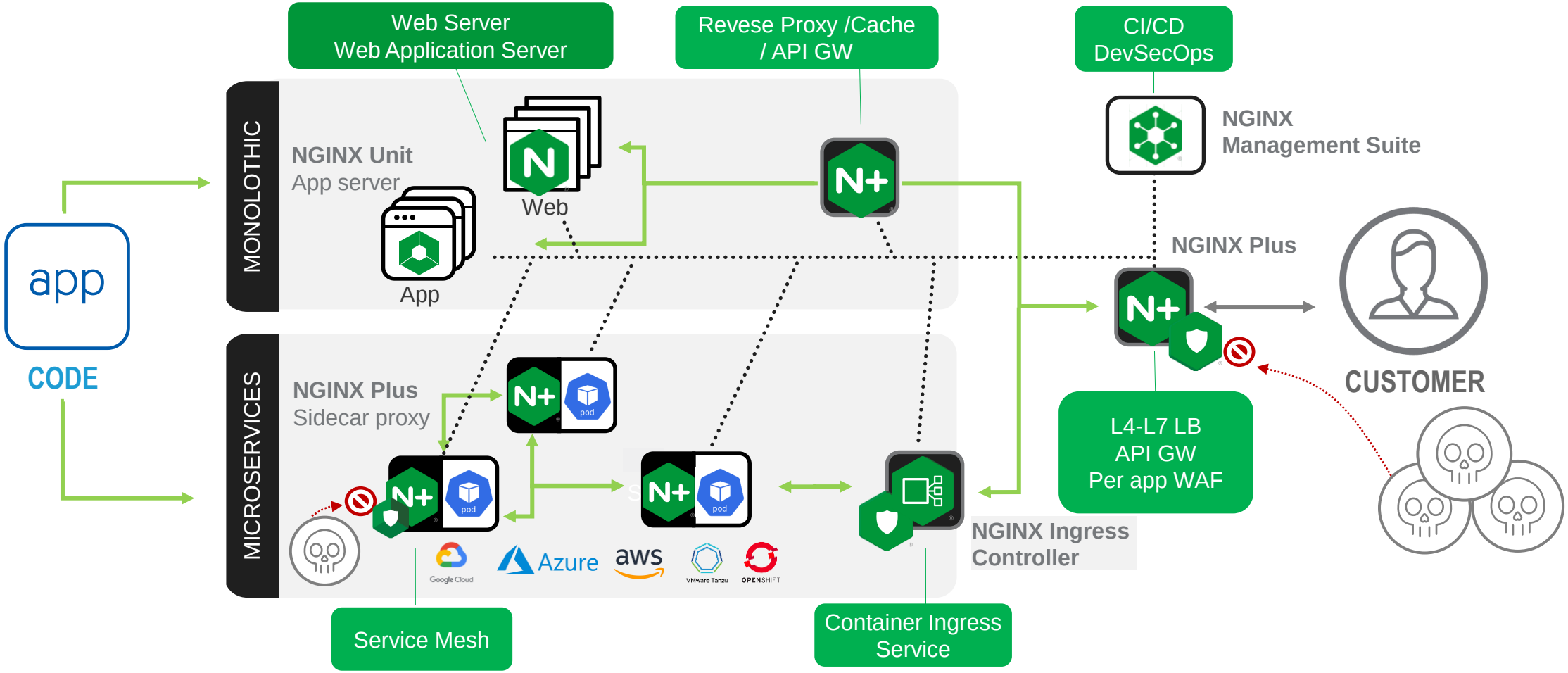
NGINX Management Suite

NGINX の統合管理ソフトウェア
GUIでのNGINX設定管理やモニタリングが可能

様々なNGINX製品をまとめて管理可能

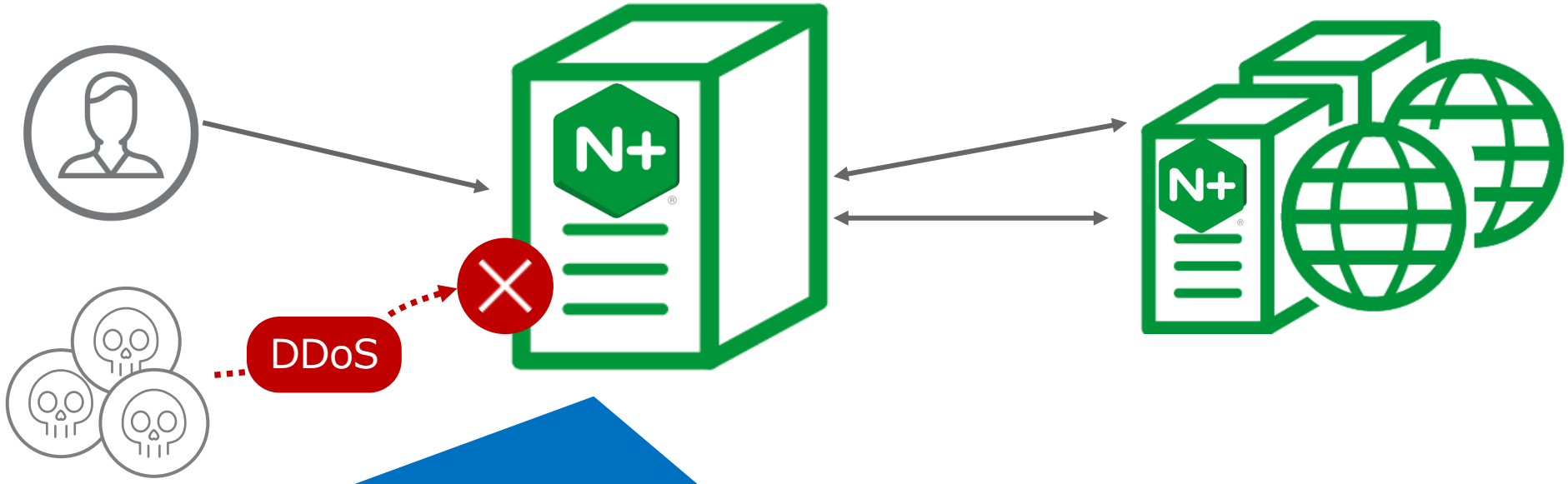
NGINX Product Portfolio

高頻度のアプリケーションリリースを実現する、
高い柔軟性・効率・品質のアプリケーション基盤



流量制御

Rate Limit



外部から大量のアクセスを防ぎ、Webサーバ・アプリケーションサーバの想定以上のリソース消費を抑え、プラットフォーム全体を安全に・安定して動作させることが可能となります

Rate Limit

limit_req_zone　httpコンテキスト内で共有メモリやRate Limitの条件をパラメータで指定

