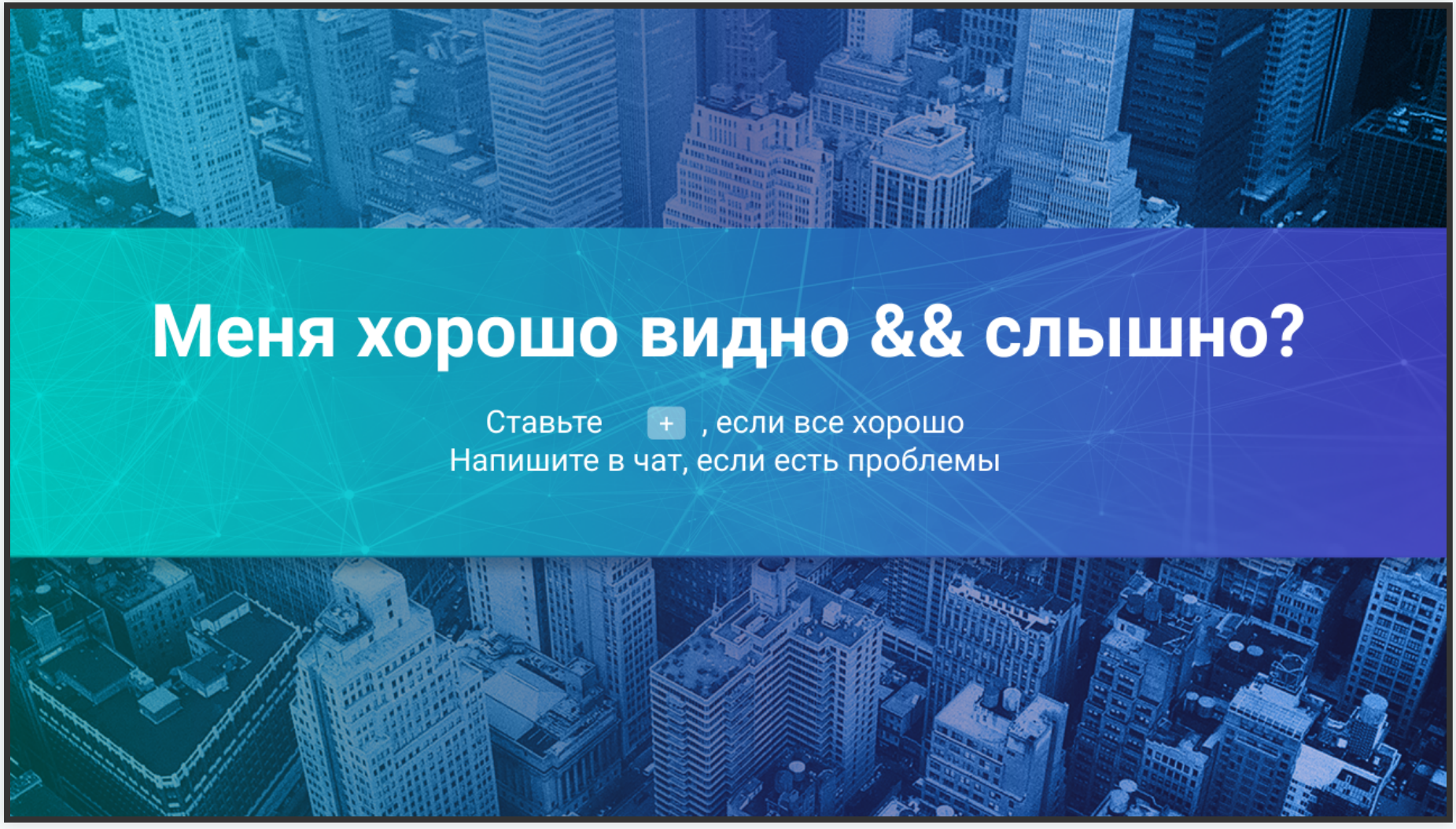

OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование


Меня хорошо видно && слышно?
Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

НЕ ЗАБЫТЬ ВКЛЮЧИТЬ ЗАПИСЬ!!!

(сегодня это тебя касается!)

Nginx. Балансировка.

