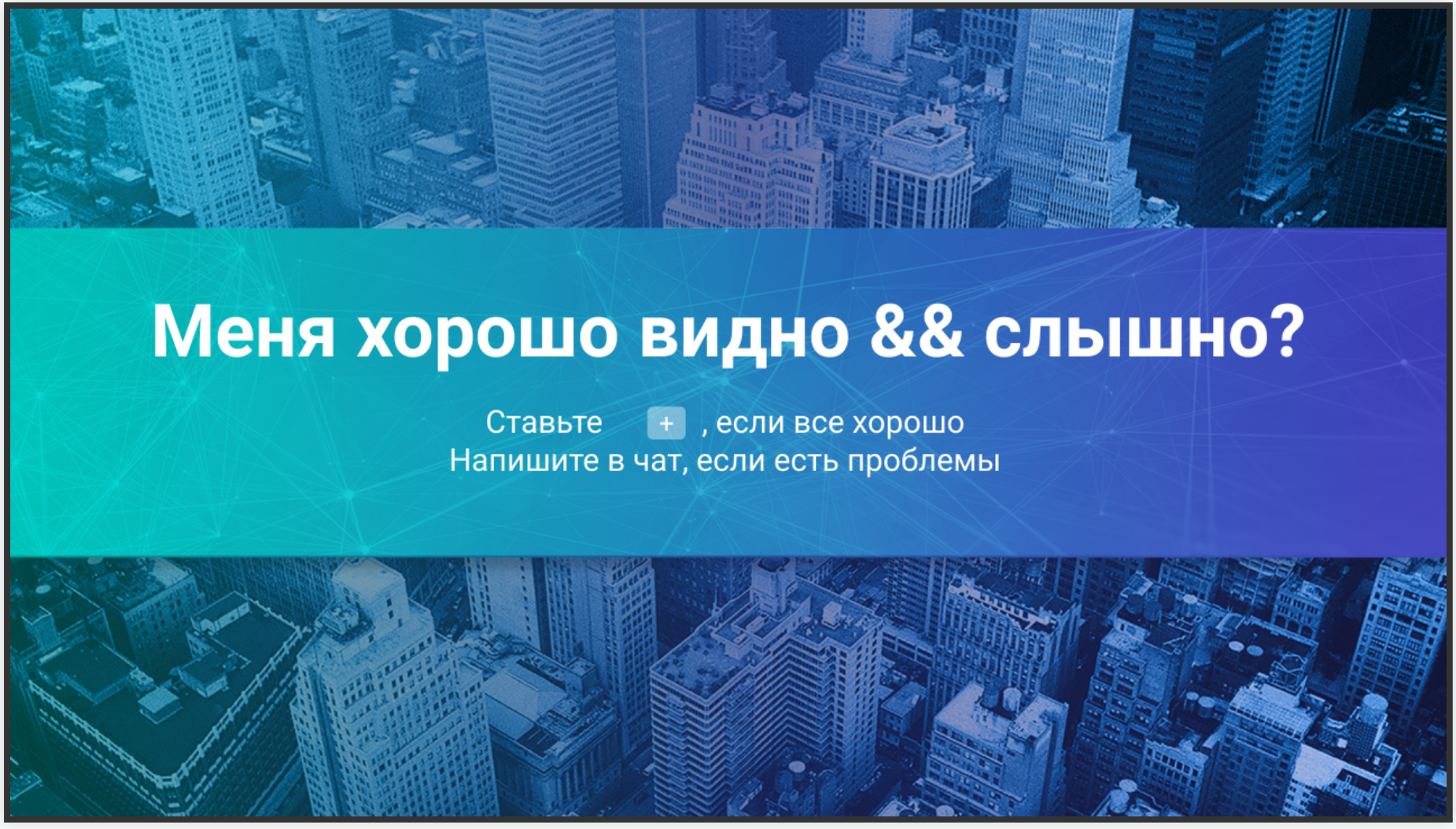

OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование


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

НЕ ЗАБЫТЬ ВКЛЮЧИТЬ ЗАПИСЬ!!!

Установка из репозитория

CentOS 7

```
# install
sudo yum install -y epel-release
sudo yum install -y nginx
# allow firewall
firewall-cmd --permanent --zone=public --add-port=8001/tcp
firewall-cmd --reload
# check selinux  ## policycoreutils-python
sudo semanage port -l | grep http_port
# add port to selinux context
semanage port -a -t http_port_t -p tcp 8001
```

Nginx. Кейсы конфигурации

Ссылка на WebDebugger от Алексея Колоскова

Здравствуйте

- Кто читал официальную документацию?