書式：limit_req_zone <key> zone=<共有メモリ名>:<共有メモリサイズ> rate=<制限するレート>;

- <key>：レート制限を行う通信を識別する条件。例えば、クライアントIPを識別の条件としてレート制限を実行する場合、$binary_remote_addr変数を指定する。この場合、それぞれのユニークなIPアドレスからのリクエストに対しrateのパラメータで指定したレート制限を実施する
- zone：レート制限のステータスを保持する共有メモリを指定する。この共有メモリを用いて、NGINXのワーカプロセス間でステータスをシェアする
- rate：レート制限を行うためのレートの最大値を指定する。NGINXでは各リクエストについて指定した値のレートで通信が行われているかミリ秒単位での評価を行う

limit_req　対象となる通信に対しレート制限を有効にする

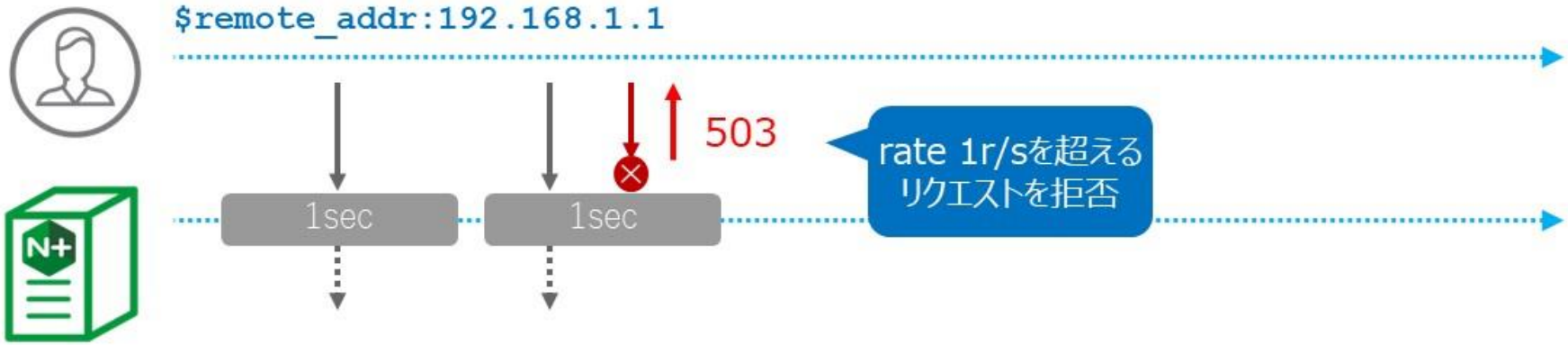
書式：limit_req zone=<共有メモリ名> ;

