



ОНЛАЙН-ОБРАЗОВАНИЕ

Домашнее задание

- 1) Создать свой RPM пакет (можно взять свое приложение, либо собрать, например, апач с определенными опциями)
- 2) Создать свой репозиторий и разместить там ранее собранный RPM

Реализовать это все либо в Vagrant, либо развернуть у себя через NGINX и дать ссылку на репозиторий.

- Для данного задания нам понадобятся следующие установленные пакеты:

```
[root@packages ~]# yum install -y \  
redhat-lsb-core \  
wget \  
rpmdevtools \  
rpm-build \  
createrepo \  
yum-utils
```

Создать свой RPM пакет

- Для примера возьмем пакет **NGINX** и соберем его с поддержкой **openssl**
- Загрузим **SRPM** пакет **NGINX** для дальнейшей работы над ним:

```
[root@packages ~]# wget  
https://nginx.org/packages/centos/7/SRPMS/nginx-1.14.1-1.el7_4.ngx.src.rpm
```

- При установке такого пакета в домашней директории создается древо каталогов для сборки:

```
[root@packages ~]# rpm -i nginx-1.14.1-1.el7_4.ngx.src.rpm
```

- Также нужно скачать и разархивировать последний исходники для **openssl** - он потребуется при сборке

```
[root@packages ~]# wget https://www.openssl.org/source/latest.tar.gz
```

```
[root@packages ~]# tar -xvf latest.tar.gz
```

- Заранее поставим все зависимости чтобы в процессе сборки не было ошибок

```
[root@packages ~]# yum-builddep rpmbuild/SPECS/nginx.spec
```

- Ну и собственно поправить сам spec файл чтобы **NGINX** собирался с необходимыми нам опциями:
 - Результирующий spec файл получился [таким](#)
 - Обратите внимание что путь до **openssl** указываем ДО каталога:
`--with-openssl=/root/openssl-1.1.1a`
- По этой [ссылке](#) можно посмотреть все доступные опции для сборки.

Создать свой RPM пакет

- Теперь можно приступить к сборке **RPM** пакета:

```
[root@packages ~]# rpmbuild -bb rpmbuild/SPECS/nginx.spec
```

```
...
```

```
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.XodhnN
```

```
+ umask 022
```

```
+ cd /root/rpmbuild/BUILD
```

```
+ cd nginx-1.14.1
```

```
+ /usr/bin/rm -rf /root/rpmbuild/BUILDROOT/nginx-1.14.1-1.el7_4.ngx.x86_64
```

```
+ exit 0
```

- Убедимся что пакеты создались:

```
[root@packages ~]# ll rpmbuild/RPMS/x86_64/
```

```
-rw-r--r--. 1 root root 1999864 Nov 29 06:15 nginx-1.14.1-1.el7_4.ngx.x86_64.rpm
```

```
-rw-r--r--. 1 root root 2488840 Nov 29 06:15 nginx-debuginfo-1.14.1-1.el7_4.ngx.x86_64.rpm
```

- Теперь можно установить наш пакет и убедиться что nginx работает

```
[root@packages ~]# yum localinstall -y \
rpmbuild/RPMS/x86_64/nginx-1.14.1-1.el7_4.ngx.x86_64.rpm
```

```
[root@packages ~]# systemctl start nginx
```

```
[root@packages ~]# systemctl status nginx
```

- nginx.service - nginx - high performance web server
Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
Active: **active (running)** since Thu 2018-11-29 07:34:19 UTC; 14min ago

- Далее мы будем использовать его для доступа к своему репозиторию

- Теперь приступим к созданию своего репозитория. Директория для статики у **NGINX** по умолчанию **/usr/share/nginx/html**. Создадим там каталог **repo**:

```
[root@packages ~]# mkdir /usr/share/nginx/html/repo
```

- Копируем туда наш собранный RPM и, например, RPM для установки репозитория Percona-Server:

```
[root@packages ~]# cp rpmbuild/RPMS/x86_64/nginx-1.14.1-1.el7_4ngx.x86_64.rpm  
/usr/share/nginx/html/repo/
```

```
[root@packages ~]# wget  
http://www.percona.com/downloads/percona-release/redhat/0.1-6/percona-release-0.1-6.noa  
rch.rpm -O /usr/share/nginx/html/repo/percona-release-0.1-6.noarch.rpm
```

- Инициализируем репозиторий командой:

```
[root@packages ~]# createrepo /usr/share/nginx/html/repo/
```

```
Spawning worker 0 with 2 pkgs
```



Видим что в репозитории два пакета

```
Workers Finished
```

```
Saving Primary metadata
```

```
Saving file lists metadata
```

```
Saving other metadata
```


```
Generating sqlite DBs
```



Обратите внимание что используется **sqlite**

```
Sqlite DBs complete
```

- Для прозрачности настроим в NGINX доступ к листингу каталога:
- В location / в файле /etc/nginx/conf.d/default.conf добавим директиву autoindex on. В результате location будет выглядеть так:

```
location / {  
    root /usr/share/nginx/html;  
    index index.html index.htm;  
    autoindex on;  Добавили эту директиву  
}
```

- Проверяем синтаксис и перезапускаем NGINX:

```
[root@packages ~]# nginx -t
```

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
```

```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

```
[root@packages ~]# nginx -s reload
```

- Теперь ради интереса можно посмотреть в браузере или curl-ануть:

```
[root@packages ~]# lynx http://localhost/repo/
```

```
[root@packages ~]# curl -a http://localhost/repo/
```

```
<html>
<head><title>Index of /repo/</title></head>
<body bgcolor="white">
<h1>Index of /repo/</h1><hr><pre><a href="..">../</a>
<a href="repodata/">repodata/</a>                                29-Nov-2018 10:23      -
<a href="nginx-1.14.1-1.el7_4.ngx.x86_64.rpm">nginx-1.14.1-1.el7_4.ngx.x86_64.rpm</a>
    29-Nov-2018 09:47                1999600
<a href="percona-release-0.1-6.noarch.rpm">percona-release-0.1-6.noarch.rpm</a>
    13-Jun-2018 06:34                14520
</pre><hr></body>
</html>
```

- Все готово для того, чтобы протестировать репозиторий.
- Добавим его в `/etc/yum.repos.d`:

```
[root@packages ~]# cat >> /etc/yum.repos.d/otus.repo << EOF
[otus]
name=otus-linux
baseurl=http://localhost/repo
gpgcheck=0
enabled=1
EOF
```

- Убедимся что репозиторий подключился и посмотрим что в нем есть:

```
[root@packages ~]# yum repolist enabled | grep otus
```

```
otus otus-linux 2
```

```
[root@packages ~]# yum list | grep otus
```

nginx	1.14.1	otus
percona-release.noarch	0.1-6	otus

- Так как **NGINX** у нас уже стоит установим репозиторий **percona-release**:

```
[root@packages ~]# yum install percona-release -y
```

- Все прошло успешно. В случае если вам потребуется обновить репозиторий (а это делается при каждом добавлении файлов), снова то выполните команду **createrepo /usr/share/nginx/html/repo/**