

1. Artem Kosenko's Home	3
1.1 Linux Basics	4
1.1.1 01. Vagrant & VirtualBox	4
1.1.2 02. Linux Distributions	9
1.1.2.1 Users & Groups. Permissions. Umask	15
1.1.2.2 SUDO	21
1.1.2.3 Package Managers	22
1.1.2.3.1 APT	22
1.1.2.3.2 Pacman & Yaourt	23
1.1.2.3.3 RPM	24
1.1.2.3.4 Yum (Yellowdog Updater, Modified)	25
1.1.3 03. Linux Boot	27
1.1.3.1 GRUB	30
1.1.3.2 Daemon Control Systems	32
1.1.3.2.1 Systemd	32
1.1.3.2.2 UNIX System V	43
1.1.3.2.3 Upstart	44
1.1.3.3 Processes & Threads	48
1.1.3.3.1 /proc/	48
1.1.3.3.2 /var/run/	52
1.1.4 04. Linux Storage & RAM	52
1.1.4.1 MBR vs GPT	52
1.1.4.2 parted (gpt разметка диска)	53
1.1.4.3 LVM	55
1.1.4.4 ext2, ext3, ext4	56
1.1.4.5 xfs	57
1.1.4.6 NFS, CIFS-UTILS	60
1.1.4.7 SWAP	62
1.1.4.8 RAM	63
1.1.5 05. Networking	64
1.1.5.1 iptables tutorial	64
1.1.5.2 IP - utility	64
1.1.5.3 OSI Model (1-7 lvl)	67
1.1.6 06. SSH advanced	68
1.1.7 07. Bash scripting	71
1.1.7.1 awk	71
1.1.7.2 sed	72
1.1.8 08. Internet services	74
1.1.8.1 DNS - BIND9	75
1.1.9 09. Linux Java stack	84
1.1.10 10. Virtualization and Containers	85
1.1.10.1 Docker	85
1.1.10.1.1 Docker. Changes	87
1.1.10.1.2 Docker. Common	87
1.1.10.1.3 Docker. Diagnostic Commands	88
1.2 SDLC approaches	90
1.2.1 01. Software lifecycle	90
1.2.2 02. Software version control systems	90
1.2.2.1 GIT	90
1.3 Project related CI/CD tools	92
1.3.1 01. CM tools	93
1.3.1.1 Ansible	93
1.3.1.2 Puppet	95
1.3.1.2.1 PCB. File	97
1.3.1.2.2 PCB. Iptables	99
1.3.1.2.3 PCB. Packages	100
1.3.1.2.4 PCB. php.ini	101
1.3.1.2.5 Puppet. Cert & Curl	101
1.3.1.2.6 Puppet. Hiera	103
1.3.1.2.7 Puppet. Ресурсы	104
1.3.1.2.8 Puppet. Структура каталогов	106
1.4 99. Other	107
1.4.1 Arch Linux	108
1.4.2 AWS	110
1.4.3 Bareos	110
1.4.3.1 Bareos. Bconsole commands	110
1.4.4 BASH SCRIPTS	112
1.4.5 Fix welcome message on Ubuntu's	113
1.4.6 HP-ILo	113
1.4.7 Icinga2. Cluster configuration	114
1.4.8 Keepalived + HAProxy + Nginx	122
1.4.9 Linux Desktop Programs	128

1.4.10 MySQL	128
1.4.11 Nagios + Git + Jenkins	130
1.4.12 Old Repos	134
1.4.13 OWA or Office 360 mail & Linux	136
1.4.14 PostgresSQL + Pacemaker	141
1.5 Java key store	143

Artem Kosenko's Home



Artem Kosenko

E-mail: Artem_Kosenko@epam.com
Skype: artem.kosenko.87

Recently Updated

-  [Puppet](#)
Sep 14, 2017 • updated by Artem Kosenko • [view change](#)
-  [Puppet. Cert & Curl](#)
Sep 14, 2017 • updated by Artem Kosenko • [view change](#)
-  [PCB. File](#)
Sep 14, 2017 • updated by Artem Kosenko • [view change](#)
-  [PCB. Packages](#)
Sep 14, 2017 • updated by Artem Kosenko • [view change](#)
-  [PCB. Iptables](#)
Sep 14, 2017 • updated by Artem Kosenko • [view change](#)
-  [Puppet. Hiera](#)
Sep 14, 2017 • created by Artem Kosenko
-  [99. Other](#)
Sep 07, 2017 • updated by Artem Kosenko • [view change](#)
-  [Ansible](#)
Sep 06, 2017 • updated by Artem Kosenko • [view change](#)
-  [PCB. php.ini](#)
Aug 31, 2017 • updated by Artem Kosenko • [view change](#)
-  [Icinga2. Cluster configuration.](#)
Aug 31, 2017 • created by Artem Kosenko
-  [Arch Linux](#)
Aug 04, 2017 • updated by Artem Kosenko • [view change](#)
-  [BASH SCRIPTS](#)
Aug 03, 2017 • created by Artem Kosenko
-  [Bareos. Bconsole commands](#)
Jul 26, 2017 • updated by Artem Kosenko • [view change](#)
-  [MySQL](#)
Jul 25, 2017 • updated by Artem Kosenko • [view change](#)
-  [Bareos](#)
Jul 24, 2017 • created by Artem Kosenko

Linux Basics

01. Vagrant & VirtualBox

Vagrant - командный интерфейс для работы с провайдерами виртуализации VMWare, AWS, LXC and libvirt.