План занятия:

- **Установка из репозитория. Структура директорий**
- Базовый конфиг, include
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- location
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

Структура директорий

```
/etc/nginx/
├── conf.d
│   └── webdebugger.conf
├── default.d
├── fastcgi.conf
├── fastcgi.conf.default
├── fastcgi_params
├── fastcgi_params.default
├── koi-utf
├── koi-win
├── mime.types
├── mime.types.default
├── nginx.conf
├── nginx.conf.default
├── scgi_params
├── scgi_params.default
```

План занятия:

- Установка из репозитория. Структура директорий
- **Базовый конфиг, include**
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- location
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

Базовый конфиг

/etc/nginx/nginx.conf

```
user nginx;  # user [group]; Пользователь от имени которого будут
worker_processes auto;  # Число рабочих процессов. Зависит от чис
error_log /var/log/nginx/error.log;  # Конфигурирует запись в лог
pid /run/nginx.pid;  # номер (PID) главного процесса

include /usr/share/nginx/modules/*.conf;  # include файл | маска;

events {  # директивы, влияющие на обработку соединений
    worker_connections 1024;  # максимальное число соединений, на
}

http {  # http (https, http2, etc...) сервер
    log_format  main  '$remote_addr - $remote_user [$time_local]
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"
```

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- **Директивы и Контексты, что это?**
- Порядок обработки запроса (директива `server`)
- location
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

Директивы и контексты

Директивы простые

```
listen 80 default_server;
listen 443 ssl default_server;
root /var/www;
```

Директивы блочные (контекст)

```
events {}
http {
    server {
        location {
            ...
        }
    }
}
```

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- Директивы и Контексты, что это?
- **Порядок обработки запроса (директива `server`)**
- location
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

ПОРЯДОК ОБРАБОТКИ ЗАПРОСА

1. **`listen`** хост:порт
2. **`server_name`** значение http-заголовка **`Host`**
3. если **`server_name`** не найдено - сервер по умолчанию

```
http {
   server {
     listen 80 default_server;
     server_name _;
   }
}
```

server

LISTEN

По умолчанию: `listen *:80 | *:8000;`

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;

listen [::]:8000;
listen [::1];

listen unix:/var/run/nginx.sock;
```

demo listen порт по умолчанию

```
sudo -s
mkdir /tmp/nginx
cat <<EOF > /tmp/nginx/nginx-listen.conf
pid /tmp/nginx.pid;
events {}
http {
  server {
    return 200 "Server W/O 'listen'!\n";
  }
}
EOF
nginx -tc /tmp/nginx/nginx-listen.conf
nginx -c /tmp/nginx/nginx-listen.conf
ps aux | grep [n]ginx
ss -lptn
curl 127.0.0.1:80
```

demo listen порт по умолчанию

