**Создание виртуальной машины.**

Будем делать конфигурацию: Ubuntu 12.04, NodeJs, Mysql.

1. **Создание и минимальная настройка виртуальной машины.**
2. Скачать <https://www.virtualbox.org/wiki/Downloads>;
3. Скачать установочный образ ОС: Ubuntu 12.04;

Можно другую, но хостинг у именно этой версии.  
Например попробовать этот url: <http://archive.ubuntu.com/ubuntu/dists/precise-updates/main/installer-i386/current/images/netboot/mini.iso>;

1. Установить VirtualBox;
2. Откроем VirtualBox, перейдем Файл-> Настройки-> Сеть-> Виртуальные сети хоста;
3. Нажать [Добаить виртуальную сеть хоста] (кнопка с красным плюсиком);
4. Выбрать новую запись и нажать [Изменить виртуальную сеть хоста] (кнопка с картой и отвёрткой);
5. Прописываем поле `адрес`: “192.168.100.1”;
6. Прописываем поле `маска подсети`: «255.255.255.0»;
7. Добавляем строку в файл C:\Windows\system32\drivers\etc\hosts, “192.169.100.2 krestiki-noliki”;
8. Создаем виртуальную машину;
9. Идём в настройки виртуальной машины в VirtualBox-е
10. Открываем Настройки-> Сеть -> Адаптер 2, далее устанавливаем следующие поля:  
    - Включить сетевой адаптер – ставим галку  
    - Тип подключения : «Виртуальный адаптер хоста»  
    - Имя: VirtualBox Host-Only Ethernet Adapter @#2
11. Устанавливаем скаченную ОС;

* primary network выбираем: eth0;
* имя хоста указать krestiki-noliki
* пароль для root: “root”
* на запрос нового пользователя указываем: login: “xo”, password: “xo”.

1. Опять логинимся и выполняем команду: «apt-get update && apt-get upgrade.”.
2. **Настраиваем разрешение ОС(опционально).**

Настроем разрешение экран ОС, дело в том, что по умолчанию у нас 640х480, это маленькое окно и нам будет сложно работать с консолью.

Если GRUB лоадер не грузиться, то можно пропустить пункты 1-3, это не критично.

1. Запускаем виртуальную машину, во время загрузки будет менеджер ОС, нажимаем клавишу ‘c’, когда он появился и попадаем в консоль.
2. Набираем vbeinfo, команда выведет список допустимых разрешений, выберите удобное, например, 1024х768
3. Набираем reboot.
4. Логинимся в систему.
5. Редактируем файл /etc/default/grub
6. Находим строку “#GRUB\_FGXMODE=640х480”, раскоменчиваем и указываем значение из пункта 2. Получиться “GRUB\_FGXMODE=1024x768”.
7. Выполняем команду: update-grub2, и затем команду: reboot.
8. **Настраиваем сеть виртуальной машины.**
9. Открываем на редактирование файл /etc/network/interfaces
10. Добавляем строки:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
allow-hotplug eth1

Iface eth1 inet static

address 192.168.100.2

netmask 255.255.255.0

network 192.168.100.0

broadcast 192.168.100.255

auto eth1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Перезагружаем машину: “reboot”;
2. Выполняем команду: apt-get install ssh;
3. Теперь машина доступна с локального компьютера, можно подключаться по SSH, а так же по SFTP , FTP.
4. Проверяем простым пингом. Должна пинговаться виртуальная машина с локальной и наоборот.
5. **Установка NodeJS**

Выполняем команды:

1. apt-get install curl
2. curl –sL https://deb.nodesource.com /setup | bash –
3. apt-get install nodejs
4. **Установка MySQL**

1. apt-get install mysql-server

2. пароль для root-а: root

1. **Установка git**
2. apt-get install git
3. git config –global core.autocrlf true
4. **Настройка deployment PhpStorm.**
5. Создадим папку на виртуальной машине для проекта:

cd / && cd var && mkdir xo && cd xo

1. Так же надо, что бы у пользователя xo были права на папку, например можно сделать так:

chown xo xo.

1. Идём в настройки phpStorm-а: Tools->Deplayment->Configure
2. @todo

**VIII. Install nGinx**

nginx 1.1.19 update  
> apt-get purge nginx  
> apt-get install python-software-properties  
> apt-get install software-properties-common  
> add-apt-repository ppa:nginx/stable  
> apt-get update  
> apt-get upgrade  
> apt-get install nginx

/etc/nginx/nginx.conf

server {

listen 80;

listen 443 ssl;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

#ssl\_password\_file /etc/keys/global.pass;

#server\_name ssl.krestiki-noliki.net;

server\_name ssl.krestiki-noliki;

ssl\_certificate /var/xo/server/ssl/ssl.crt;

ssl\_certificate\_key /var/xo/server/ssl/ssl.key;

location / {

proxy\_pass http://127.0.0.1:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

location /images {

root /var/xo;

access\_log /var/log/nginx/access\_images.log;

error\_log /var/log/nginx/error\_images.log;

access\_log off;

}

location /js {

access\_log /var/log/nginx/access\_js.log;

error\_log /var/log/nginx/error\_js.log;

root /var/xo;

}

}

**/etc/init.d/xoserver**

#!/bin/bash

# xoserver daemon

# chkconfig: 345 20 80

# description: xoserver daemon

# processname: xoserver

DAEMON\_PATH="/var/xo/server/"

DAEMON="run.sh"

DAEMONOPTS=""

OUTPUT\_FILE="/var/xo/server/xoserver\_last.out"

NAME=xoserver

DESC="XO service daemon description"

PIDFILE=/var/run/$NAME.pid

SCRIPTNAME=/etc/init.d/$NAME

case "$1" in

start)

printf "%-50s" "Starting $NAME..."

cd $DAEMON\_PATH

PID=`./$DAEMON $DAEMONOPTS > $OUTPUT\_FILE 2>&1 & echo $!`

echo "Saving PID" $PID " to " $PIDFILE

if [ -z $PID ]; then

printf "%s\n" "Fail"

else

echo $PID > $PIDFILE

printf "%s\n" "Ok"

fi

;;

status)

printf "%-50s" "Checking $NAME..."

if [ -f $PIDFILE ]; then

PID=`cat $PIDFILE`

if [ -z "`ps axf | grep ${PID} | grep -v grep`" ]; then

printf "%s\n" "Process dead but pidfile exists"

else

echo "Running"

fi

else

printf "%s\n" "Service not running"

fi

;;

stop)

printf "%-50s" "Stopping $NAME"

PID=`cat $PIDFILE`

cd $DAEMON\_PATH

if [ -f $PIDFILE ]; then

kill -HUP $PID

printf "%s\n" "Ok"

rm -f $PIDFILE

else

printf "%s\n" "pidfile not found"

fi

;;

restart)

$0 stop

$0 start

;;

\*)

echo "Usage: $0 {status|start|stop|restart}"

exit 1

esac