Все тесты проводились на ноуте

OS: Arch Linux x86_64

Kernel Release: 4.4.35-1-lts

VirtualBox

Установка

```
| $ yaourt -S virtualbox
```

Модули ядра

для запуска VirtualBox надо подгрузить модули ядра. Яставил "lts-headers" под свое ядро, ставим в соответствии со своим ядром

```
| $ yaourt -S linux-lts-headers
```

добавляем в автостарт

```
$ cat /etc/modprobe.d/virtualbox  
vboxdrv vboxnetflt vboxnetadp
```

подгружаем ручками

```
$ modprobe vboxdrv  
$ modprobe vboxnetflt  
$ modprobe vboxnetadp
```

роверяю что модули загружены

```
$ lsmod | egrep 'vboxdrv|vboxnetflt|vboxnetadp'  
vboxnetflt 28672 0  
vboxnetadp 28672 0  
vboxdrv 385024 3 vboxnetadp,vboxnetflt,vboxpci
```

Vagrant

Базовые понятия

Vagrant Box - базовый образ для быстрого клонирования виртуальных машин (некая базовая болванка заготовка). На основе бокса вы будете создавать виртуальные машины. Сами боксы никаких не изменяются. Размер боксов варьируется от 200MB до 2,5GB.

Vagrantfile - это файл конфигурации виртуальной машины, расположен в директории проекта. Описывает производительность виртуальной машины, форвардинг портов, установку приложений, прочее.
Если вы работает в команде или хотите чтобы другие пользователи могли развернуть ваше окружение - держите этот файл в GIT репозитории.

Provider - система для виртуализации (VirtualBox, VMWare, AWS). По умолчанию используется VirtualBox.

Установка

```
| $ yaourt -S vagrant
```

Подготовка к установке

скачать "бокс" Ubuntu 16.04.1 LTS

```
| $ vagrant box add ubuntu/trusty64  
| $ ls ~/.vagrant.d/boxes/  
|   ubuntu-VAGRANTSLASH-trusty64
```

Просмотреть лист уже скачанных боксов

```
| $ vagrant box list  
|   ubuntu/trusty64 (virtualbox, 20161122.0.0)
```

Установка и подключение к VM

готовим папку для VMки

```
| $ mkdir -p ~/vagrant/ubuntu-test  
| $ cd ~/vagrant/ubuntu-test  
| $ vagrant init ubuntu/trusty64
```

появится файл Vagrantfile. Правим его если надо что-то поменять

```
| $ vim Vagrantfile
```

запускаем саму установку. при этом нужно находиться в папке с Vagrantfile. Самаже VMка попадет в каталог ~/VirtualBox VMs/ (дефолтный каталог VirtualBox'a. Можно изменить при необходимости)

```
| $ vagrant up
```

Настройка ~/.ssh/config

```
| $ vagrant ssh-config  
  
Host default  
  HostName 127.0.0.1  
  User vagrant  
  Port 2222  
  UserKnownHostsFile /dev/null  
  StrictHostKeyChecking no  
  PasswordAuthentication no  
  IdentityFile /home/lime/vagrant/ubuntu-test/.vagrant/machines/default/virtualbox/private_key  
  IdentitiesOnly yes  
  LogLevel FATAL
```

можно сразу дописать себе в конфиг

```
| vagrant ssh-config >> ~/.ssh/config
```

Работа к VM через Vagrant

Просмотр ID маинки

```
| $ vagrant global-status
```

```
id      name      provider      state      directory  
-----  
08bc45a  default  virtualbox  running  /home/<username>/vagrant/ubuntu-test
```

Если VM удалялась не через Vagrant, она может остаться в global-status. Лечится это так:

```
| $ vagrant global-status --prune
```

Для подключения к VM из каталога с Vagrantfile

```
| $ vagrant ssh
```

Подключение по ID из любого каталога

```
| $ vagrant ssh 08bc45a
```

Виртуалки можно ставить на паузу

```
| $ vagrant suspend 08bc45a  
$ vagrant global-status
```

```
id      name      provider      state      directory  
-----  
08bc45a  default  virtualbox  saved   /home/<username>/vagrant/ubuntu-test
```

Снять с паузы

```
| $ vagrant resume 08bc45a  
$ vagrant global-status
```

```
id      name      provider      state      directory  
-----  
08bc45a  default  virtualbox  running  /home/<username>/vagrant/ubuntu-test
```

Выключить / Включить VM

```
| $ vagrant halt 08bc45a  
$ vagrant up 08bc45a
```

Удалить VM

```
| $ vagrant destroy 08bc45a
```

Box export

Можно создать из виртуалки бокс, сказав в каталоге с Vagrantfile:

```
| $ vagrant package 08bc45a
```

файл появится в каталоге, из которого вызвана команда

в итоге получу файл ~350Mb (если хлама ненаставил внутри). Это уже сшитый файл, его можно импортировать в Vagrant и использовать для последующей развертки похожих машинок.

Box import

Импортируем box, просмотрим его и удалим

```
| $ vagrant box add myubuntu package.box  
| $ vagrant box list  
| myubuntu      (virtualbox, 0)  
| ubuntu/trusty64 (virtualbox, 20161122.0.0)  
|  
| $ vagrant box remove myubuntu
```

Provision, запуск shell скрипта после старта VM

Подготовить окружение виртуальной машины (provision, обеспечение) под наш проект мы можем несколькими способами. Один из них, это запуск shell скрипта, в котором прописываются все действия необходимые для настройки окружения. Давайте установим mc (здесь пример установки Apache). Создайте в корне проекта файл bootstrap.sh:

```
| $ cat bootstrap.sh  
| #!/usr/bin/env bash  
| apt-get update  
| apt-get install mc -y
```