```
nginx -c /tmp/nginx/nginx-listen.conf -s stop
exit
sudo -u nginx -s
nginx -tc /tmp/nginx/nginx-listen.conf
nginx -c /tmp/nginx/nginx-listen.conf
ps aux | grep [n]ginx
ss -lptn
curl 127.0.0.1:8000
```

server

Тест без имени сервера

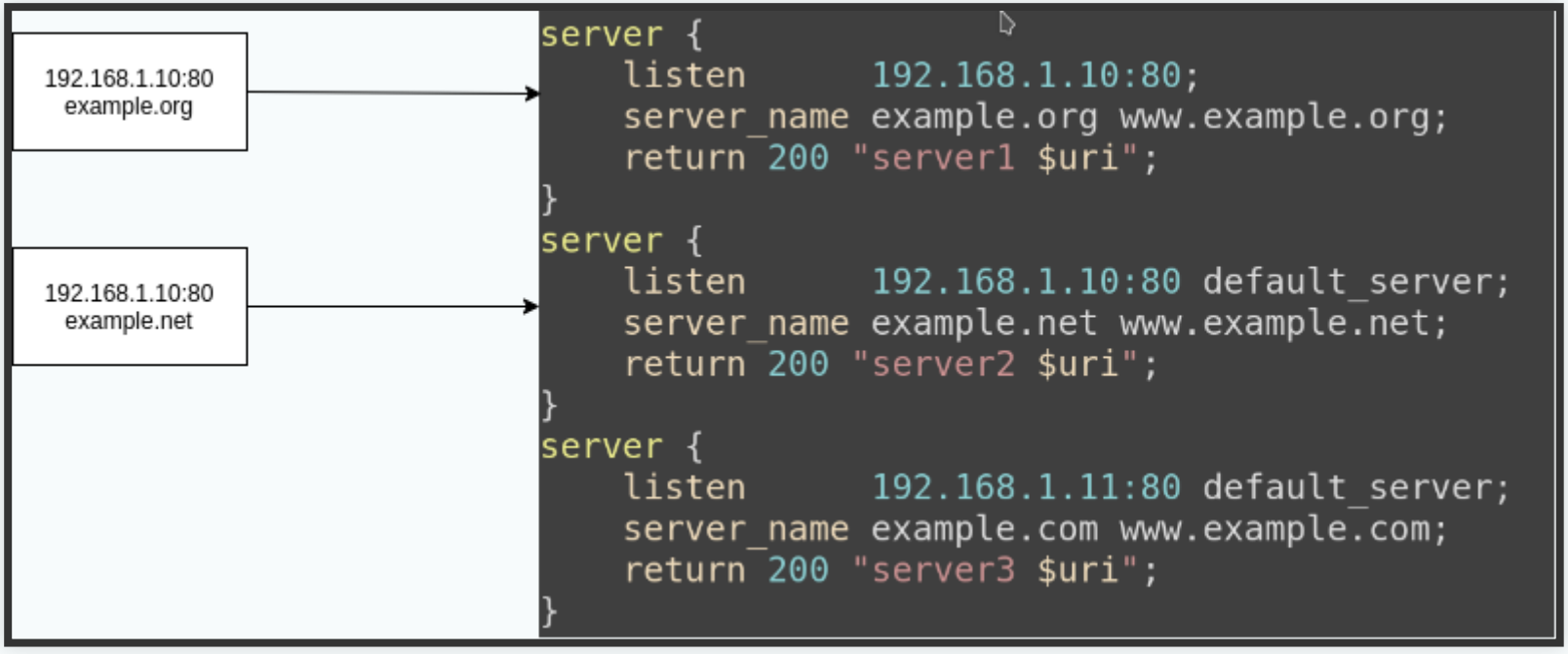
```
#!/bin/bash
set -e
IP=${1:-127.0.0.1}
PORT=${2:-80}
HOSTNAME=$3
exec 3<>/dev/tcp/$IP/$PORT
#echo -e "GET / HTTP/1.1\r\nConnection: close\r\n\r\n" >&3
echo -e "GET / HTTP/1.1\r\nhost: $HOSTNAME\r\nConnection: close\r
cat <&3
```

server

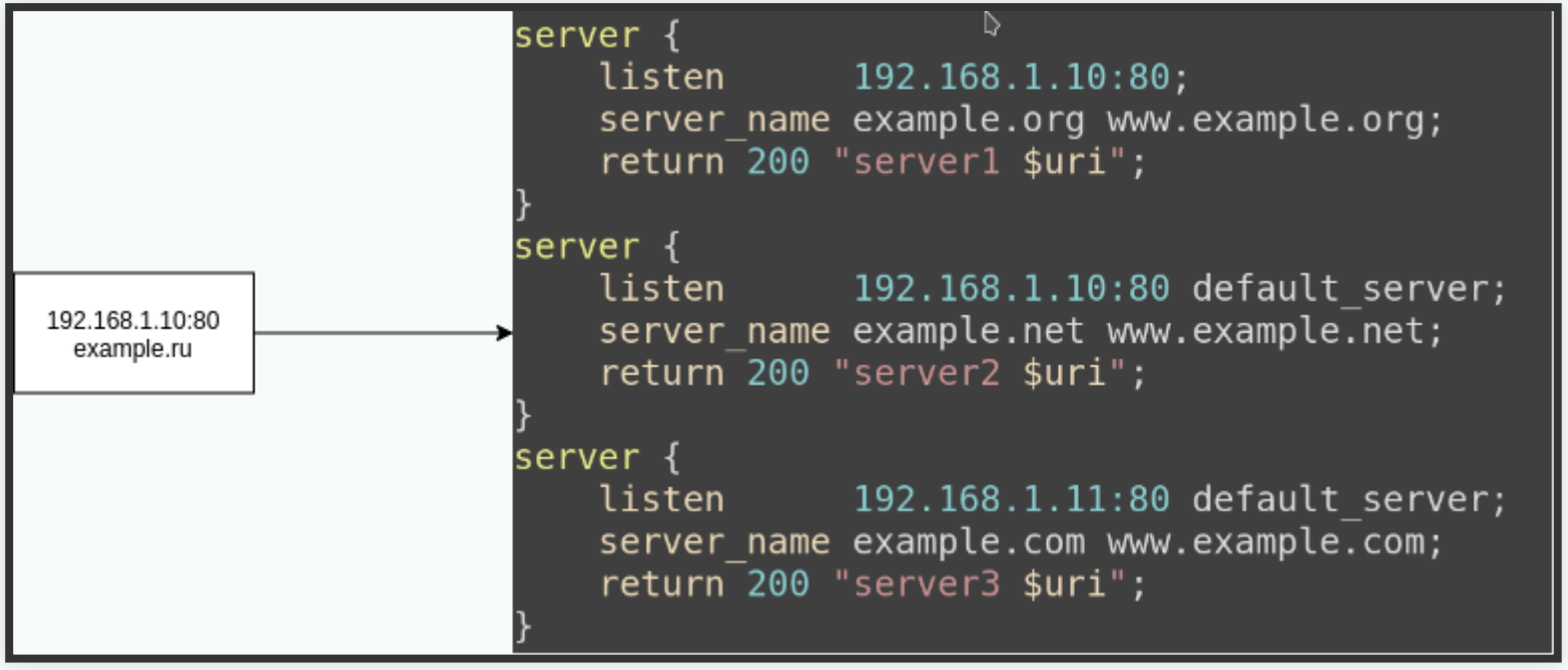
У каждого **listen** свой **default_server**