- <共有メモリ名>：limit_req_zoneで予め作成した、共有メモリ名を指定

Rate Limit

Rate Limit 設定例：rate

```
limit_req_zone $remote_addr zone=req:1M rate=1r/s;
server {
    listen 80;
    location / {
        limit_req zone=req;
        proxy_pass http://my_backend;
    }
```



CONFIDENTIAL

Rate Limit

Rate Limit 設定例：burst

```
limit_req_zone $remote_addr zone=req:1M rate=1r/s;
server {
    listen 80;
    location / {
        limit_req zone=req burst=2;
        proxy_pass http://my_backend;
    }
```

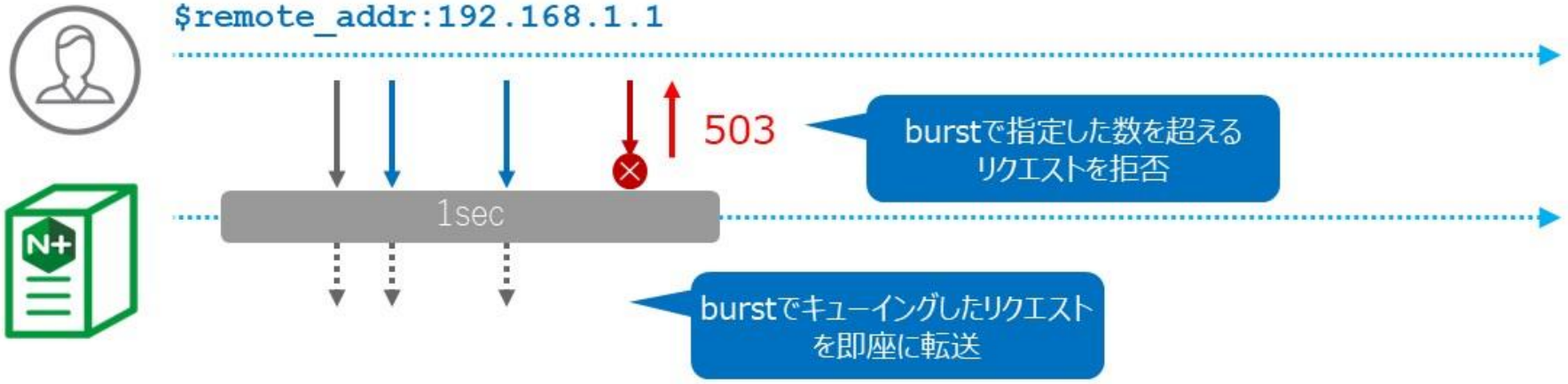
指定したレートを超える通信を、burstで指定した数だけキューイング

CONFIDENTIAL

Rate Limit

Rate Limit 設定例：nodelay

```
limit_req_zone $remote_addr zone=req:1M rate=1r/s;
server {
    listen 80;
    location / {
        limit_req zone=req burst=2 nodelay;
        proxy_pass http://my_backend;
    }
```