И пропишите в Vagrantfile что необходимо запустить этот скрипт после старта виртуальной машины:

```
| $ vim Vagrantfile  
| ...  
| config.vm.provision :shell, :path => "bootstrap.sh"  
| ...
```

Перезагрузите конфиг с обновлением обеспечения:

```
| $ vagrant reload --provision
```

Шпаргалка по Vagrant

VAGRANT_HOME - , VagrantBox'

По умолчанию текущий каталог проекта (в котором находится Vagrantfile) уже синхронизируется с директорией /vagrant

VMка попадет в каталог ~/VirtualBox VMs/ (дeфолтный каталог VirtualBox'a. Можно изменить при необходимости)

~/.vagrant.d/boxes/ - каталог с боксами

Command	Description
vagrant global-status	ID,
vagrant global-status --prune	
vagrant up	VM
vagrant ssh	SSH vagrant
vagrant suspend	
vagrant halt	
vagrant reload	(provision)
vagrant reload --provision	c provision
vagrant destroy	
vagrant ssh-config	сгенерировать блок для ~/.ssh/config
vagrant ssh-config >> ~/.ssh/config	сразу вписать нужный блок в ~/.ssh/config
vagrant box list	c ""
vagrant package	[package.box]
vagrant box add [box_name] [package.box]	package.box [box_name]
vagrant box remove [box_name]	[box_name]
vagrant box add ubuntu/trusty6	box Ubuntu [ubuntu/trusty64]
vagrant init ubuntu/trusty64	VM [ubuntu/trusty64]
vagrant init centos/7	CentOS7 (,)

Vagrantfile

после правки файла, надо перезагрузить VMку:

```
| vagrant reload 08bc45a
```

line	description
config.vm.network "private_network", ip: "192.168.33.10"	static ip
config.vm.box = "lucid10x64"	
config.vm.network :forwarded_port, host: 8080, guest: 80	(8080 80)
config.vm.synced_folder "/home/stas/www/vagrant/src", "/var/www"	2 (,)
config.vm.provision :shell, :path => "bootstrap.sh"	"bootstrap.sh", vagrant reload --provision
config.vm.hostname = "Ubuntu-16-04"	
config.vm.define "Ubuntu-16-04" do v end	"vagrant global-list"
config.vm.provider "virtualbox" do v v.name = "Ubuntu-16-04" v.memory = "512" v.cpus = 1 end	"VirtualBox" RAM Mb CPU -

Альтернативные провайдеры виртуализации

В качестве примера альтернативного провайдера рассмотрим vagrant-lxc. Он позволяет делать все, что было описано выше, используя LXC вместо VirtualBox. Если в двух словах:

```
# ставим плагин
vagrant plugin install vagrant-lxc

# создаем и запускаем виртуалку
vagrant init frehm/precise64-lxc
vagrant up --provider=lxc
```

AWS

```
vagrant up --provider=vmware_fusion
```

Ссылки

<https://docs.vagrantup.com/v2/> — документация по Vagrant;
<https://atlas.hashicorp.com/boxes/search> — каталог готовых боксов;
<http://stackoverflow.com/a/18104177/1565238> — автозапуск виртуалки;
<http://serverfault.com/a/358040> — оверхед, правда, в случае VMWare;
<https://groups.google.com/forum/#!forum/vagrant-up> — список рассылки;

02. Linux Distributions

Differences

Family	OS	Package Manager	Network Configuration	Autostart	Apache
RHEL	Red Hat CentOS Fedora	RPM YUM yum search vim yum info vim yum install vim -y yum remove vim yum update vim yum update	/etc/sysconfig/network-scripts/ifcfg-*service restart network	chkconfig --list chkconfig --list httpd chkconfig --add httpd chkconfig --del httpd chkconfig --level 23 httpd on chkconfig --level 01456 httpd off runlevel	HTTPD /etc/httpd/conf/httpd.conf /etc/httpd/conf.d/DocumentRoot "/var/www/html"
DEBIAN	Debian Ubundu	DPKG APT apt search vim apt install vim -y apt remove vim apt update vim apt update apt upgrade	/etc/network/interfaces service networking restart	apt install rcconf dialog rcconf update-rc.d apache2 defaults update-rc.d apache2 start 20 2 3 4 5 . apache2 20 0 1 6 . update-rc.d -f apache2 remove chmod +x /etc/init.d/apache2	APACHE2 /etc/apache2/apache2.conf /etc/apache2/sites-available/*.conf /etc/apache2/sites-enabled/*.conf (links) DocumentRoot /var/www/html
SUSE	openSUSE SUSE Linux Enterprise Server (SLES)	RPM YaST ZYPPER zypper se vim zypper if vim zypper in vim zypper up vim zypper dup zypper lr zypper ar zypper rr	/etc/sysconfig/network/ifcfg-*rcnetwork restart	chkconfig --list chkconfig --list httpd chkconfig --add httpd chkconfig --del httpd chkconfig --level 23 httpd on chkconfig --level 01456 httpd off runlevel	APACHE2/etc/apache2/httpd.conf /etc/apache2/default-server.conf ... /etc/apache2/vhosts.d/*.conf /etc/apache2/conf.d/*.conf DocumentRoot "/srv/www/htdocs" UserDir public_html

Directory structure

folder	description
/	корневой раздел
/bin	основной состав команд (cd, ls, cp, etc...)
/boot	раздел загрузки. тут хранятся ядра и конфиги загрузчика (grub, lilo, etc...)

