

АВТОМАТИЗАЦИЯ РАЗВЕРТЫВАНИЯ VDI
РЕШЕНИЯ НА ОСНОВЕ ЛЕГКОВЕСНОЙ
ВИРТУАЛИЗАЦИИ

ВВЕДЕНИЕ

- Целью данной выпускной квалификационной работы является автоматизация создания и управление VDI решением на основе Linux контейнеров.
- Я напишу сценарии для автоматизации создания и управления контейнерами, которые будут представлять собой виртуальные рабочие места и протестирую удаленное подключение к ним.

Технология VDI

- VDI (Virtual Desktop Infrastructure) – это технология, которая позволяет разворачивать полноценные рабочие места прямо на серверах. Технология VDI позволяет создавать виртуальные персональные компьютеры, которые могут быть централизованно развернуты на базе одного сервера, либо группы серверов.

Задачи

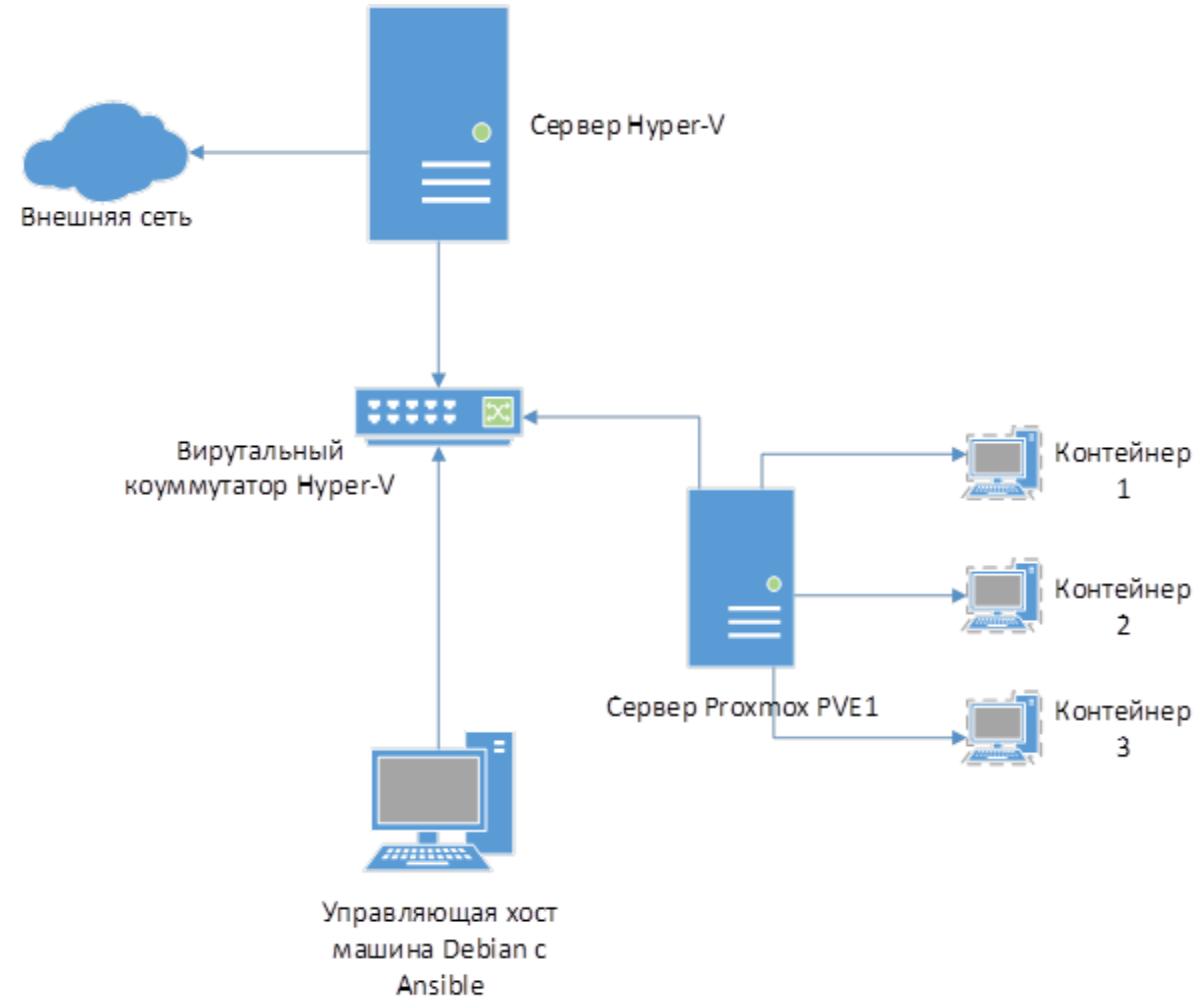
Для достижения цели моей работы необходимо решить следующие задачи:

- 1) установить и настроить сервер виртуализации, на котором будут разворачиваться и работать виртуальные рабочие столы;
- 2) развернуть хост машину, на которой будет установлена система автоматизации для управления сервером виртуализации;
- 3) создать контейнеры с помощью средства автоматизации;
- 4) установить необходимые пакеты и обновления в контейнеры для создания полноценного рабочего стола с помощью средства автоматизации;
- 5) установить и настроить средство удаленного доступа к виртуальному рабочему столу;

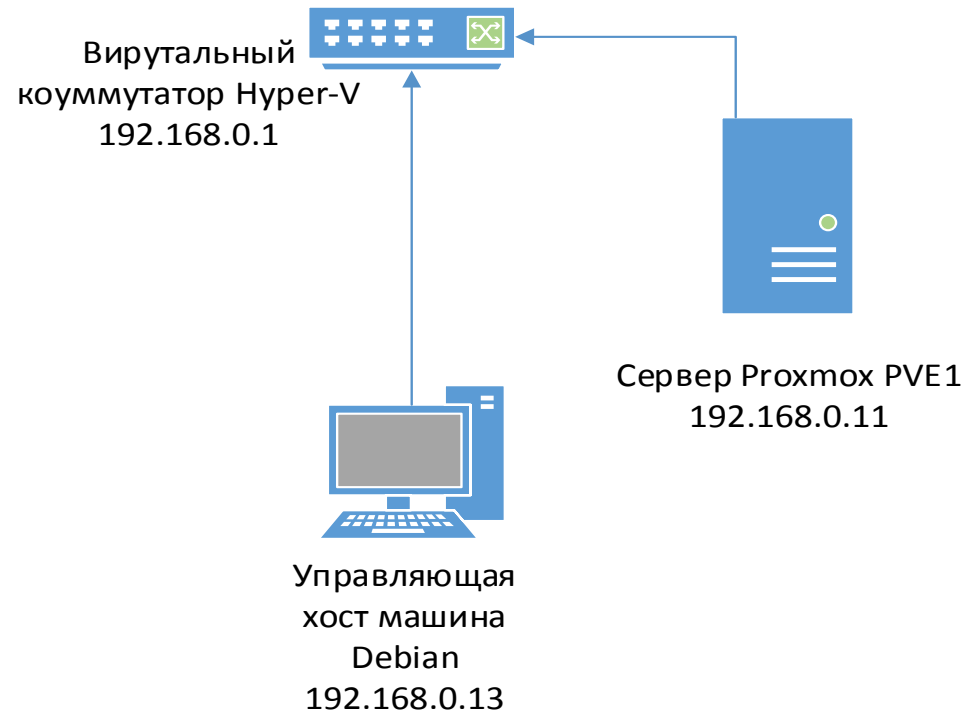
Инструменты, использованные в работе:

- Proxmox , как сервер виртуализации.
- Linux containers LXC в качестве легковесных контейнеров.
- Ansible в качестве средства автоматизации. Ansible — система управления конфигурациями.
- X2go В качестве средства удалённого доступа. X2go - это программное обеспечение с открытым исходным кодом для удаленного подключения к машинам Linux, использующее протокол NX.
- Хост машина с установленной ОС Debian. С этой машины будет происходить управление сервером виртуализации Proxmox, с помощью Ansible.

Схема стенда



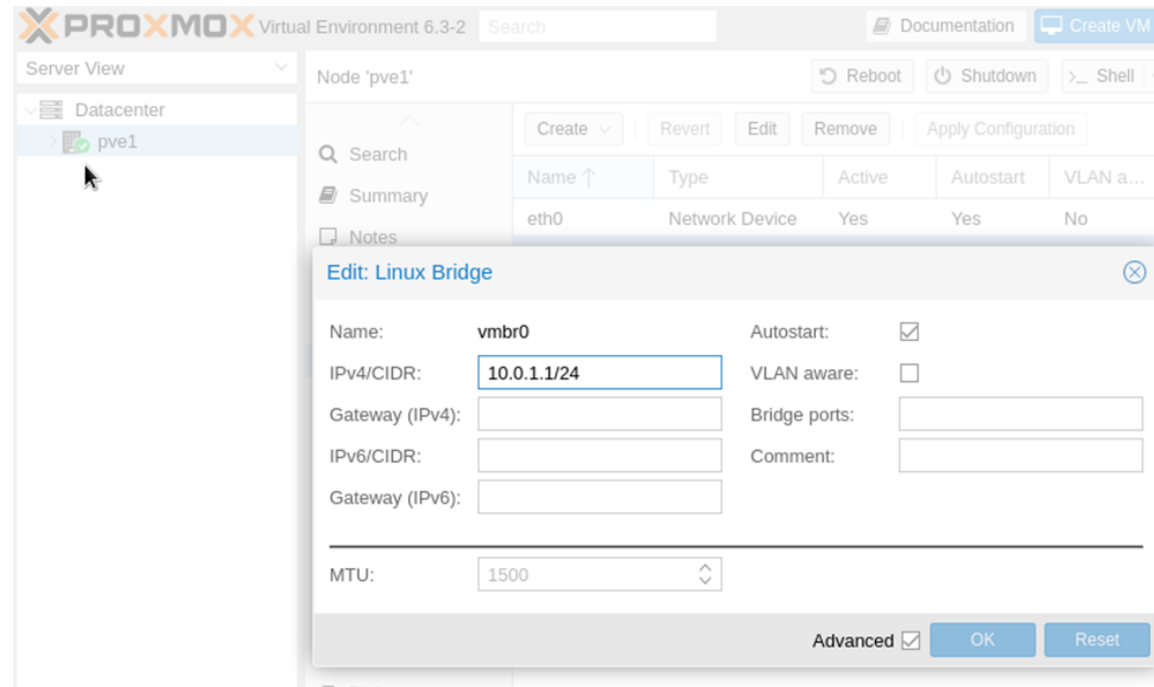
На первом этапе работы, получаем следующую схему:



После того, как удалось проверить доступность хоста Debian и сервера Proxmox, с помощью команды `ping`, необходимо обменяться SSH ключами с помощью команды `ssh-copy-id`

Настройка сетевого моста

Важным этапом, является создание виртуального сетевого моста `vbr0`, для того чтобы контейнеры имели доступ в сеть. `vbr0` будет иметь IP адрес `10.0.1.1/24`:



Контейнеры будут находится в сети `10.0.1.0/24`, но для того чтобы у них был доступ к сети нужно настроить так называемый «Маскарадинг», для этого в файл `/etc/network/interfaces` сервера Proxmox, необходимо приписать настройки:

Настройки Masquerade:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.0.11/24
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1 8.8.8.8

auto vubr0
iface vubr0 inet static
    address 10.0.1.1/24
    dns-nameservers 192.168.0.1 8.8.8.8
    bridge-ports none
    bridge-stp off
    bridge-fd 0

post-up echo 1 > /proc/sys/net/ipv4/ip_forward
post-up iptables -t nat -A POSTROUTING -s '10.0.1.0/24' -o eth0 -j MASQUERADE
post-down iptables -t nat -D POSTROUTING -s '10.0.1.0/24' -o eth0 -j MASQUERADE
```

Установка пакета Ansible на хост Debian

1. Установка дополнительных библиотек python
2. Добавление репозитория Ansible
3. Установка пакета Ansible
4. Установка дополнительных библиотек для управления сервером Proxmox: `pip install proxmoxer` и `pip install requests`;

Основные настройка Ansible

Файл ansible.cfg:

```
[defaults]
# some basic default values...

inventory      = /etc/ansible-proxmox/hosts
#library       = /usr/share/my_modules/
#module_utils  = /usr/share/my_module_utils/
#remote_tmp    = ~/.ansible/tmp
#local_tmp     = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
```

Файл hosts для доступа к серверу Proxmox:

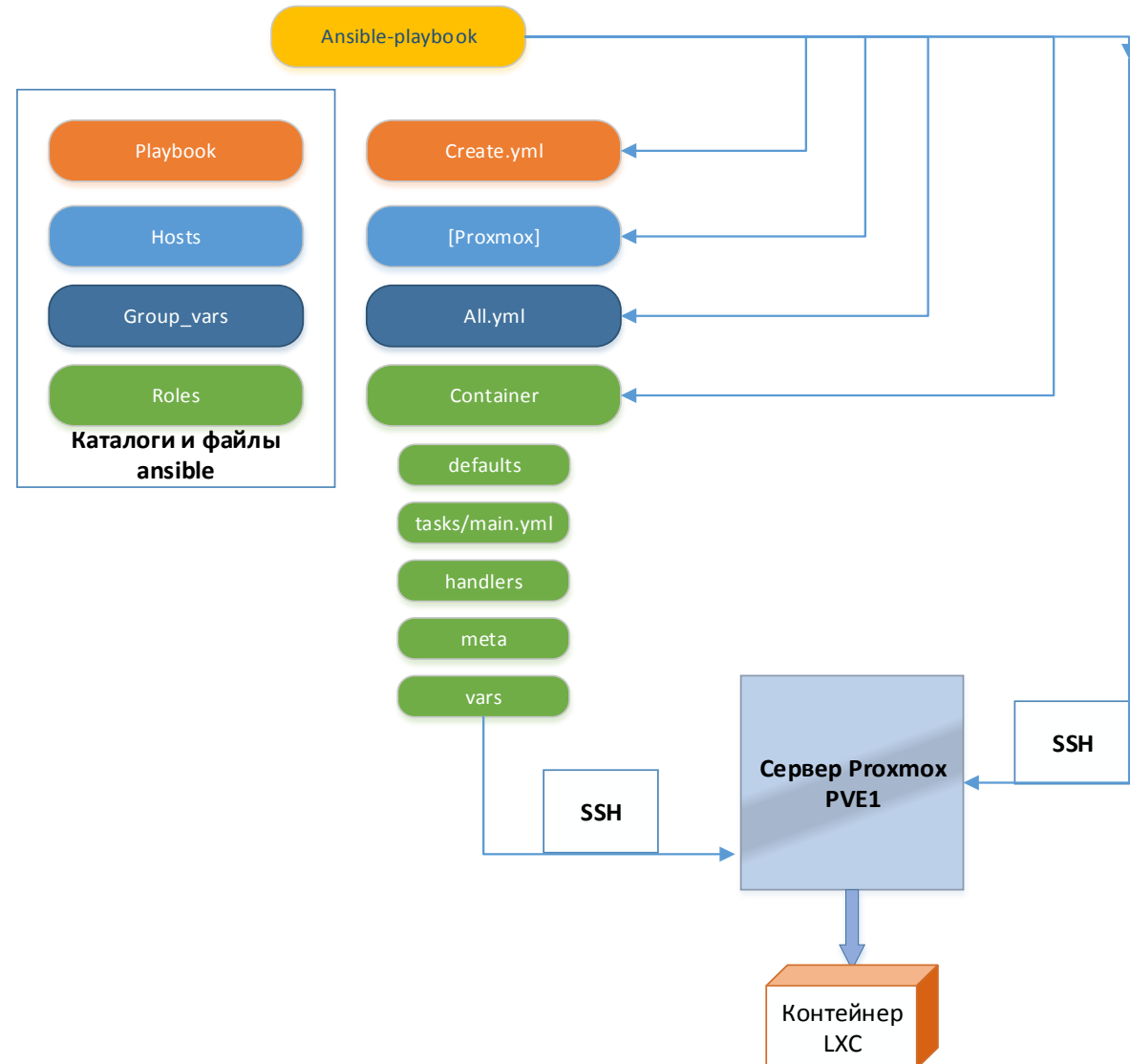
```
GNU nano 3.2                                hosts