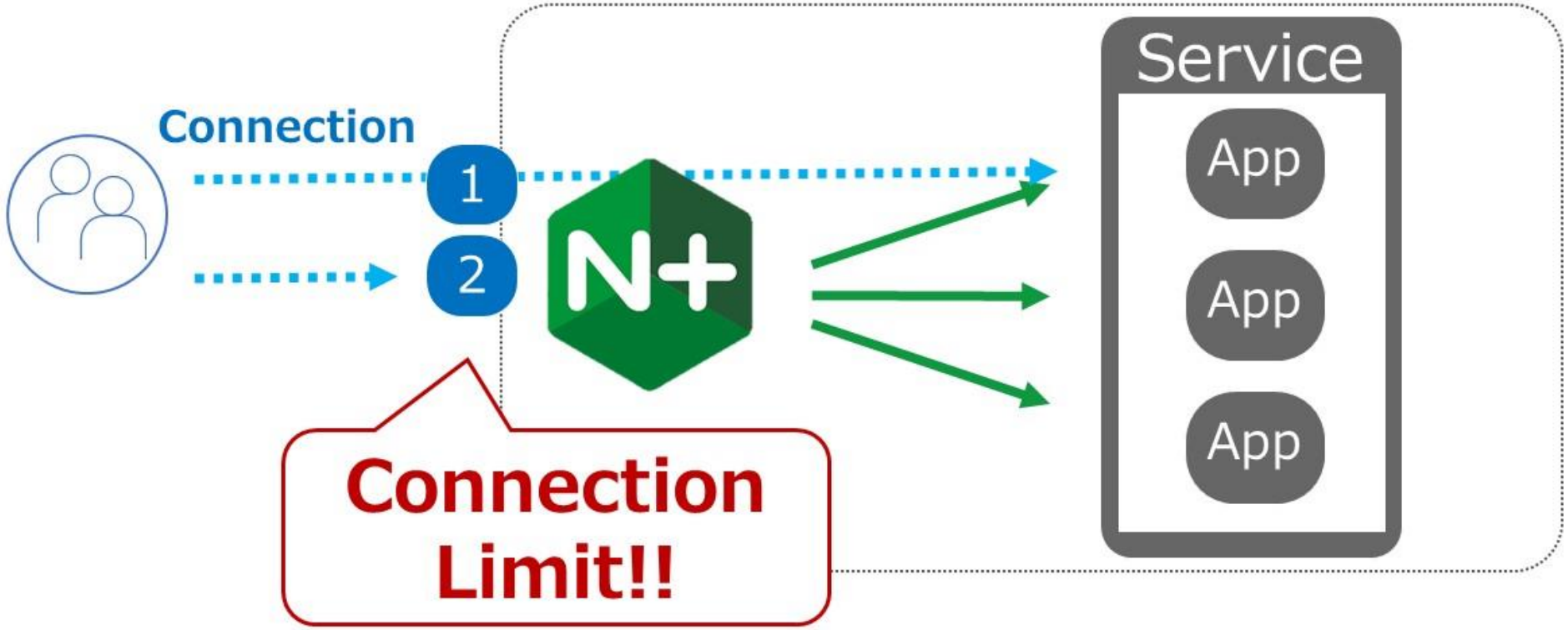


Rate Limit

Rate Limit 設定例：delay

```
limit_req_zone $remote_addr zone=req:1M rate=1r/s;
server {
    listen 80;
    location / {
        limit_req zone=req burst=2 delay=1;
        proxy_pass http://my_backend;
    }
```

Connection Limit



Connection Limit

limit_conn_zone　httpコンテキスト内で共有メモリを指定

書式：limit_conn_zone <key> zone=<共有メモリ名>:<共有メモリサイズ>;

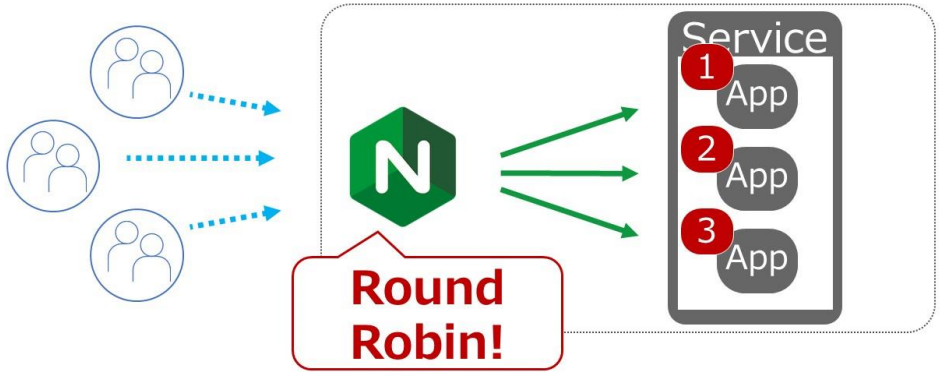
- <key>：レート制限を行う通信を識別する条件。クライアントIPを識別の条件とする場合、binary_remote_addr 変数を指定する。この場合、それぞれのユニークなIPアドレスからのリクエストに対しrateのパラメータで指定したレート制限を実施する
- zone：レート制限のステータスを保持する共有メモリを指定する。この共有メモリを用いて、NGINXのワーカプロセス間でステータスをシェアする

limit_conn　対象となる通信に対しコネクション制限を有効にする

書式：limit_conn zone=<共有メモリ名> <同時接続数>;