/dev	файлы подключенных устройств (интерфейсы взаимодействия с ними)
/etc	файлы конфигурации ОС и Приложений
/home	папка с домашними каталогами пользователей
/lost+found	"потерянные файлы" появляются при сбоях удаления файлов. В журналируемых файловых системах обычно исправляется прогоном fsck
/lib	либы для приложений в /bin и /sbin
/media	каталог для автоматического монтирования схемных носителей
/mnt	каталог для ручного монтирования
/opt	опциональный софт, применительно к больших размеров (либра-офис например)
/proc	Сюда примонтирана "procfs", виртуальная "ФС", с наличием множественной информации, которую можно получить. Модули ядра загружены, это будет файл - /proc/modules сведения о процессоре - /proc/cpuinfo
/root	хома рута. отдельно, чтоб не отваливаться если что-то произойдет с отдельным разделом /home
/sbin	каталог системных приложений для различных настроек. ifup, iptables-save, lvdisplay - живут тут
/srv	обычно пустой каталог
/sys	начиная с ядра v_2.6 и в нее примонтируется "sysfs", с информацией о ядре, устройствах и драйверах.
/sys/block	директории блочных устройств, которые имеются в системе в реальное время.
/sys/bus	перечень шин ядра: eisa, pci и тд. и тп.
/sys/class	перечень группированных устройств по классификации: printer, scsi-devices и тд. и тп.
/tmp	временные файлы (777)
/usr	место установленных программ, маны, бинарники, иногда логи и конфиги программы тоже можно найти туту в папках вроде /usr/local/PROGRAM/etc или /usr/local/PROGRAM/var/log
/usr/bin	бинарники приложений, доступные для всех пользователей
/usr/games	игрушки
/usr/include	Заголовочные файлы C++
/usr/lib	Системные библиотеки для приложений в /usr
/usr/local	в идеале каталог /usr смонтирован по сети, а каталог /usr/local находится на локальном диске и не доступен по сети для монтирования на других серверах
/usr/sbin	Системные приложения дополнительного плана
/usr/share	Общая информация об установленных приложениях. Ярлычки, темы, иконки и т.п.
/usr/share/icons	системные иконки
/usr/share/doc	маны установленного ПО
/usr/src	Исходные коды, например ядра.
/var	Постоянно меняющаяся информация, log/cache - файлы.
/var/cache	Кэш-файлы приложений
/var/games	Местоположение достигнутых игровых рекордов (сейвы)
/var/lib	Информация изменяемая приложениями в результате деятельности: базы данных, метаданные и тд.
/var/lock	Нхождение lock-файлов, указывающие на занятые ресурсы

/var/log	логи
/var/spool	Запланированные задания в очереди: печать, отправка email, cron и пр...
/var/www	Серверная директория (Apache), для размещения веб/страниц.
<swap>	Раздел виртуальной памяти для ускорения обменных процессов с оперативной памятью.

File types

Типы файлов в Linux

Типы файлов		Назначение
Обычные файлы	—	Хранение символьных и двоичных данных
Каталоги	d	Организация доступа к файлам
Символьные ссылки	l	Предоставление доступа к файлам, расположенных на любых носителях
Блочные устройства	b	Предоставление интерфейса для взаимодействия с аппаратным обеспечением компьютера
Символьные устройства	c	
Каналы	p	Организация взаимодействия процессов в операционной системе
Сокеты	s	

<http://younglinux.info>

Формат файла можно проверить вот так:

```
$ file /dev/null
/dev/null: character special (1/3)

$ file /etc/passwd
/etc/passwd: ASCII text
```

а так же по первому символу перед правами в выводе команды ls -la

- (-) -rwxr-xr-x. Обычный файл
- (d) drwxr-xr-x Каталог
- (l) lrwxrwxrwx. Символическая ссылка
- (b) brw-rw---- Блочные файлы
- (c) crw----- Символьные файлы
- (p) prw----- 1 Туннели и именованные туннели
- (s) srwxr-xr-x Файлы сокетов

Обычные файлы

(-) Обычные файлы используются для хранения информации:

- Текстовые файлы
- Исполняемые файлы
- Файлы изображений
- Файлы архивов
- Файлы библиотек программ
- И другие подобные типы

Каталоги

(d) директории

Специальные файлы

для устройств и туннелей.

(!) Символические ссылки - это файлы, которые указывают на другие файлы в системе по их имени.

(b) Блочные файлы - это файлы устройств, которые обеспечивают буферизованный доступ к аппаратным компонентам. При записи данных на жесткий диск или на флешку нет смысла записывать данные сразу же после их поступления. Так мы будем только понапрасну расходовать ресурса устройства и энергии. Можно подождать пока наберется достаточно большое количество данных а потом записать их за один раз. Эти данные и собираются в буфере. С помощью таких файлов, файловая система и другие утилиты могут обращаться к драйверам аппаратных устройств. Такие файлы могут передавать большой блок данных за небольшой один раз.

(c) Символьные файлы обеспечивают не буферизованный доступ к аппаратным компонентам и ядру. Поскольку у них нет буфера, они позволяют передавать только по одному символу за один раз. А в остальном, это такие же файлы устройств, как и блочные файлы.

(p) Туннели и именованные туннели - это файлы, позволяющие настроить связь между двумя процессами перенаправив вывод одного процесса на вход другого. Именованные туннели используются для связи между двумя процессами и работают так же как и обычные туннели.

(s) Файлы сокетов - это файлы, обеспечивающие прямую связь между процессами, они могут передавать информацию между процессами, запущенными в разных средах или даже разных машинах. Это значит, что с помощью сокетов программы могут обмениваться данными даже по сети. По сути, сокет работает так же как туннели, но только в обе стороны.

ссылке по теме

<https://losst.ru/tipy-fajlov-v-linux>

Special Devices

В линукс есть несколько (с) символьных файлов, связанных напрямую с памятью ядра вместо физических устройств.