```
server {
    listen       192.168.1.10:80;
    server_name example.org www.example.org;
    return 200 "server1\n";
}
server {
    listen       192.168.1.10:80 default_server;
    server_name example.net www.example.net;
    return 200 "server2\n";
}
server {
    listen       192.168.1.11:80;
    server_name example.com www.example.com;
    return 200 "server3\n";
}
server {
```

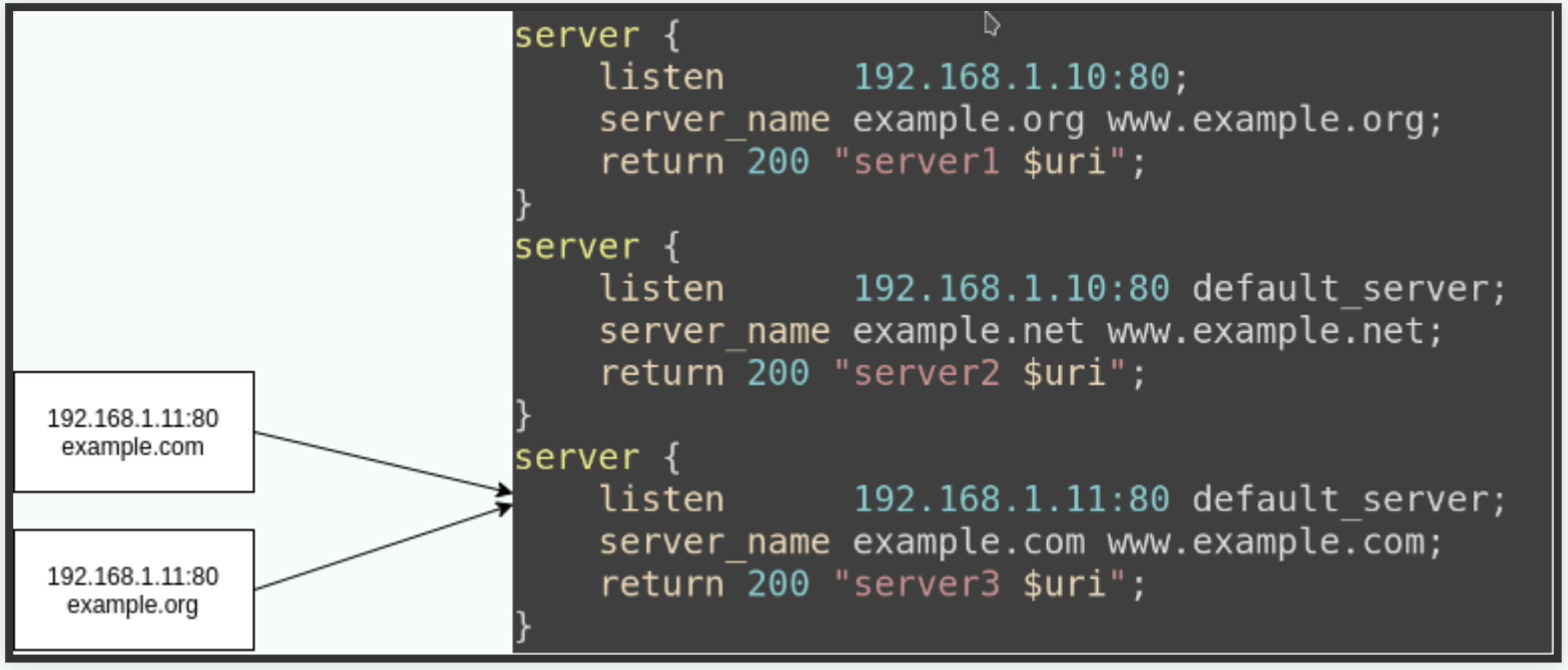
server


192.168.1.10:80
example.org
192.168.1.10:80
example.net
server {
listen 192.168.1.10:80;
server_name example.org www.example.org;
return 200 "server1 $uri";
}
server {
listen 192.168.1.10:80 default_server;
server_name example.net www.example.net;
return 200 "server2 $uri";
}
server {
listen 192.168.1.11:80 default_server;
server_name example.com www.example.com;
return 200 "server3 $uri";
}

server


192.168.1.10:80
example.ru
server {
listen 192.168.1.10:80;
server_name example.org www.example.org;
return 200 "server1 $uri";
}
server {
listen 192.168.1.10:80 default_server;
server_name example.net www.example.net;
return 200 "server2 $uri";
}
server {
listen 192.168.1.11:80 default_server;
server_name example.com www.example.com;
return 200 "server3 $uri";
}

server