- <共有メモリ名>：limit_req_zoneで予め作成した、共有メモリ名を指定
- <同時接続数>：コネクションの最大同時接続数を指定する

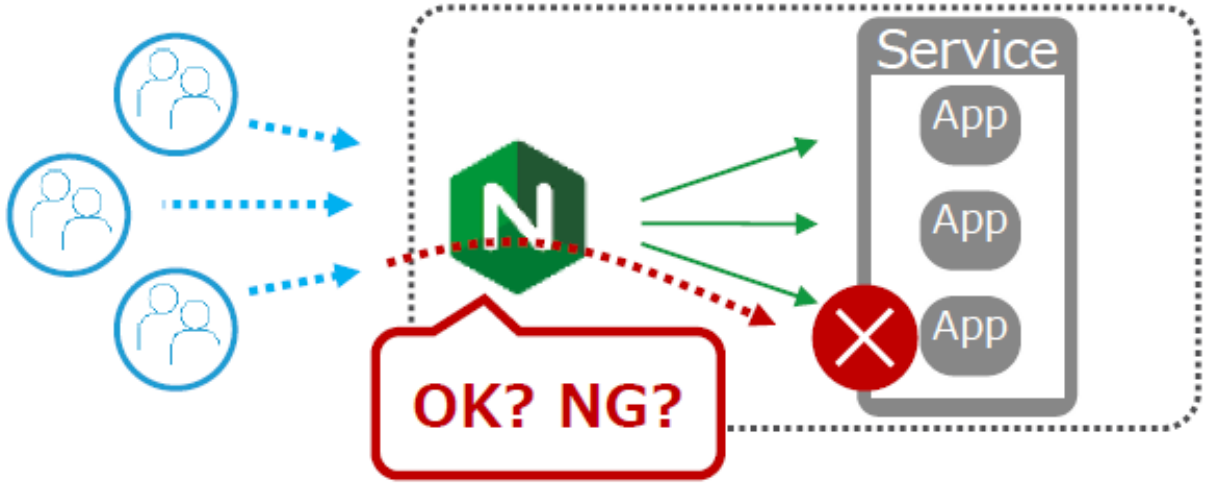
Load Balancing Method



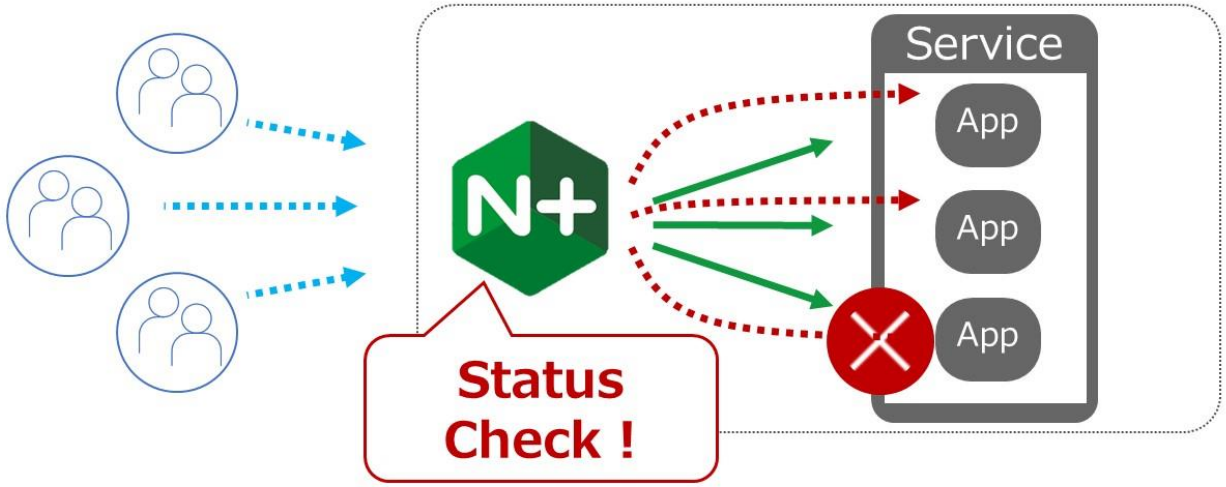
Method	振り分け方法	備考
Round Robin	配下のサーバーに均等に振り分ける	Default 設定
ip_hash	送信元IPアドレスが同じであれば、同じサーバーに振り分ける	
Hash <key>	指定したパラメーターのハッシュ値で振り分ける	
Least_conn	サーバーとのアクティブな connection が一番少ないサーバーを選択します	
least_time	平均レイテンシーとアクティブな接続数をベースに振り分け先サーバーを決定する	NGINX Plus で利用可

Active Health Check

Passive HealthCheck



Active HealthCheck



Active Health Check

```
upstream  backend  {
        zone backend 64k;
        server backend1.example.com;
        server backend2.example.com;
}
match hc_ok  {
         status 200;
         body ~ ”OK";
}

server {
        listen 80;
        location / {
          proxy_pass http://backend;
          health_check interval=10 fails=3 passes=2 uri=/echo match=hc_ok;
      }
}
```

