Shared state within the Instance

- Configuration (read-only)

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers
- Upstream health-check status

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers
- Upstream health-check status
- Sticky sessions to upstreams

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers
- Upstream health-check status
- Sticky sessions to upstreams
- API module metrics

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers
- Upstream health-check status
- Sticky sessions to upstreams
- API module metrics
- Key-Value Store

Shared state within the Instance

- Configuration (read-only)
- Runtime state (read-write)

- Connection limits data
- Request limits data
- Proxy cache metadata
- Upstream servers status
- SSL sessions cache
- Stub status counters

- Dynamically configured upstream servers
- Upstream health-check status
- Sticky sessions to upstreams
- API module metrics
- Key-Value Store
- HTTP Session log data

Implementation of state sharing

nginx uses *shared memory* approach:

- requires explicit configuration

Implementation of state sharing

nginx uses *shared memory* approach:

- requires explicit configuration
+ data access is effective

Implementation of state sharing

nginx uses *shared memory* approach:

- requires explicit configuration
+ data access is effective
± data is always consistent, but locked access is usually required

Implementation of state sharing

nginx uses *shared memory* approach:

- requires explicit configuration
+ data access is effective
± data is always consistent, but locked access is usually required
− if a process crashes during an update, data is left in inconsistent state

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:
  - to response, node must interact with others

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:
  - to response, node must interact with others
  - hard to implement and configure properly

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:
  - to response, node must interact with others
  - hard to implement and configure properly
  - network nodes are ephemeral, chances to fail are high

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:
  - to response, node must interact with others
  - hard to implement and configure properly
  - network nodes are ephemeral, chances to fail are high
- the result of such efforts will be a *distributed database*

Sharing Information within Cluster

Can we just extend shared memory approach to the network ?

- consistent access to data is **expensive** and **complex**:
  - to response, node must interact with others
  - hard to implement and configure properly
  - network nodes are ephemeral, chances to fail are high
- the result of such efforts will be a *distributed database*
- . . . nginx is *not* a distributed database (and is not going to be)!

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes
- process new input with new knowledge

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes
- process new input with new knowledge

What useful scenarios we can implement?

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes
- process new input with new knowledge

What useful scenarios we can implement?

- configuration updates

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes
- process new input with new knowledge

What useful scenarios we can implement?

- configuration updates
- session caching

Use Cases for Clustering with Nginx

So what we want from an nginx in a clustered environment?

- process input on each node separately
- inform other nodes about local changes
- process new input with new knowledge

What useful scenarios we can implement?

- configuration updates
- session caching
- apply resource limiting

Offered Solutions in Nginx-Plus

The following modules are available:

- Sticky sessions synchronization (R15)

Offered Solutions in Nginx-Plus

The following modules are available:

- Sticky sessions synchronization (R15)
- Key-Value store synchronization (R16)

Offered Solutions in Nginx-Plus

The following modules are available:

- Sticky sessions synchronization (R15)
- Key-Value store synchronization (R16)
- Distributed request rate limiting (R16)

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh
- Cluster nodes discovery via DNS

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh
- Cluster nodes discovery via DNS
- Very simple push protocol with minimal overhead

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh
- Cluster nodes discovery via DNS
- Very simple push protocol with minimal overhead
- Content-based synchronization (sync objects, not bytes)