```
server {
    listen      192.168.1.10:80;
    server_name example.org www.example.org;
    return 200 "server1 $uri";
}
server {
    listen      192.168.1.10:80 default_server;
    server_name example.net www.example.net;
    return 200 "server2 $uri";
}
server {
    listen      192.168.1.11:80 default_server;
    server_name example.com www.example.com;
    return 200 "server3 $uri";
}
```

server_name

ПОРЯДОК

Проверяет только поле "**`Host`**"

1. точное имя, "**www.example.com**"
2. самое длинное * в начале, "***.example.org**"
3. самое длинное * в конце, "**api.***"
4. первое подходящее **регулярное выражение** (в **порядке следования** в конфигурационном файле)
5. **default_server**

server

SERVER_NAME

```
server {
    server_name example.com www.example.com;
}
server {
    server_name example.com *.example.com www.example.*;
    # otus.example.com, www.example.net
}
server {
    server_name .example.com;
    # example.com, otus.example.com, www.example.com
}
server {
    server_name www.example.com ~^www\d+\.example\.com$;
    # www1.example.com, www44.example.com
}
```

server

SERVER_NAME

```
/sites/
├── site1.com
│   └── index.html
└── site2.com
    └── index.html

2 directories, 2 files
```

Как обработать одной директивой server?

server

SERVER_NAME

```
/sites/
├── site1.com
│   └── index.html
└── site2.com
    └── index.html

2 directories, 2 files
```

```
server {
    server_name ~^(www\.)?(.+)$;  # www.site1.com site2.com
    root /sites/$2;  # /sites/site1.com или /sites/site2.com
}
```

server

SERVER_NAME

```
/sites/
├── site1.com
│   └── index.html
└── site2.com
    └── index.html

2 directories, 2 files
```

```
server {
  server_name ~^(www\.)?(?<domain>.+)$;  # www.site1.com site2.co

  root /sites/$domain;  # /sites/site1.com или /sites/site2.com
}
```

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- **location**
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

location

Устанавливает конфигурацию в зависимости от URI запроса.

```
http://www.site1.com/index.php

http://site1.com:8000/index.php?page=1&user=john

http://www.site2.com/index.php?page=1&something+else&user=john
```

location одинаковый, **`/index.php`**

location

Что такое location

- в нормализованном виде
- после декодирования текста, заданного в виде "`%XX`"
- преобразования относительных элементов пути "`.`" и "`..`" в реальные
- возможной замены двух и более подряд идущих слэшей на один (в зависимости от значения директивы merge_slashes).

location

Типы и приоритеты

1. **=** - точное совпадение (**префиксный** location)

```
location = / { ... }
```

2. **^~** - преимущество (**префиксный** location)

```
location ^~ /images { ... }
```

3. **~** или **~*** - регулярное выражение

```
location ~* ^/.* { ... }
```

4. **префиксный** location без модификатора

```
location / { ... }
```

location

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

- Модификатор "~*" без учёта регистра
- Модификатор "~" с учётом регистра

location

```
location = / {  # запрос `/`
    return 200 "1: location = / {";
}

location / {  # запрос `/index.html`
    return 200 "2: location / {";
}

location /documents/ {  # запрос `/documents/document.html`
    return 200 "3: location /documents/ {";
}

location ^~ /images/ {  # запрос `/images/1.gif`
    return 200 "4: location ^~ /images/ {";
}
```

location

если

- location задан **префиксной строкой** со **слэшом в конце**