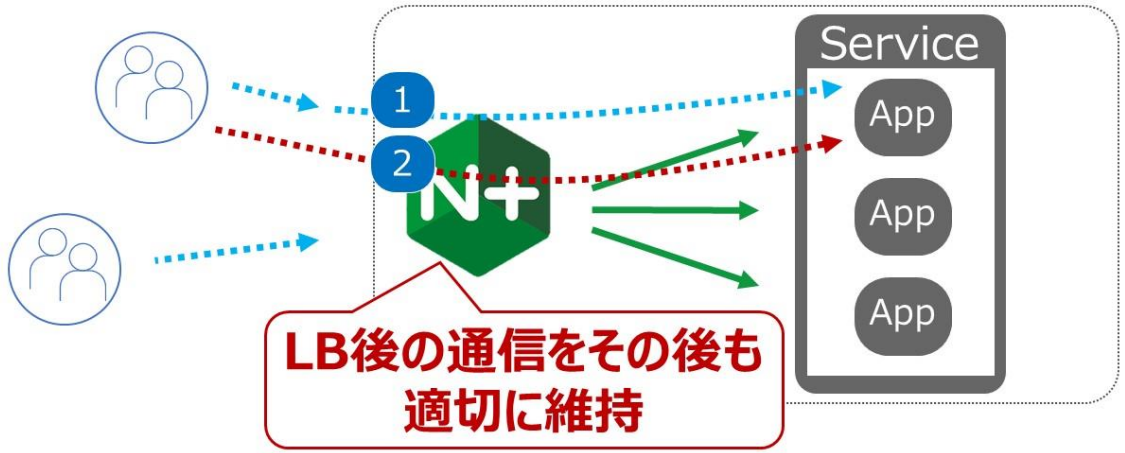
・・Monitrを成功と判定する際の条件を指定
ステータスコード「200」、レスポンスに「OK」と含む場合、「up」と判定

10秒ごとにuri /echo に対し、hc_okの条件でモニターを実施
3回失敗した場合に「Fail」と判定、2回成功した際に「up」と判定

セッションパーシステンス

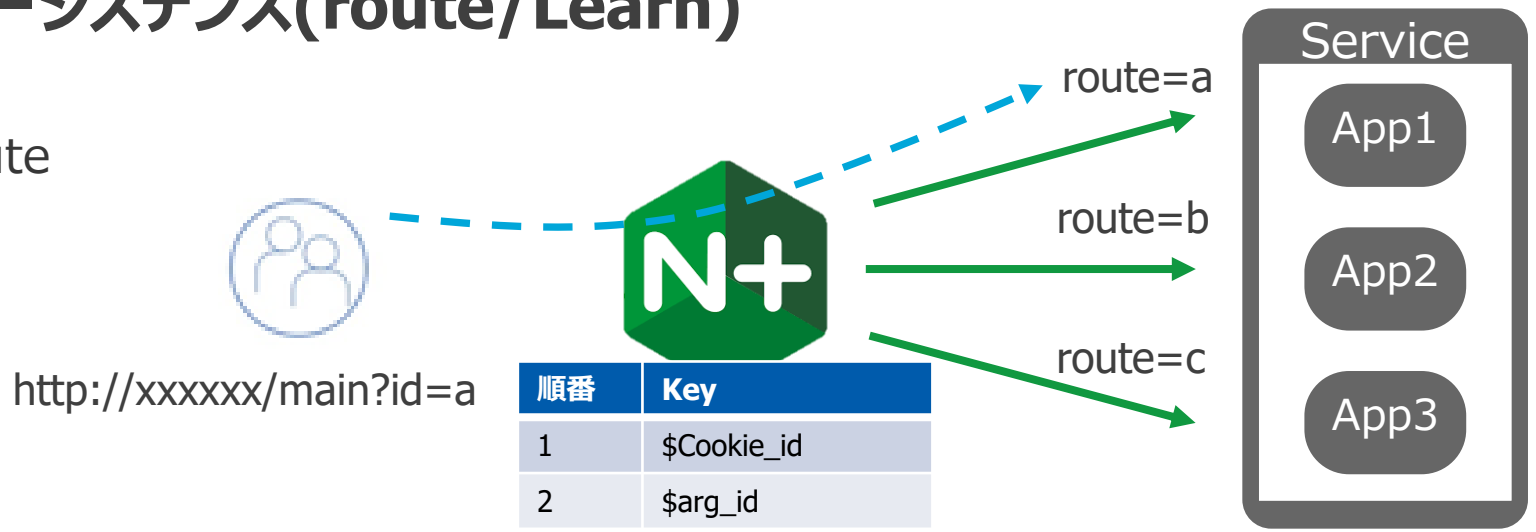
NGINX OSS

NGINX Plus



セッションパーシステンス(route/Learn)