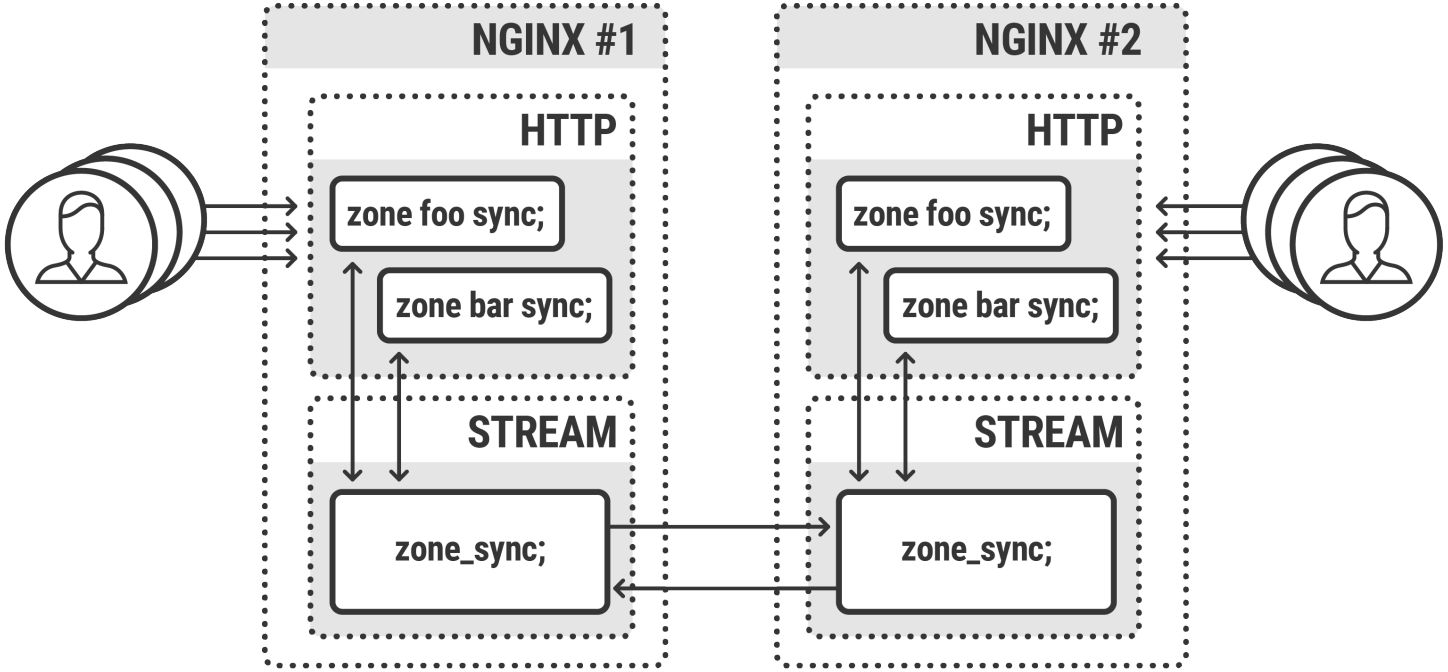
Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh
- Cluster nodes discovery via DNS
- Very simple push protocol with minimal overhead
- Content-based synchronization (sync objects, not bytes)
- Full TLS support

Nginx-Plus R15: the ngx_zone_sync Module

- TCP based stream module
- Native nginx asynchronous I/O
- Full mesh
- Cluster nodes discovery via DNS
- Very simple push protocol with minimal overhead
- Content-based synchronization (sync objects, not bytes)
- Full TLS support
- Configurable latency/resources balance

Syncing Zones Configuration



Syncing Sticky Sessions

```
http {
    upstream backend {
        server ...;
        server ...;

        sticky learn create=$ups_key
                     lookup=$key
                     zone=z:10m     ;
    }
    server {
        ...
        location / {
             proxy_pass http://backend;
        }
    }
}
```

Syncing Sticky Sessions

```
http {
    upstream backend {
        server ...;
        server ...;

        sticky learn create=$ups_key
                     lookup=$key
                     zone=z:10m     ;
    }
    server {
        ...
        location / {
             proxy_pass http://backend;
        }
    }
}
```

```
stream {
    server {
        zone_sync;

    }
}
```

Syncing Sticky Sessions

```
http {
    upstream backend {
        server ...;
        server ...;

        sticky learn create=$ups_key
                     lookup=$key
                     zone=z:10m     ;
    }
    server {
        ...
        location / {
             proxy_pass http://backend;
        }
    }
}
```

```
stream {
    server {
        zone_sync;
        listen 192.168.1.1:12345;

        # cluster nodes list (including self)
        zone_sync_server 192.168.1.1:12345;
        zone_sync_server 192.168.1.2:12345;
    }
}
```