План занятия

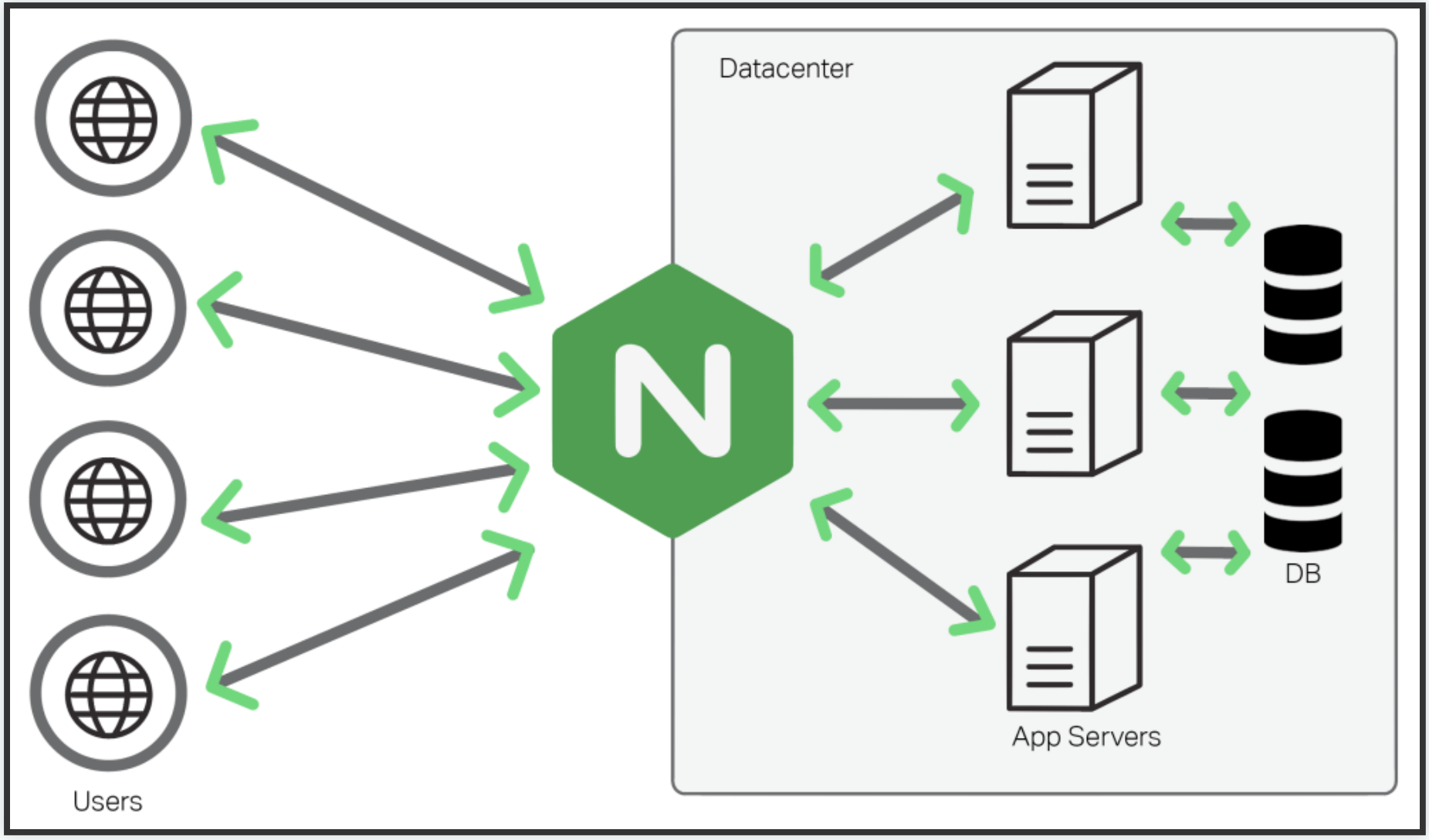
БАЛАНСИРОВКА В ОБЩИХ ЧЕРТАХ

БАЛАНСИРОВКА HTTP

- основные методы балансировки
- параметры балансировки

БАЛАНСИРОВКА TCP И UDP

Балансировка в общем виде



Конфигурация балансировщика в nginx

```
# ngx_http_upstream_module
upstream backend {
    server backend1.example.com;
    server 127.0.0.1:8080;
    server unix:/tmp/backend2;
}
server {
    server_name example.com;
    listen 80;
    location / {
        # ngx_http_proxy_module
        proxy_pass http://backend;
    }
}
```