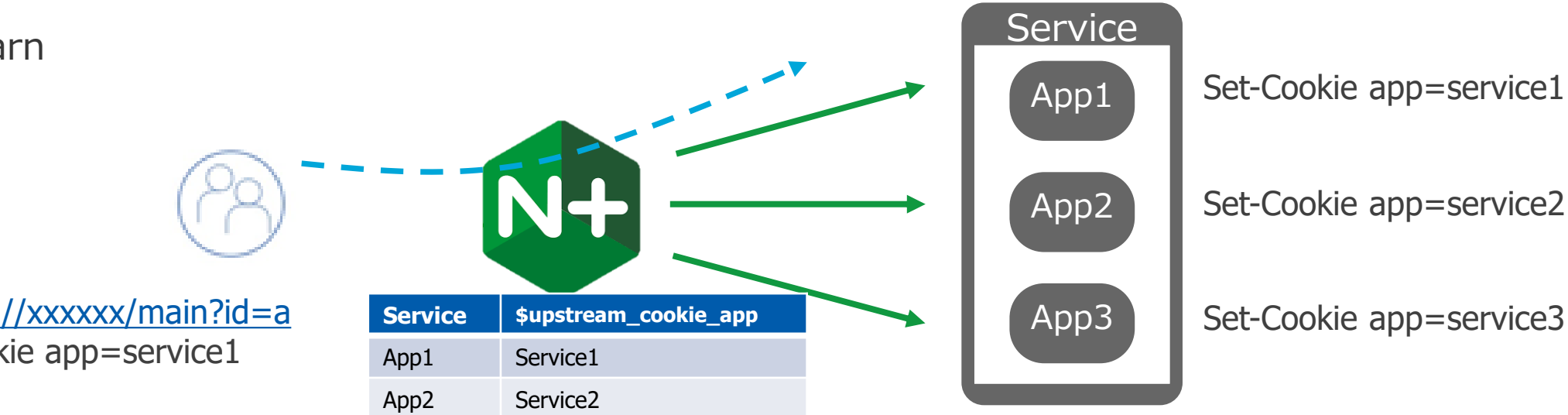
route



http://xxxxxx/main?id=a

順番	Key
1	$Cookie_id
2	$arg_id

Learn



http://xxxxxx/main?id=a
cookie app=service1

Service	$upstream_cookie_app
App1	Service1
App2	Service2
App3	Service3

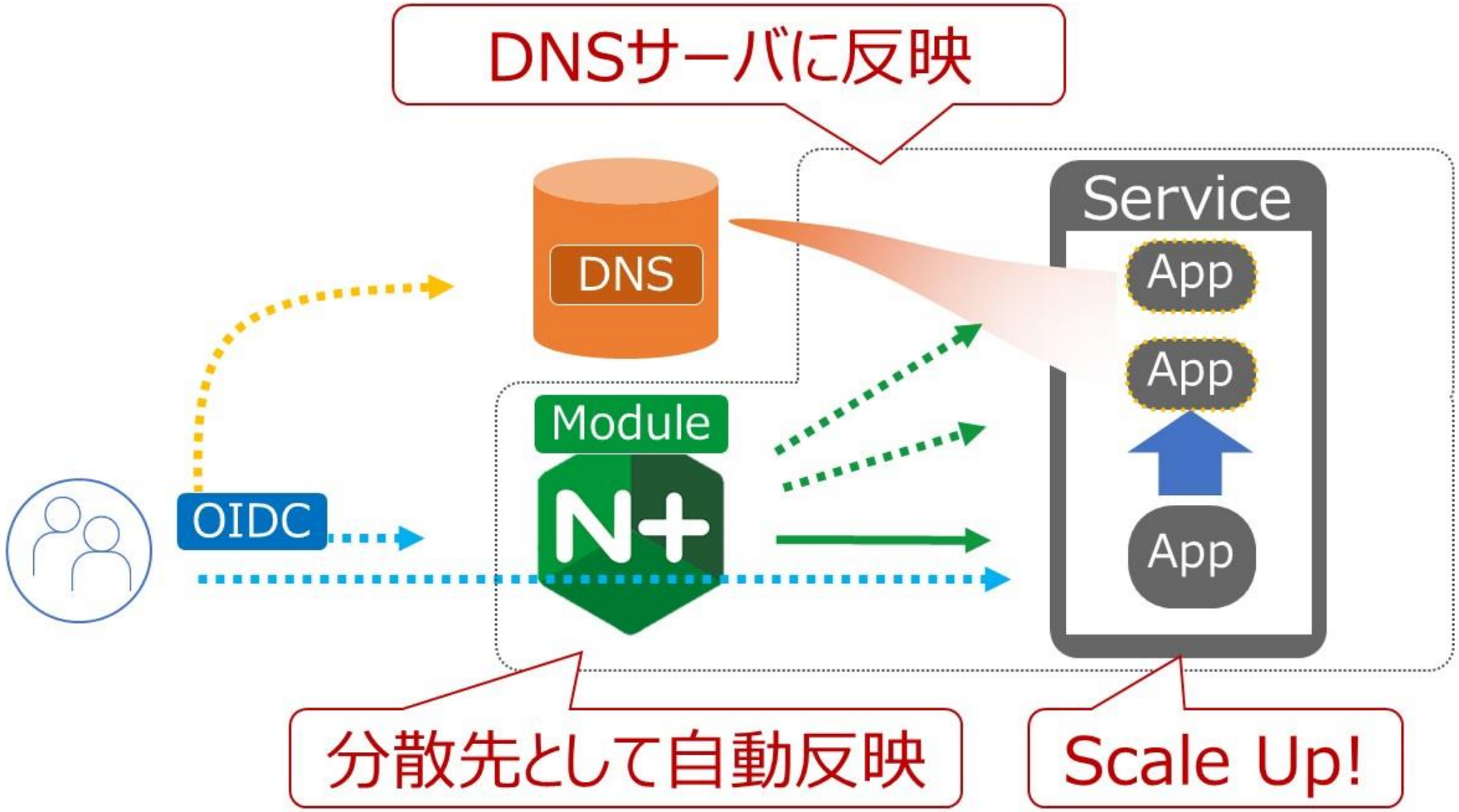
セッションパーシステンス(Learn)

```
upstream server_sticky_learn {
    zone backend 64k;
    sticky learn
        create=$upstream_cookie_srv-id   ・・ upstream レスポンスの srv-id Cookie
        lookup=$cookie_srv-id            ・・ クライアントリクエストの srv-id Cookie
        zone=sticky_learn:1m;

    server backend1;     ・・ レスポンスに cookie srv-id:aaaaaa  が含まれる
    server backend2;     ・・ レスポンスに cookie srv-id:bbbbbb  が含まれる
}

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

サービスディスカバリ



まとめ

まとめ

以下についてご紹介しました。

- Rate Limit