Device	Description	Example
--------	-------------	---------

/dev/full	<p>"полное устройство". Запись в него ненулевого количества байт происходит с ошибкой «недостаточно места», независимо от объема «записанной» информации. Чтение из /dev/full эквивалентно считыванию запрошенного количества нулевых байтов, как для /dev/zero.</p> <p>Запись в /dev/full завершается успешно только для нулевого количества байт. Это устройство используется для добавления проверок того, что программа ничего не выводит. Если программа сама не обработает ошибку вывода, то она завершится с ошибкой при первом нарушении требования ничего не выводить.</p>	<p>создаётся с помощью утилиты mknod следующим образом: mknod FILE c 1 7</p> <p>Проверить как ведет себя программа при обращении к "полному" устройству \$ cp test-file /dev/full cp: writing /dev/full": No space left on device</p>
/dev/null	<p>"пустое устройство". Запись в него происходит успешно, независимо от объема «записанной» информации. Чтение из /dev/null эквивалентно считыванию конца файла (EOF)</p>	<p>создаётся с помощью утилиты mknod следующим образом: mknod FILE c 1 3</p> <p>Вывод потока стандартного вывода (STDOUT) и потока ошибок (STDERR) в /dev/null: /dev/null: сделать что-то > /dev/null 2>&1</p> <p>Вывод потока ошибок (STDERR) в /dev/null: сделать что-то 2>/dev/null</p>
/dev/random	генератор случайных чисел. выдает произвольный "шум из пула" если пул опустел, то ничего не выдаст.	<p>генерация случайного числа</p> <pre>dd if=/dev/random count=1 2>/dev/null od -t u1 awk '{print \$2}' head -1</pre>
/dev/urandom	генератор псевдо случайных чисел. выдаст столько бит шума, сколько будет запрошено.	<p>Заполняем раздел случайными данными для удаления остаточной информации:</p> <pre>dd if=/dev/urandom of=/dev/sdb3</pre>
/dev/zero	<p>источник нулевых байтов (ASCII NUL, 0x00). При чтении этого файла никогда не достигается его конец.</p> <p>Любые данные, записанные в /dev/zero, будут игнорированы, а сама запись завершается успешно — точно так же, как и при записи в /dev/null (хотя последнее намного чаще используется как «чёрная дыра», чем /dev/zero)</p>	<p>mknod :</p> <p>mknod FILE c 1 5</p> <p>чаще всего используется для создания файлов заданного размера</p> <pre>dd if=/dev/zero of=/swapfile bs=1M count=2048</pre>

https://www.suse.com/documentation/sled11/book_sle_admin/data/sec_basicnet_manconf.html

RHEL

Network Interfaces https://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-networkscripts.html

Interface Configuration Files https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-networkscripts-interfaces.html

Interface Control Scripts https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-networkscripts-interfaces.html

Users & Groups. Permissions. Umask

Пользователи и группы в линукс нужны для простого управления правами.

Группы применяются для расширенного управления правами.

Управление пользователями

Информация о пользователях находится в файле /etc/passwd. Чтобы узнать список всех пользователей в системе:

```
| $ cat /etc/passwd
```

На каждый аккаунт приходится по одной строке, имеющей следующий формат:

```
| account:password:UID:GID:GECOS:directory:shell
```

- account — имя пользователя
- password — пароль пользователя
- UID — идентификационный номер пользователя
- GID — идентификационный номер основной группы пользователя
- GECOS — необязательное поле, используемое для указания дополнительной информации о пользователе (например, полное имя пользователя)
- directory — домашний каталог (\$HOME) пользователя
- shell — командный интерпретатор пользователя (обычно /bin/sh)

пароли хранятся в зашифрованном виде в файле /etc/shadow

```
| $ ls -l /etc/shadow
| -rw----- 1 root root 986 6 08:25 /etc/shadow
```

Добавление пользователя

Команда "useradd"

используется для добавления пользователей

```
| # useradd -m -g [ ] -G [ ] -s [ ] [ ]
```

- -m — создаёт домашний каталог пользователя, вида /home/[имя пользователя]; в пределах которого, пользователь, не имеющий прав доступа root, может создавать и удалять файлы, устанавливать программы, и т. д.
- -g — определяет имя или номер основной группы пользователя; группа должна существовать; номер группы должен относиться к уже существующей группе; если параметр не указан, пользователю будет присвоена группа в соответствии с переменной USERGROUPS_ENAB, находящейся в /etc/login.defs.
- -G — определяет список дополнительных групп, в которые входит пользователь; каждая группа отделяется от другой запятой без пробелов; по умолчанию пользователь принадлежит только основной группе.
- -s — определяет командную оболочку пользователя; сценарии запуска Arch Linux используют Bash; после завершения запуска системы, командная оболочка будет той, что указана в данном параметре; при выборе отличного от Bash интерпретатора, убедитесь, что он установлен в систему.

Стандартный пример - добавление нового пользователя archie, с выбором баша в качестве оболочки и включением его в группы wheel и audio:

```
| # useradd -m -g users -G wheel,audio -s /bin/bash archie
```

Команда "chfn"

задать или изменить персональную информацию о пользователе

```
| # chfn [ ]  
Name [ ]:  
Office [ ]:  
Office Phone [ ]:  
Home Phone [ ]:
```

Удаление пользователя

```
| # userdel -r [ ]
```

Опция `-r` также удаляет домашнюю директорию и почту пользователя.

Управление группами

`/etc/group` - файл, хранящий информацию о группах пользователей.

Чтобы отобразить группы, в которые включен пользователь, можно использовать команду `groups`:

```
| $ groups [ ]
```

Если имя пользователя не указано, команда отобразит группы текущего пользователя.