[proxmox]
pve1.local ansible_host=192.168.0.11 ansible_user=root ansible_ssh_private_key_file=/root/.ssh/id_rsa
```

Playbook для создания контейнера Proxmox

Данный playbook(приложение 1) создает контейнер и запускает его сразу после создания.

- В `role/container/tasks/main.yml`, находится основное задание
- В `group_vars/all.yml` находятся Переменные для основного задания (`hostname`, `ip`, пароли и т.д.)
- Файл `create.yml` запускает сам `playbook`



Обмен SSH ключами с контейнерами LXC, обновление файла hosts Ansible

На этом этапе, после создания контейнеров, необходимо сделать их доступными для хоста с Ansible:

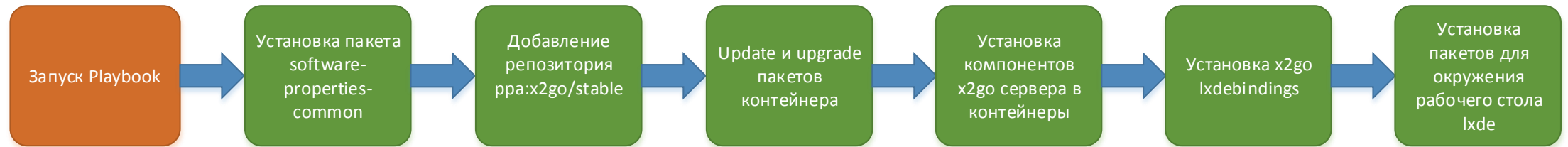
1. Нужно добавить маршрут до сети 10.0.1.0/24, в которой находятся контейнеры, с помощью команды *post-up ip route add 10.0.1.0/24 via 192.168.0.11*.
2. Обменяться ключами шифрования со всеми контейнерами с помощью команды *ssh-copy-id*
3. В каталоге ansible в файл hosts необходимо добавить группу [containers], для того чтобы Ansible мог к ним обращаться, в результате файл hosts теперь выглядит так:

```
[прохмох]
pve1.local ansible_host=192.168.0.11 ansible_user=root ansible_ssh_private_key_file=/root/.ssh/id_rsa