Методы балансировки

NGINX Open Source

- Round Robin
- Least Connection
- IP Hash
- Hash
- Random

NGINX Plus

- Least Time
- Random with least_time

Round Robin

```
upstream backend {
    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com;
}
server {
    server_name example.com;
    listen 80;
    location / {
        proxy_pass http://backend;
    }
}
```

Least Connection

```
upstream backend {
    least_conn;
    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com;
}
server {
    server_name example.com;
    listen 80;
    location / {
        proxy_pass http://backend;
    }
}
```

- У запросов как правило разное время обработки, а значит серверы загружены по-разному
- Запрос передаётся серверу с наименьшим числом активных соединений, с учётом весов серверов.
- Если подходит сразу несколько серверов, они выбираются в режиме round-robin с учётом их весов.

IP Hash

```
upstream backend {
    ip_hash;
    server web1.example.com;
    server web2.example.com;
    server web3.example.com;
}
server {
    server_name example.com;
    listen 80;
    location / {
        proxy_pass http://backend;
    }
}
```

Hash

```
upstream backend {
    hash $scheme$request_uri;
    server web1.example.com;
    server web2.example.com;
    server web3.example.com;
}
server {
    server_name example.com;
    listen 80;
    location / {
        proxy_pass http://backend;
    }
}
```

Random

```
upstream backend {
    random; #two least_conn; # наименьшее число активных соединен
    server web1.example.com;
    server web2.example.com;
    server web3.example.com;
}
server {
    server_name example.com;
    listen 80;
    location / {
        proxy_pass http://backend;
    }
}
```

Вопросы

Параметры балансировки

- weight
- max_conns
- backup
- down
- max_fails
- fail_timeout

Weight

```
upstream backend {
    server web1.example.com    weight=3;
    server web2.example.com; # weight=1 (default)
    server web3.example.com;
}
```

demo: https://ticket2ride.systemctl.su

```
siege -r150 -c25 https://ticket2ride.systemctl.su
```

max_fails и fail_timeout;

```
upstream backend {
    server web1.example.com  max_fails=2 fail_timeout=3;
    server web2.example.com; # max_fails=1 fail_timeout=10
    server web3.example.com;
}
```

max_conns

```
upstream backend {
    server web1.example.com max_conns=1000;
    server web2.example.com max_conns=500;
    server web3.example.com; #max_conns=0 (default - unlimited)
}
```

Параметры backup и down

```
upstream backend {
    server web1.example.com;
    server web2.example.com backup;
    server web3.example.com down;

}
```

Вопросы

Балансировка TCP и UDP

Да, Nginx и такое умеет!

Балансировка TCP и UDP

Простой конфиг для проксирования TCP трафика:

```
stream {
    server {
        listen 9088;
        proxy_pass 127.0.0.1:7001;
    }
}

http {

}
```

Балансировка TCP и UDP

Директива `listen`

```
listen 127.0.0.1:12345;
listen *:12345;
listen 12345;
listen localhost:12345;
listen [::1]:12345;
listen [::]:12345;
listen unix:/var/run/nginx.sock;
listen 127.0.0.1:12345-12399;
listen 12345-12399;
```

Балансировка TCP и UDP

Директива `proxy_pass`

```
proxy_pass localhost:12345;
proxy_pass unix:/tmp/stream.socket;
proxy_pass $upstream;
proxy_pass upstream_name;
```

Балансировка TCP и UDP

Давайте попробуем!