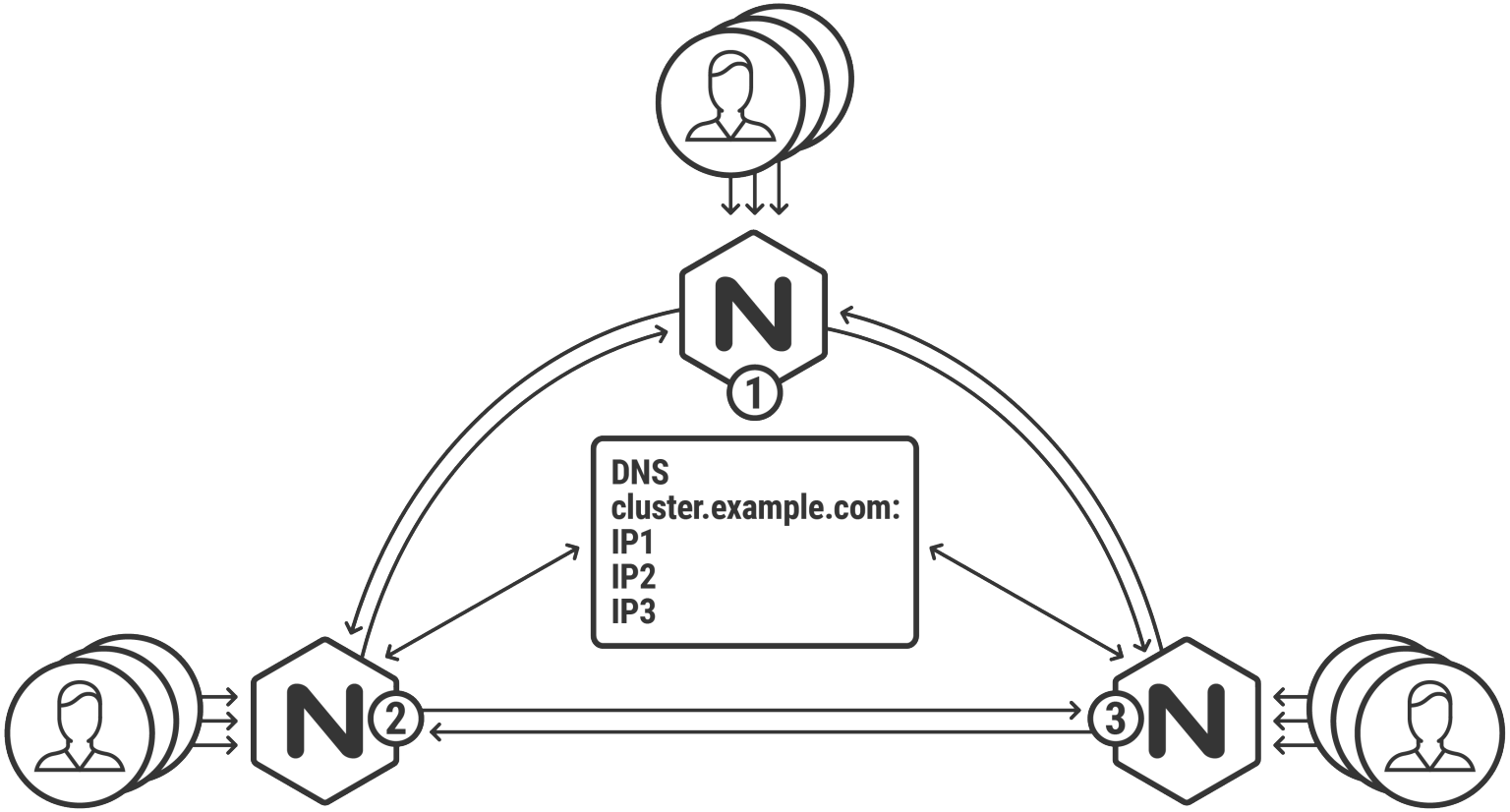
Syncing Sticky Sessions

```
http {
    upstream backend {
        server ...;
        server ...;
        # mark zone 'z' for synchronization
        sticky learn create=$ups_key
                     lookup=$key
                     zone=z:10m sync;
    }
    server {
        ...
        location / {
             proxy_pass http://backend;
        }
    }
}
```

```
stream {
    server {
        zone_sync;
        listen 192.168.1.1:12345;

        # cluster nodes list (including self)
        zone_sync_server 192.168.1.1:12345;
        zone_sync_server 192.168.1.2:12345;
    }
}
```

Using DNS to Obtain Cluster Nodes



Using DNS to Obtain Cluster Nodes

```
stream {

    server {
        zone_sync;
        listen 192.168.1.1:12345;

        # cluster nodes list (including self)
        zone_sync_server 192.168.1.1:12345;
        zone_sync_server 192.168.1.2:12345;

    }
}

http { ... }
```

Using DNS to Obtain Cluster Nodes

```
stream {
    resolver 127.0.0.1;
    server {
        zone_sync;
        listen 192.168.1.1:12345;




        zone_sync_server cluster.example.com:12345 resolve;
    }
}

http { ... }
```

Secure Connections Between Nodes with SSL

```
zone_sync;
listen 192.168.1.1:12345 ssl;

ssl_certificate foo.crt;
ssl_certificate_key foo.key;

zone_sync_ssl on;
zone_sync_ssl_certificate node.crt;
zone_sync_ssl_certificate_key node.key;
zone_sync_ssl_verify on;
zone_sync_ssl_trusted_certificate trusted.crt;
```

Tuning Synchronization: latency vs bandwidth/CPU

```
zone_sync;
listen ...;
zone_sync_server ...;

zone_sync_interval 500ms;

zone_sync_buffers 256 8k;
zone_sync_recv_buffer_size 8k;
zone_sync_timeout 5s;
```

More modules

- Key-Value storage (http and stream):

```
keyval_zone zone=one:32k state=one.keyval sync;
```

- HTTP Request Limiting:

```
limit_req_zone $binary_remote_addr
               zone=one:10m
               rate=1r/s sync;
```

Thank you!

Links:

- https://nginx.org/en/docs/stream/ngx_stream_zone_sync_module.html
- https://www.nginx.com/blog/nginx-plus-r15-released/#state-sharing
- https://www.nginx.com/blog/nginx-plus-r16-released/#r16-cluster-rate-limiting
- https://www.nginx.com/blog/nginx-plus-r16-released/#r16-cluster-key-value

Questions?

- Owen Garret / owen@nginx.com
- Vladimir Khomutov / vl@nginx.com