[containers]
vd1 ansible_host=10.0.1.2 ansible_user=root ansible_ssh_private_key_file=/root/.ssh/id_rsa
vd2 ansible_host=10.0.1.3 ansible_user=root ansible_ssh_private_key_file=/root/.ssh/id_rsa
vd3 ansible_host=10.0.1.4 ansible_user=root ansible_ssh_private_key_file=/root/.ssh/id_rsa
```

Playbook Ansible, для установки дополнительных пакетов, установки обновлений и установки сервера терминального доступа x2go в контейнер

На этом этапе создается playbook (приложение 2), который «превращает» контейнеры LXC в виртуальные рабочие столы. В моей работе я использовал окружение рабочего стола LXDE. Playbook последовательно запускает установку и обновления пакетов на всех созданных контейнерах. Последовательность выглядит следующим образом:

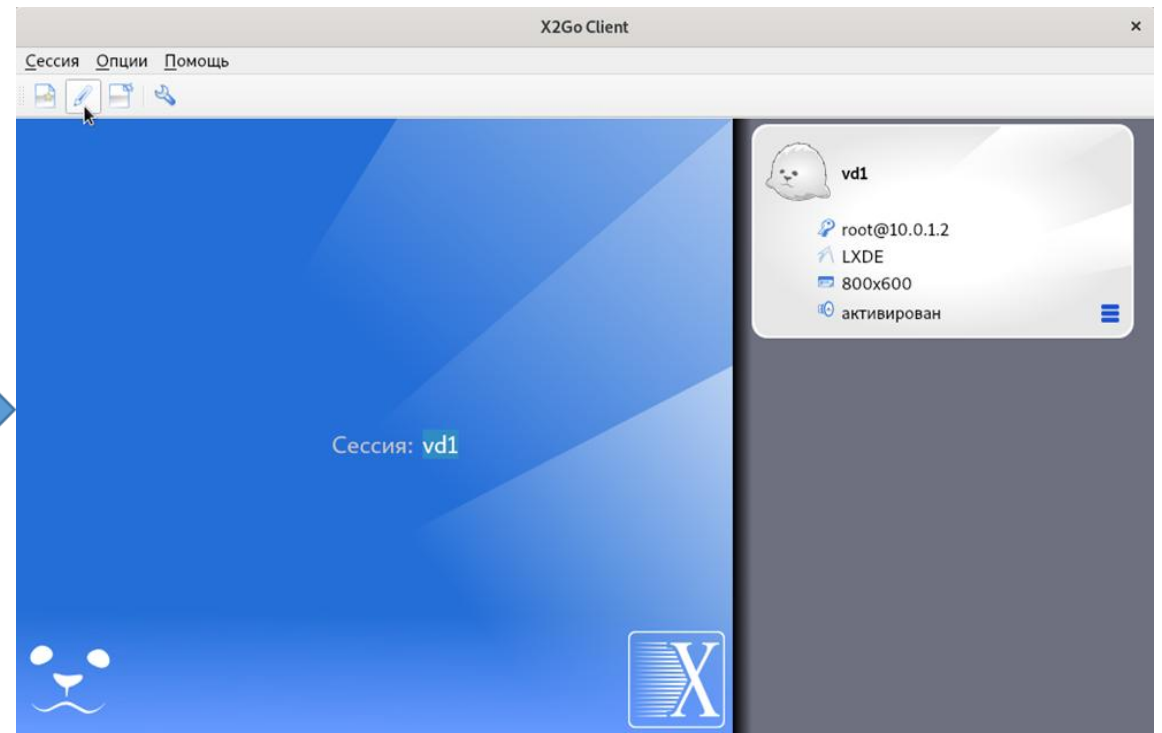
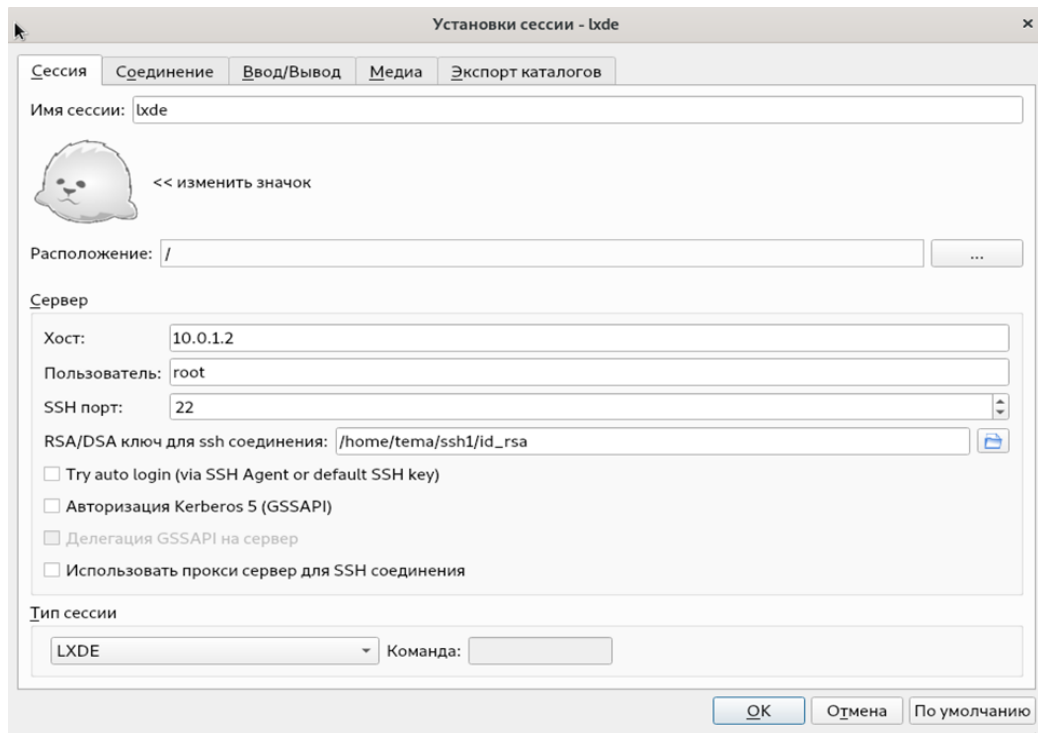


В итоге получаются виртуальные рабочие столы, на основе окружения LXDE, с установленным терминальным сервером x2go.

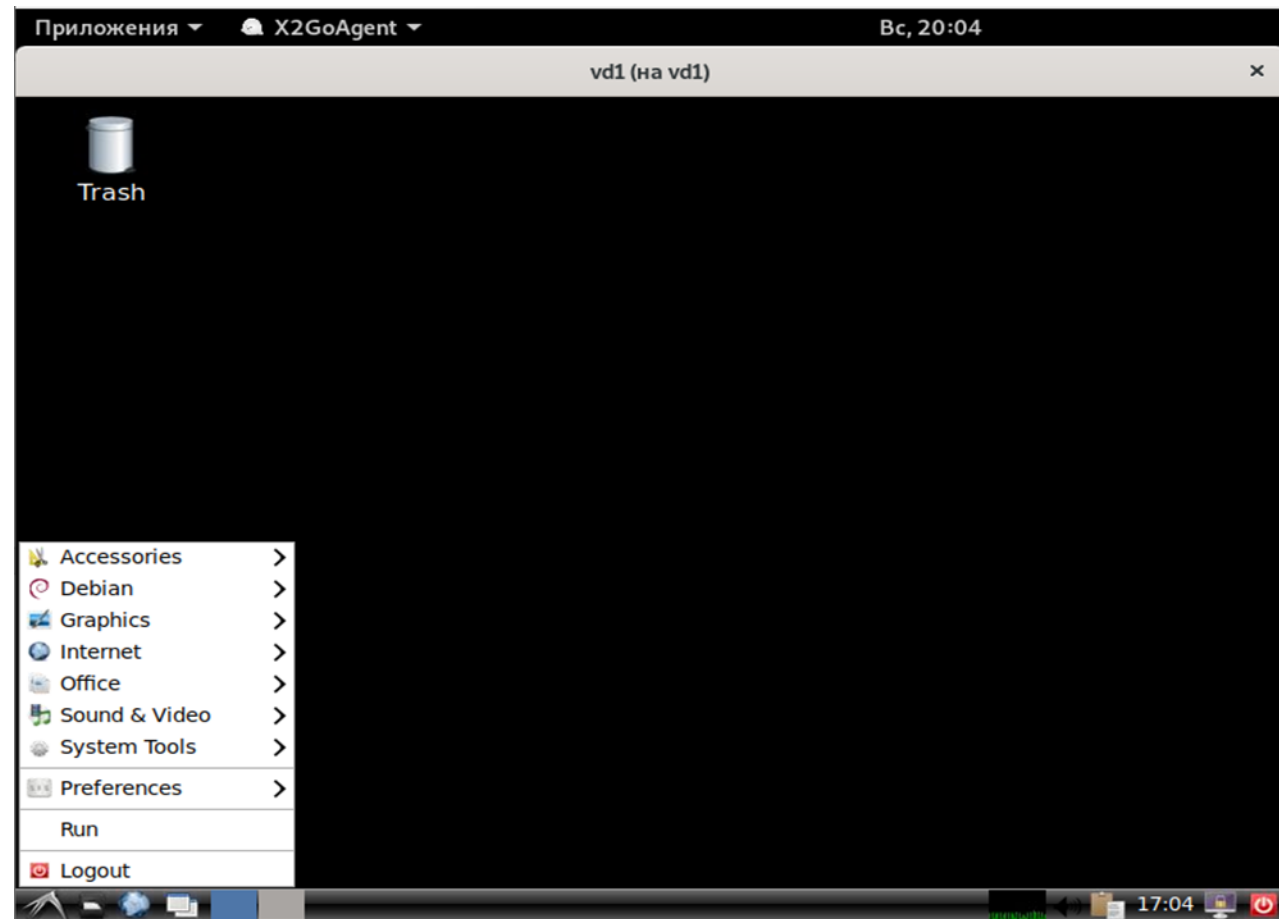
Установка x2go клиента на хост машину с Debian и проверка подключения

На этом этапе происходит установка клиента x2go на хост Debian, я использую ту же машину, на которой установлен Ansible.

Для установки используем команду `apt install x2goclient`, после установки запускаем клиент.

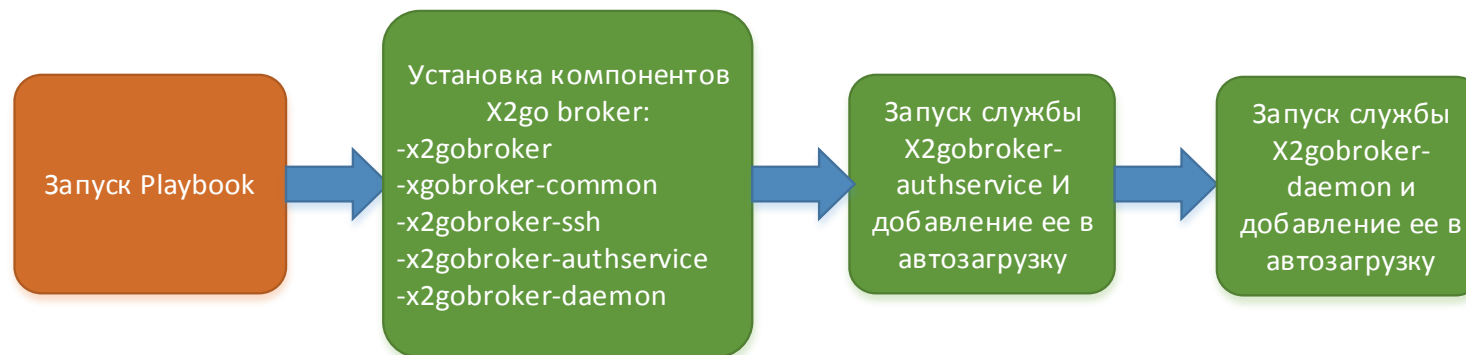


Вводим логин и пароль и подключаемся к рабочему столу:



Плейбук Ansible для установки брокера терминального соединения x2go broker