Балансировка TCP и UDP

Для балансировки добавляем блок upstream:
(ngx_stream_upstream_module)

```
stream {
    upstream stream_backend {
        server 127.0.0.1:7001;
        server 127.0.0.1:7002;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Балансировка TCP и UDP

Пример для UDP:

```
stream {
    upstream dns_servers {
        server 192.168.136.130:53;
        server 192.168.136.131:53;
    }
    server {
        listen 53 udp;
        proxy_pass dns_servers;
    }
}
```

Балансировка TCP и UDP

- Так же есть поддержка SSL/TLS модулем **ngx_stream_ssl_module**.
- Возможность ограничивать число соединений - модуль **ngx_stream_limit_conn_module**.
- Для ограничения доступа можно использовать директивы **allow** и **deny** из модуля **ngx_stream_access_module**.
- Модуль **ngx_stream_log_module** добавляет логирование.
- **ngx_stream_geo_module** - создаёт переменные, значения которых зависят от IP-адреса клиента.
- **ngx_stream_...**

Но мы здесь об этом разговаривать не будем.

Балансировка TCP и UDP

backup - запасной сервер. Будет обрабатывать подключения, если не работают основные (не работает совместно с hash и random).

```
stream {
    upstream stream_backend {
        server 127.0.0.1:7001;
        server 127.0.0.1:7002;
        server 127.0.0.1:7003 backup;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Балансировка TCP и UDP

weight - вес сервера.

```
stream {
    upstream stream_backend {
        server 127.0.0.1:7001;
        server 127.0.0.1:7002 weight=2;
        server 127.0.0.1:7003 backup;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Балансировка TCP и UDP

max_conns - ограничивает максимальное число одновременных подключений.

```
stream {
    upstream stream_backend {
        server 127.0.0.1:7001 max_conns=3;
        server 127.0.0.1:7002 weight=2;
        server 127.0.0.1:7003 backup;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Попробуем?

Балансировка TCP и UDP

- **max_fails** - задаёт число неудачных попыток работы с сервером за **fail_timeout**, после которого сервер считается недоступным на время **fail_timeout** [default 1].
- **fail_timeout** - [default 10 секунд]

```
stream {
  upstream stream_backend {
      server 127.0.0.1:7001 max_conns=3;
      server 127.0.0.1:7002 weight=2 max_fails=2 fail_timeout=15;
      server 127.0.0.1:7003 backup;
  }
  server {
      listen 9088;
      proxy_pass stream_backend;
  }
}
```

Балансировка TCP и UDP

Скажите, что будет если добавить **down** второму серверу и послать Nginx-у сигнал reload?

```
stream {
    upstream stream_backend {
        server 127.0.0.1:7001 max_conns=3;
        server 127.0.0.1:7002 weight=2 down;
        server 127.0.0.1:7003 backup;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Балансировка TCP и UDP

Давайте выведем сервер на обслуживание!

Балансировка TCP и UDP

Балансировка:

- **Round Robin** - соединения устанавливаются с первого и далее по списку с учётом весов.
- **least_conn** - выбирается сервер с наименьшим числом активных соединений с учётом весов серверов.
- **random** - сервер выбирается случайным образом.
- **hash** - сервер выбирается на основании ключа.

Балансировка TCP и UDP

least_conn:

```
stream {
    upstream stream_backend {
        least_conn;
        server 127.0.0.1:7001;
        server 127.0.0.1:7002;
        server 127.0.0.1:7003 backup;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