Используя команду `id` можно получить дополнительную информацию, такую как ID пользователя и его групп (UID и GID):

```
| $ id [ ]
```

Отобразить список всех групп:

```
| $ cat /etc/group
```

Добавить новую группу командой `groupadd`:

```
| # groupadd [ ]
```

Добавить пользователя в группу, команда `gpasswd`:

```
| # gpasswd -a [ ] [ ]
```

Удалить группу:

```
| # groupdel [имя группы]
```

Убрать пользователя из группы:

```
| # gpasswd -d [ ] [ ]
```

Установить пароль на группу

```
| # gpasswd [ ]
```

Пользователю необходимо перезайти в систему, чтобы изменения вступили в силу.

Наиболее часто используемые группы

- audio
- floppy
- lp
- network
- optical
- power
- storage
- video
- wheel

статья по теме групп:

[https://wiki.archlinux.org/index.php/Users_and_groups_\(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9\)](https://wiki.archlinux.org/index.php/Users_and_groups_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9))

статья по теме паролей

http://citforum.ru/operating_systems/unix/kravchuk/3.shtml

Команда "who"

показывает всех пользователей входивших в систему и время из последнего входа

```
$ who
lime      tty7          2016-12-07 21:39 (:0)
```

Команда "ls -l"

```
$ ls -l
total 32
-rw-r--r-- 1 lime users 19663  1 09:13 config
-rw-r--r-- 1 lime users  4476 17 21:27 i3blocks.conf
drwxr-xr-x 2 lime users  4096 11 14:42 scrypts
```

Команда "stat"

пользователь

```
$ stat -c %U config
lime
```

группа

```
$ stat -c %G config
users
```

права доступа

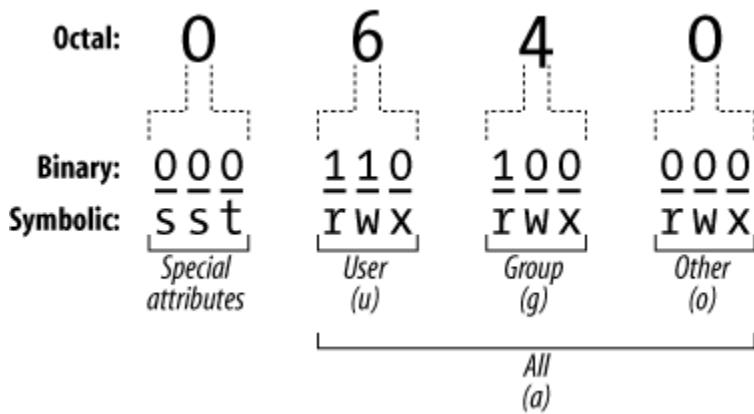
```
$ stat -c %A config
-rw-r--r--
```

Команда "find"

Узнать список файлов, которыми владеет тот или иной пользователь и группа,

```
# find / -group [group]
# find / -user [user]
```

Permissions



BIN	OCT	MASK	comment
001	1	--x	только запуск
010	2	-w-	только запись
011	3	-wx	запись и запуск
100	4	r--	только чтанеи
101	5	r-x	чтение и запуск
110	6	r w-	чтение и запись
111	7	rwx	чтение, запись, запуск

Назначение прав

задание прав и владельца

```
chmod 644 file          #
chown -R user:group [dir] #      (-R )
```

изменить только указанную часть прав, остальное оставить как есть

```
chmod u+r file          #
chmod g-rw file          #
chmod o+rwx file         #      " "
```

```
chmod u+rwx,g+rwx file      #      " "      "
```

Umask

маска прав

File Permissions and Ownership (cont'd)

- Set default mode and group
 - umask
 - /etc/profile
 - newgrp (change default group -l)

Umask	Created Files	Created Directories
000	666 (rw-rw-rw-)	777 (rwxrwxrwx)
002	664 (rw-rw-r--)	775 (rwxrwxr-x)
022	644 (rw-r--r--)	755 (rwxr-xr-x)
027	640 (rw-r----)	750 (rwxr-x--)
077	600 (rw-----)	700 (rwx-----)
277	400 (r-----)	500 (r-x-----)

для файлов считается 666 - UMASK

для каталогов считается 777 - UMASK

дефолтно задается в /etc/profile

разово задается командой umask

```
umask 002
touch file
makedir dir
ls -l

-rw-rw-r-- (664) file
drwxrwxr-x (775) dir
```

SUID, GID, Sticky bit

setuid и setgid (сокращения от англ. *set user ID upon execution* — «установка ID пользователя во время выполнения» и англ. *set group ID upon execution* — «установка ID группы во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами владельца или группы исполняемого файла.

setuid

запускает файл с правами пользователя, которому он принадлежит

```
chmod 4xxx      #  xxx -
chmod u+s      #  SUID
chmod u-s      #  SUID

ls -l file
---S-----      #  (S)      ( - )
---S-----      #  (s)      (x)
```

setgid

запускает файл с правами группы, которой он принадлежит

```
chmod 2xxx      #  xxx -
chmod g+s       #  SUID
chmod g-s       #  SUID

ls -l file
----S--      #  (S)      (-)
----s--      #  (s)      (x)
```

Sticky bit

применяется для защиты файлов в папке. Если на общую папку установлен Sticky bit, то пользователь сможет удалять только файлы, владельцем которых он является (пример /tmp)

Sticky bit на исполняемых файлах нужен для кеширования их и более быстром последующем запуске (применялся на старых компах, сейчасредко используется)

```
chmod 1xxx file      #  xxx -
chmod o+t file       #  Stiki bit
chmod o-t file       #  Stiki bit

ls -l file
-----T      #  (T)      (-)
-----t      #  (t)      (x)
```