- Connection Limit
- ロードバランシングメソッド
- アクティブヘルスチェック　★
- パーシステンス　★
- サービスディスカバリ　★

参考情報

NGINXの情報発信しています！

NGINXがまるっと分かる！ブログ更新中！

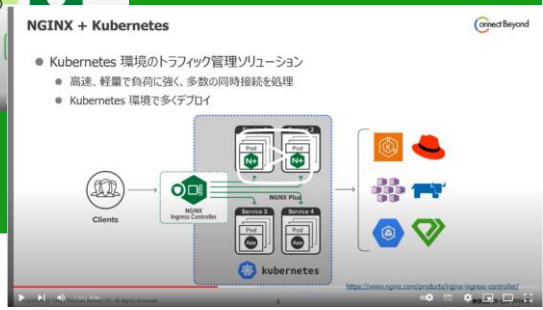
https://cn.teldevice.co.jp/blog/search/?q=NGINX

ブログ

YouTube

NGINXを簡単解説！

F5 NGINX - YouTube



東京エレクトロン デバイス

無償トライアルと個別勉強会のご案内

NGINX Plus　無償トライアルライセンス

https://cn.teldevice.co.jp/product/f5-nginx/

個別勉強会

基礎編、応用編だけでなく、こんなこと知りたいも可能

NGINX の変数 (Tips)

設定ファイルで変数を使用

```
location / {
        proxy_set_header X-Forwarded-For $remote_addr
        proxy_pass http://backend;
 }
```

主な変数

変数名	概要
$remote_addr	クライアント(送信元)の IP アドレス
$request_uri	リクエストの URI 情報 http://xxxxxxx/main の「/main」の部分
$arg_<パラメータ>	リクエストの パラメータ情報 http://xxxxxxx/main?uid=123 の「123」の部分
$http_<ヘッダ名>	HTTP リクエストのヘッダの値
$cookie_<cookie名>	リクエストに含まれる <cookie名> の値