```
location /front/
```

- и запросы обрабатываются при помощи `proxy_pass`, `fcgi_pass`, ...

то

- Запрос **без завершающего слэша** http://example.com/front
- Ответ **постоянное перенаправление** с кодом "`301`" на URI с **добавленным в конец слэшом**

location

Если такое поведение нежелательно, можно задать точное совпадение URI и location, например:

```
location /user/ {
    proxy_pass http://user.example.com;
}
location = /user {
    proxy_pass http://login.example.com;
}
```

location

- location: `/front` без завершающего слеша
- proxy_pass: `http://127.0.0.1:8000` без URI

query uri	result uri
`http://127.0.0.1/front`	`/front`
`http://127.0.0.1/front/`	`/front/`
`http://127.0.0.1/frontotus`	`/frontotus`
`http://127.0.0.1/frontotus/`	`/frontotus/`
`http://127.0.0.1/front/otus`	`/front/otus`
`http://127.0.0.1/front/otus/`	`/front/otus/`
`http://127.0.0.1/front/otus/index.html`	`/front/otus/index.html`

location

- location: `/front` без завершающего слеша
- proxy_pass: `http://127.0.0.1:8000/` добавили URI `/`

query uri	result uri
`http://127.0.0.1/front`	`/`
`http://127.0.0.1/front/`	`//`
`http://127.0.0.1/frontotus`	`/otus`
`http://127.0.0.1/frontotus/`	`/otus/`
`http://127.0.0.1/front/otus`	`//otus`
`http://127.0.0.1/front/otus/`	`//otus/`
`http://127.0.0.1/front/otus/index.html`	`//otus/index.html`

location

- location: `/front/` с завершающим слешем
- proxy_pass: `http://127.0.0.1:8000/webapp` без завершающего слеша

query uri	result uri
`http://127.0.0.1/front`	301 `/front/`
`http://127.0.0.1/front/`	`/webapp`
`http://127.0.0.1/frontotus`	404
`http://127.0.0.1/frontotus/`	404
`http://127.0.0.1/front/otus`	`/webappotus`
`http://127.0.0.1/front/otus/`	`/webappotus/`
`http://127.0.0.1/front/otus/index.html`	`/webappotus/index.html`

location

- location: `/front/` с завершающим слешем
- proxy_pass: `http://127.0.0.1:8000/webapp/` с завершающим слешем

query uri	result uri
`http://127.0.0.1/front`	301 `/front/`
`http://127.0.0.1/front/`	`/webapp/`
`http://127.0.0.1/frontotus`	404
`http://127.0.0.1/frontotus/`	404
`http://127.0.0.1/front/otus`	`/webapp/otus`
`http://127.0.0.1/front/otus/`	`/webapp/otus/`
`http://127.0.0.1/front/otus/index.html`	`/webapp/otus/index.html`

ИМЕНОВАНЫЙ LOCATION

Префикс "@" задаёт именованный location

- не используется при обычной обработке запросов
- предназначен только для перенаправления в него запросов
- не могут быть вложенными и не могут содержать вложенные location’ы

Пример будет далее

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- location
- **proxy_pass, fcgi, uwsgi**
- return, rewrite, try_files
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

proxy_pass, fcgi, uwsgi

- **proxy_pass** - http proxy
- **fastcgi_pass** - fastcgi proxy (php)
- **uwsgi_pass** - uwsgi proxy (python)
- **scgi_pass** - scgi proxy (???)
- **memcached_pass** - memcached proxy (memcached)
- **grpc_pass** - grpc proxy (grpc)

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- location
- proxy_pass, fcgi, uwsgi
- **return, rewrite, try_files**
- error_page
- Сжатие трафика
- Кэширование трафика
- Отдача статики

return

Синтаксис

```
return (301 | 302 | 303 | 307) url;
```

```
return (1xx | 2xx | 4xx | 5xx) ["text"];
```

return

- Перенаправить на новый домен

```
server_name www.old.com old.com;
return 301 $scheme://www.new.com$request_uri;
```

- Добавить/удалить префикс `www.`

```
server_name example.com;
return 301 $scheme://www.example.com$request_uri;
```

- Перенаправить на корректный домен

```
server {
  listen 80 default_server;
  listen 443 ssl default_server;
  server_name _;
  return 301 $scheme://www.example.com;
}
```

rewrite

Синтаксис