Т.к. каждый раз прописывать параметры виртуального рабочего стола пользователю будет не удобно, необходимо установить x2go broker. X2go broker позволит определять сессию на самой виртуальной машине. Для установки компонентов будет создан playbook, который запускает следующую последовательность действий на всех контейнерах:

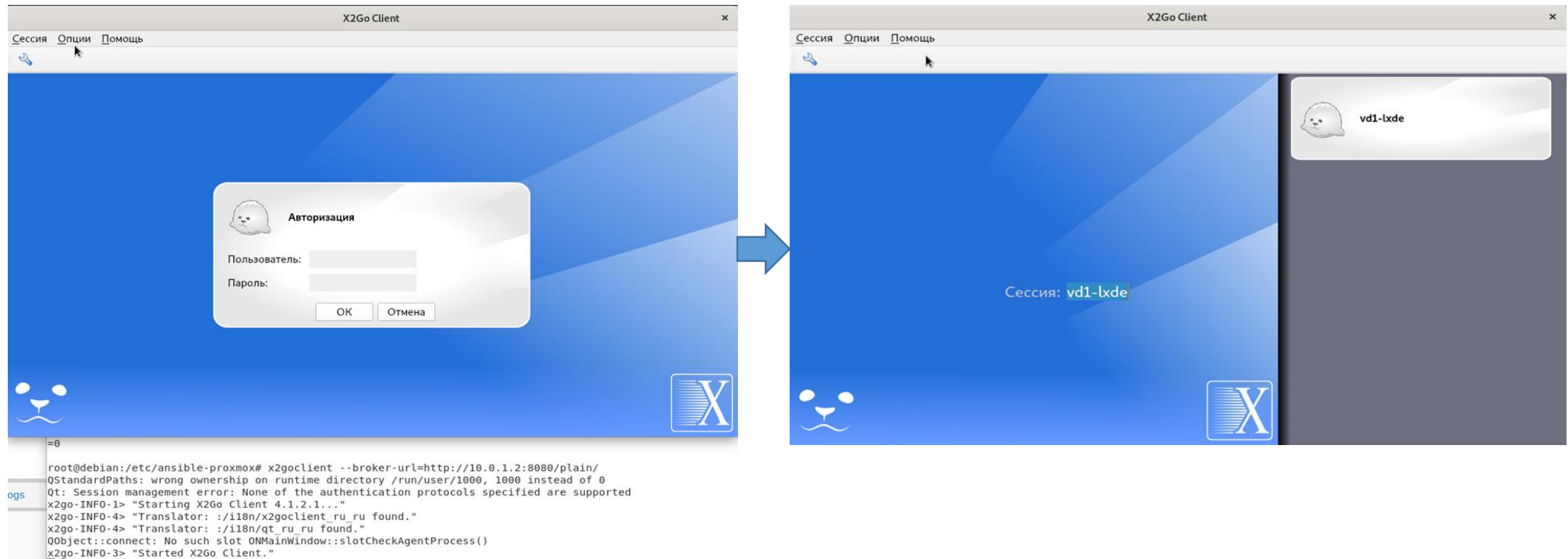


После установки компонентов, необходимо, зайти на один из виртуальных столов и отредактировать файл `nano /etc/x2go/broker/x2gobroker-sessionprofiles.conf`, чтобы определить сессию на самом виртуальном рабочем столе:

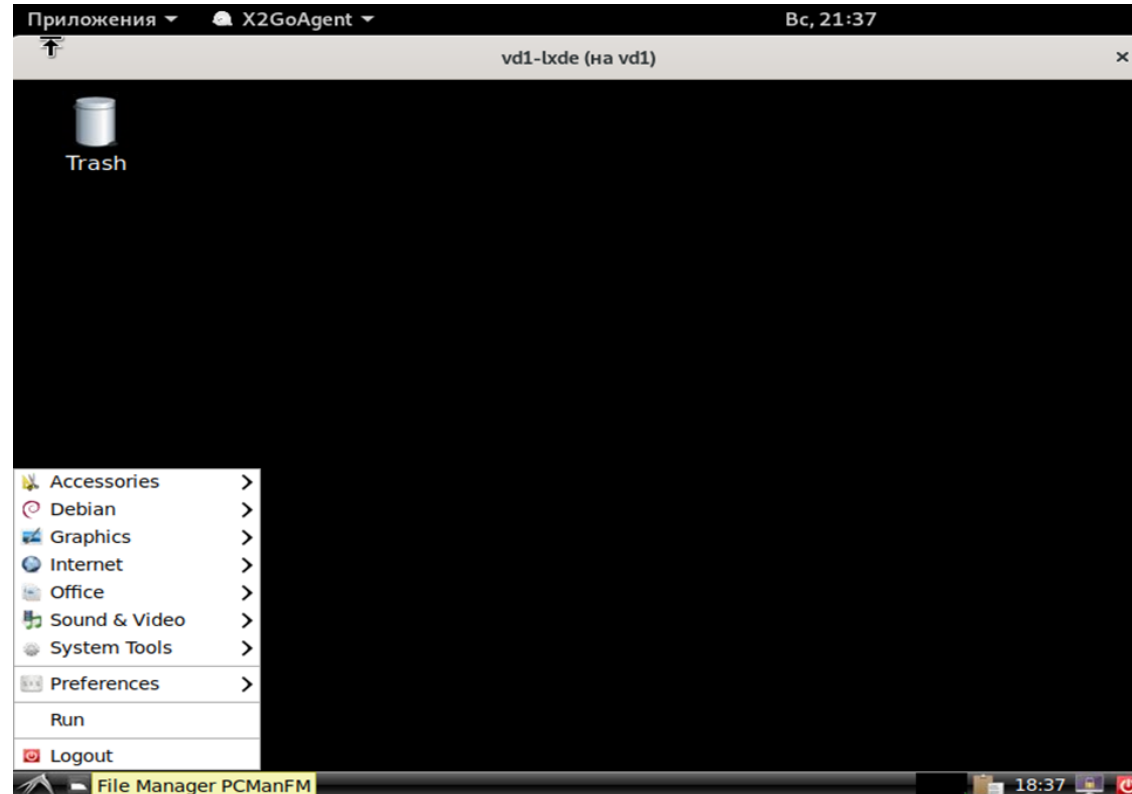
```
/etc/x2go/broker/x2gobroker-sessionprofiles.conf  
  