BIN		MASK
0xxx	ugo-st	-----
1xxx	o+t	-----T
2xxx	g+s	----S--
3xxx	go+st	----S--T
4xxx	u+t	---S----
5xxx	uo+st	---S----T
6xxx	ug+s	---S--S--
7xxx	ugo+st	---S--S--T

просмотр аривиатуры и цифрового представления прав

```
ls -l file
---S--S--T. 1 vagrant vagrant 0 Dec  8 13:48 file

stat -c %A file
---S--S--T

stat -c %a file
7000
```

Lightdm, вход без пароля и автовход (для домашнего PC)

[https://wiki.archlinux.org/index.php/LightDM_\(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9\)](https://wiki.archlinux.org/index.php/LightDM_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9))

http://help.ubuntu.ru/wiki/%D0%BF%D1%83%D1%81%D1%82%D0%BE%D0%B9_%D0%BF%D0%B0%D1%80%D0%BE%D0%BB%D1%8C

SUDO

sudo - выполнение команд от пользователя root

command	description
sudo -l	список доступных команд для выполнения через sudo
sudo -ll	~"~ в расширенном виде
visudo	открыть файл настрокек sudo

примеры из конфига visudo

Разрешить пользователям, входящим в группу "admin", выполнять команды с правами любого пользователя (с запросом пароля):

```
%admin ALL=(ALL) ALL
```

Позволить пользователю «user1» выполнять команды с правами «user2» или «user3», не запрашивая пароль:

```
user1 ALL=(user2, user3)NOPASSWD: ALL
```

Разрешить пользователю «backup» выполнять команду /usr/bin/rsync без запроса пароля:

```
backup ALL=NOPASSWD: /usr/bin/rsync
```

примеры испоользования

выдаст ошибку, т.к. от sudo выполнится только "cat", а перенаправление будет выполняться от "sh/bash" с правами обычного пользователя

```
sudo cat sources.list > /etc/apt/sources.list
```

для корректной отработки команды используем конеер

```
cat sources.list | sudo tee /etc/apt/sources.list
```

или так

```
sudo sh -c 'cat sources.list > /etc/apt/sources.list'
```

Package Managers

APT

apt-install

apt search

apt purge

apt remove

apt update

apt upgrade

список реп тут

/etc/apt/sources.list

deb http://archive.ubuntu.com/ubuntu xenial main restricted

deb	
http://archive.ubuntu.com/ubuntu	ссылка на репу
xenial	название дистрибутива
main restricted	подключаемые папки из каталога выбранного дистрибутива

можно повыбирать тут:

<http://archive.ubuntu.com/ubuntu/dists/>

Добавить репу из консоли

\$ sudo apt-add-repository "deb <http://ppa.launchpad.net/shutter/ppa/ubuntu> wily main"

Добавить ключ

\$ curl -L http://debian.datastax.com/debian/repo_key | sudo apt-key add

Удалить репозиторий

\$ sudo apt-add-repository --repository "deb <http://ppa.launchpad.net/shutter/ppa/ubuntu> wily main"

Добавить PPA репу

```
$ sudo add-apt-repository ppa:deluge-team/ppa && sudo apt-get update
```