```
rewrite regex URL [flag];
```

- **`last`**
  - завершаем обработку `ngx_http_rewrite_module`
  - ищем новый `location`
- **`break`**
  - завершаем обработку `ngx_http_rewrite_module`
  - продолжаем в текущем `location`
- **`redirect`**
  - возвращает **`302`**
- **`permanent`**
  - возвращает **`301`**

rewrite

Преобразовать

```
/index?uid=425&doc=18
/users/425/documents/18
```

```
  location /index {
    if ($args ~ "^uid=(\d+)&doc=(\d+)") {
      set $key_uid $1;
      set $key_doc $2;
      rewrite ^.*$ /users/$key_uid/documents/$key_doc? last;
    }
  }
```

```
  location /index {
    if ($args ~ "^uid=(?P<key_uid>\d+)&doc=(?P<key_doc>\d+)") {
      rewrite ^.*$ /users/$key_uid/documents/$key_doc? last;
    }
  }
```

rewrite

- запрос: **/index?uid=425&doc=18**

```
location /index {
  proxy_pass http://127.0.0.1:8080/users/$arg_uid/documents/$arg_
}
```

- $arg_uid = 425
- $arg_doc = 18

- результат: **/users/425/documents/18**

try_files

```
location / {
    try_files $uri $uri/index.html $uri.html =404;
}
```

try_files

```
location / {
    try_files /system/maintenance.html
              $uri $uri/index.html $uri.html
              @namedloc;
}

location @namedloc {
    proxy_pass http://127.0.0.1:8080;
}
```

План занятия:

- Установка из репозитория. Структура директорий
- Базовый конфиг, include
- Директивы и Контексты, что это?
- Порядок обработки запроса (директива `server`)
- location
- proxy_pass, fcgi, uwsgi
- return, rewrite, try_files
- **error_page**
- Сжатие трафика
- Кэширование трафика
- Отдача статики

error_page

Синтаксис:

```
error_page error_code ... [=[ответ]] uri;
```

Возвращаем страничку:

```
error_page 404              /404.html;
error_page 500 502 503 504 /50x.html;
```

Возвращаем с другим кодом:

```
error_page 404 =200 /404.html;
```

error_page

Именованый location

```
location / {
    error_page 404 = @error_backend;
}
location @error_backend {
    proxy_pass http://backend_for_error;
}
```

Сжатие трафика

Синтаксис:

```
gzip              on|off;
gzip_types        text/html;
```

Типичный сайт:

```
gzip_types text/css application/javascript application/json;
```

Кэширование трафика

Настраиваем:

```
proxy_cache_path /tmp/nginx/cache levels=1:2 keys_zone=proxy_zone
proxy_cache proxy_zone;
proxy_cache_methods GET HEAD;
proxy_cache_valid  200 302  60m;
```

Проверяем заголовок:

```
add_header X-Proxy-Cache $upstream_cache_status;
```

Подключаем AWS S3

Генерируем секретный ключ для авторизации в AWS S3.

```
echo -n "user:nginx-demo\npassword:nginx-demo-2020-05-27" | \
sha256sum | sed 's/\s\+.*-.*$//' | base64 -w 0

NjEyMmQxMzlkY2MyM2MzYzYwMThkYzU5ZjQ2YTg2ZjQ0N2Y1ZGNlNzdiZTcxMWUzM
```

Подключаем AWS S3

AWS S3 bucket policy:

```
{
    "Version": "2012-10-17",
    "Id": "nginx-demo-policy",
    "Statement": [
        {
            "Sid": "Static for nginx-demo.",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::nginx-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:Referer": "NjEyMmQxMzlkY2MyM2MzYzYwMThkY
                }
            }
        }
```

Подключаем AWS S3

Проверка: Без заголовка

```
curl -I https://s3.eu-central-1.amazonaws.com/nginx-demo-bucket/m
HTTP/1.1 403 Forbidden
```

С заголовком

```
curl -I https://s3.eu-central-1.amazonaws.com/nginx-demo-bucket/m
-H 'Referer:NjEyMmQxMzlkY2MyM2MzYzYwMThkYzU5ZjQ2YTg2ZjQ0N2Y1ZGNlN
HTTP/1.1 200 OK
```

Конфигурация для otus.vscoder.ru