Балансировка TCP и UDP

random:

```
stream {
    upstream stream_backend {
        random two;
        server 127.0.0.1:7001;
        server 127.0.0.1:7002;
        server 127.0.0.1:7003;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

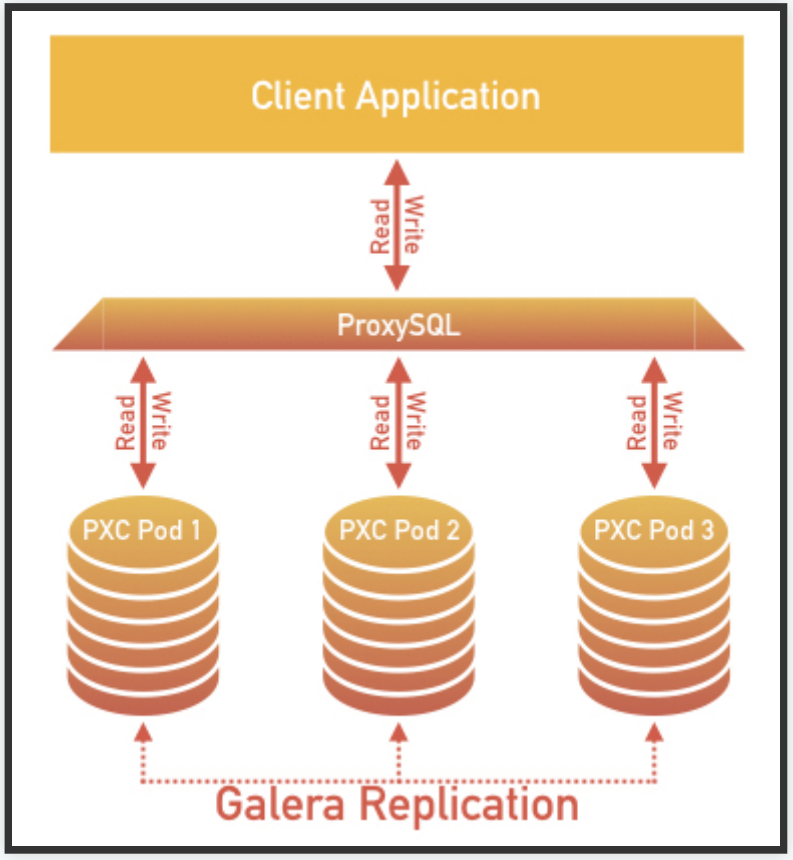
Балансировка TCP и UDP

hash:

```
stream {
    upstream stream_backend {
        hash $remote_addr consistent;
        server 127.0.0.1:7001;
        server 127.0.0.1:7002;
        server 127.0.0.1:7003;
    }
    server {
        listen 9088;
        proxy_pass stream_backend;
    }
}
```

- **consistent** - постоянство выбора бэкенда при отключении или добавлении новых серверов. Использует метод хэширования Ketama.

Балансировка TCP и UDP



Балансировка TCP и UDP

Nginx proxy for Percona Cluster:

```
upstream stream_sql {
   server 10.51.21.114:3307;
   server 10.51.21.114:3308;
   server 10.51.21.114:3309;
}

server {
   listen        3306;
   access_log  /var/log/nginx/stream.log stream;

   proxy_pass stream_sql;
}
```

Динамические upstream

```
resolver 127.0.0.1:8600 valid=2s;
upstream name {
    server service.consul service=docs resolve max_fails=50 fail_
}
```

Видео streaming

На примере rtmp модуля для Nginx и CentOS 7.

Видео streaming

Добавляем репозиторий и устанавливаем Nginx:

```
cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/\$releasever/\$basearch/
gpgcheck=0
enabled=1
EOF

yum install -y nginx pcre-devel openssl-devel
```

Видео streaming

Скачиваем исходники Nginx-а и rtmp модуля и устанавливаем:

```
export NGINX_VERSION=$(nginx -v 2>&1 | awk -F"/" '{print $2}')

curl -O http://nginx.org/download/nginx-$NGINX_VERSION.tar.gz

git clone https://github.com/arut/nginx-rtmp-module.git

cd nginx-$NGINX_VERSION

nginx -V 2>&1 | grep -oP 'configure arguments: \K.+' \
  | xargs ./configure --add-module=../nginx-rtmp-module

make
make install
```

Видео streaming

Добавляем на самый верхний уровень в /etc/nginx/nginx.conf:

```
rtmp {
    server {
        listen 1935;
        application live {
            live on;
        }
    }
}
```

```
nginx -t
nginx -s reload
```

Видео streaming

Ставим ffmpeg:

```
yum install -y epel-release
rpm -Uvh http://li.nux.ro/download/nux/dextop/el7/x86_64/nux-dext

yum install -y ffmpeg rtmpdump
```

И запускаем трансляцию:

```
ffmpeg -re -i mi.mp4 -vcodec copy -loop -1 -c:a aac \
  -b:a 160k -ar 44100 -strict -2 -f flv rtmp:127.0.0.1/live/bbb
```

Видео streaming

`rtmp://207.180.210.166/live/mi`

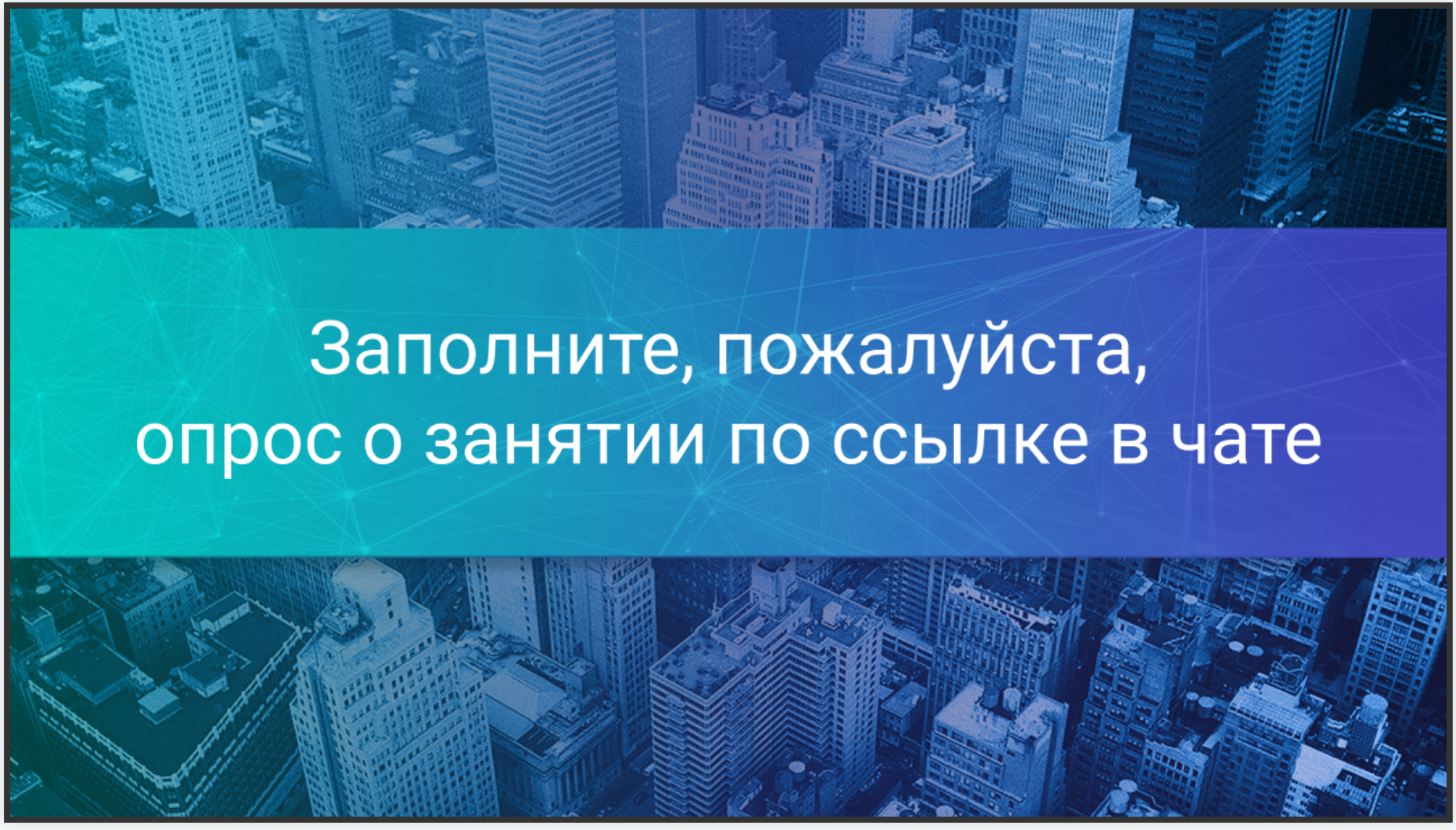
Для mplayer-a:

```
mplayer rtmp://207.180.210.166/live/mi
```

Рефлексия

Отметьте 3 пункта, которые вам запомнились с вебинара

Что вы будете применять в работе из сегодняшнего вебинара?


Заполните, пожалуйста,
опрос о занятии по ссылке в чате