[vd1-lxde]  
name=vd1-lxde  
host=10.0.1.2  
command=lxde  
userbrokerpass=true
```

Проверка подключения к виртуальному рабочему столу

Теперь, на машине, на которой установлен x2go клиент, необходимо запустить команду: `x2goclient --broker-url=http://10.0.1.2:8080/plain/`, После запуска команды увидим окно:



На следующем экране вводим учетные данные пользователя и подключаемся к рабочему столу:



Для удобства можно сразу сделать из команды `x2goclient --broker url=http://10.0.1.2:8080/plain/` скрипт. Для этого необходимо ввести команду `echo x2goclient --broker url=http://10.0.1.2:8080/plain/ >> /home/x2go.sh`.

Заключение

Во время выполнения работы удалось реализовать все поставленные задачи в тестовой среде:

1. Удалось написать плейбуки Ansible, которые автоматизируют создание и управление контейнерами LXC.
2. С помощью плейбуков удалось превратить контейнер в виртуальное рабочее место.
3. Удалось установить и настроить терминальный доступ к виртуальным рабочим местам при помощи автоматизации процесса.
4. Удалось упростить процесс подключения к удаленному рабочему месту.
5. Все этапы протестированы и работают.

Считаю, поставленные передо мной задачи, выполнены.

Спасибо за внимание!