```
proxy_cache_path /tmp/nginx/cache levels=1:2 keys_zone=proxy_zone
server {
        root /srv/www/site;
        location /fonts/ {
                rewrite ^/fonts/(.*) /gallery/$1 break;
        }
        location /images/ {
                proxy_pass https://s3.eu-central-1.amazonaws.com/
                proxy_set_header Referer 'NjEyMmQxMzlkY2MyM2MzYzY
                proxy_hide_header x-amz-id-2;
                proxy_hide_header x-amz-request-id;
                proxy_intercept_errors on;
                error_page 403 =404 @aws_errors;
                proxy_cache proxy_zone; # Cache
                proxy_cache_methods GET HEAD; # Cache
                proxy_cache_valid 200 60m; # Cache
```

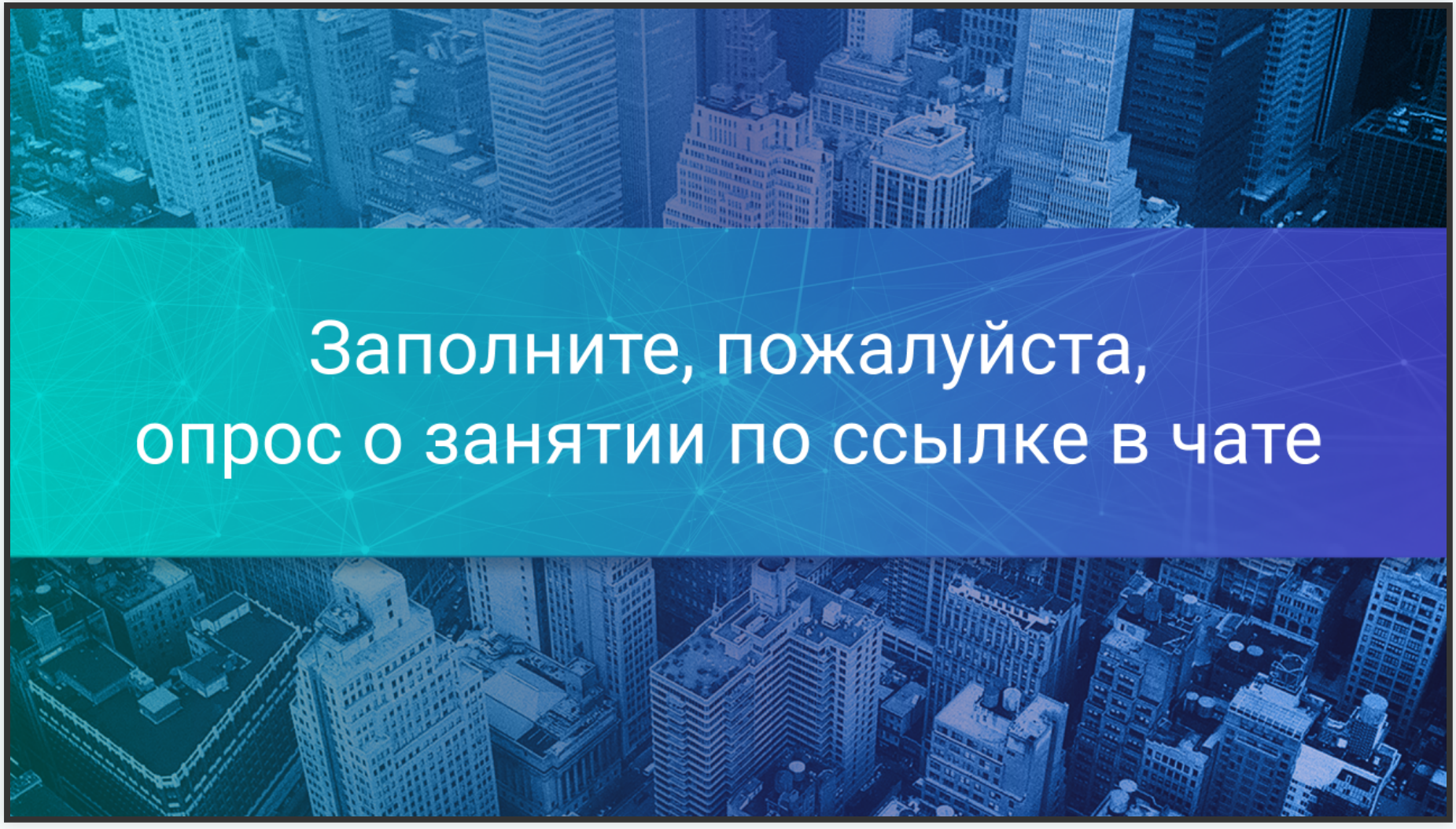
Конфигурация для otus.vscoder.ru

```
        location @aws_errors {
                add_header Content-Type text/html always;
                # add_header Content-Type image/jpeg always;
                return 404 "<html><body><img src='https://c7.hotp
                <br>Was requested: $request_uri<br></body></html>
        }
}
```

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