Добавление ключа

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 12345678
```

Pacman & Yaourt

[https://wiki.archlinux.org/index.php/Pacman_\(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9\)](https://wiki.archlinux.org/index.php/Pacman_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9))

Команда	Описание
pacman -S _1 _2 ...	установка одного пакета или списка пакетов (включая зависимости)
pacman -R _	удаление пакета без удаления установленных зависимостей
pacman -Rs _	удаление пакета со всеми зависимостями, не используемыми другими установленными пакетами
pacman -Rsc _	удаление пакета, его зависимостей и всех пакетов, зависящих от целевого пакета <i>Эта операция рекурсивна и должна использоваться с осторожностью, так как появляется риск удалить много потенциально необходимых пакетов</i>
pacman -Rdd _	удаление пакета, который требуется другому пакету, без удаления зависимого пакета:
pacman -Rn _	pacman создает резервные копии конфигурационных файлов удаляемых приложений и добавляет к ним расширение .pacsave. Если вы хотите удалить эти файлы, используйте ключ -p
pacman -Syu	Обновление всех пакетов
yaourt -Syua	Обновление всех пакетов и пакетов из AUR репозитория
pacman -Ss 1 2 ...	искать пакеты в базе данных как по названиям, так и по описаниям
pacman -Qs 1 2 ...	поиск среди установленных пакетов
pacman -Si _	отображение подробной информации об указанном пакете:
pacman -Qi _	отображение подробной информации об указанном пакете (для установленных пакетов)
pacman -Qii _	Использование сразу двух флагов -i позволит просмотреть список резервных копий файлов и список измененных файлов
pacman -Ql _	получение списка файлов установленного пакета

pacman -Qk _	проверить наличие файлов, установленных пакетом Использование сразу двух флагов k выполнит более тщательную проверку
pacman -Qo _/_/_	узнать, какому пакету принадлежит нужный файл
pacman -Qdt	получение списка пакетов, которые больше не требуются как зависимости (пакеты, которые могут быть безопасно удалены или "осиротевшие" пакеты)
pacman -Qet	получение списка пакетов, которые были установлены явно и от которых не зависят другие пакеты
pactree _	отображения зависимостей в виде дерева
whoneeds _ pactree -r _	получение списка пакетов, которые зависят от конкретного установленного пакета, можно использовать утилиту whoneeds из состава пакета pkgtools:

RPM

<http://centos.name/?page=tipsandtricks/YumAndRpm>

Полезное

Yum (Yellowdog Updater, Modified)

yum [key] [command] [package]

command	description
yum list	список названий пакетов из репозиторий
yum list available	список всех доступных пакетов
yum list installed	список всех установленных пакетов
yum list installed httpd	установлен ли указанный пакет
yum list all	список установленных и доступных пакетов
yum list kernel	список пакетов, относящихся к ядру
yum info httpd	отображение информации о пакете
yum deplist httpd	список зависимостей и необходимых пакетов
yum provides "*bin/top"	найти пакет, который содержит файл
yum search httpd	поиск пакета по имени и описанию
yum updateinfo list security	получить информацию о доступных обновлениях безопасности
yum grouplist	вывести список групп
yum groupinfo "Basic Web Server"	вывести описание и содержимое группы
yum groupinstall "Basic Web Server"	установка группы пакетов «Basic Web Server»
yum groupremove "Basic Web Server"	удаление группы
yum check-update	Проверка на доступные обновления
yum repolist	список подключенных репозиториев
yum repoinfo epel	информация об определенном репозитории
yum repo-pkgs epel list	информация о пакетах в указанном репозитории
yum repo-pkgs repename install	установить все пакеты из репозитория
yum repo-pkgs repename remove	удалить пакеты установленные из репозитория
yum makecache	создать кэш
yum check yum check dependencies	проверить локальную базу rpm (поддерживаются параметры dependencies, duplicates, obsoletes, provides)
yum history list	просмотр yum истории (вывод списка транзакций)
yum history info 9	просмотр информации определенной транзакции (установленные пакеты, установленные зависимости)
yum history undo 9	отмена транзакции
yum history redo 9	повторить
cat /var/log/yum.log	дополнительно можно просмотреть лог
yum clean packages	удалить пакеты сохраненные в кэше
yum clean all	удалить все пакеты и метаданные

<code>yum install httpd</code>	установить пакет
<code>yum remove httpd</code>	удаление пакета
<code>yum update httpd</code>	обновить пакет
<code>yum update</code>	обновить все пакеты
<code>yum update-to</code>	обновить до определенной версии
<code>yum localinstall httpd.rpm yum install httpd.rpm</code>	установить из локальной директории (поиск/установка зависимостей будут произведены из подключенных репозиториев)
<code>yum localinstall http://server/repo/httpd.rpm</code>	установить с http
<code>yum downgrade</code>	откатиться к предыдущей версии пакета
<code>yum reinstall httpd</code>	переустановка пакета (восстановление удаленных файлов)
<code>yum autoremove</code>	удаление ненужных более пакетов
<code>createrepo</code>	создание локальных репозиториев (createrepo ставится отдельно)
<code>yum-cron</code>	установка обновлений по расписанию (yum-cron устанавливается отдельно)

Опции Yum

key	description
<code>-y yum update -y</code>	ответить «yes» при запросе
<code>--assumeno</code>	ответить «по» при запросе
<code>--nopugins</code>	использовать Yum без плагинов
<code>--disableplugin=fastestmirror</code>	или отключить определенный плагин
<code>yum --enableplugin=ps</code>	включить плагины, которые установлены, но отключены
<code>yum update -y --enablerrepo=epel</code>	включить отключенный репозиторий
<code>yum update -y --disablerrepo=epel</code>	отключить репозиторий
<code>yum install httpd --downloadonly</code>	скачать пакеты, но не устанавливать (на Centos 7 x86_64 будут скачаны в '/var/cache/yum/x86_64/7/base/packages/')

Следующие команды доступны после установки пакета yum-utils

command	description
<code>find-repos-of-install httpd</code>	найти из какого репозитория установлен пакет
<code>needs-restarting</code>	найти процессы, пакеты которых обновлены и требуют рестарта
<code>repoquery --requires --resolve httpd</code>	запрос к репозиторию, узнать зависимости пакета, не устанавливая его
<code>reposync -p repo1 --repoid=updates</code>	синхронизировать yum репозиторий updates в локальную директорию repo1
<code>verifytree URL</code>	проверить локальный репозиторий на целостность
<code>yum-complete-transaction</code>	завершить транзакции
<code>yum-builddep</code>	установить необходимые зависимости для сборки RPM пакета
<code>yum-config-manager</code>	управление конфигурационными опциями и репозиториями yum

yumdb info httpd	запрос к локальной базе yum, отображение информации о пакете (использованная команда, контрольная сумма, URL с которого был установлен и другое)
yumdownloader	скачать rpm пакеты из репозитория
yumdownloader --source php	качать src.rpm пакет из репозитория (должен быть подключен соответствующий репозиторий, например в '/etc/yum.repos.d/CentOS-Sources.repo' в CentOS)

Конфигурационные файлы Yum и их расположение

file	desctiprion
/etc/yum.conf	Основной конфигурационный файл
/etc/yum/	директория, с конфигурациями (например, yum плагины)
/etc/yum.repos.d/	директория, содержащая информацию о репозиториях

Некоторые опции yum.conf

option	description
cachedir=/var/cache/yum/\$basearch/\$releasever	Директория, где yum хранит кэш и файлы базы (по умолчанию '/var/cache/yum')
keepcache=1	Определяет должен или нет Yum хранить кэш заголовков и пакетов после успешной установки. Значения: 0 или 1. (по умолчанию 1)
debuglevel=2	уровень вывода отладочных сообщений. Значения: 1-10 (по умолчанию 2)
logfile=/var/log/yum.log	лог файл (по умолчанию '/var/log/yum.log')
obsoletes=1	обновлять устаревшие пакеты
gpgcheck=1	проверка подписи пакетов. Значения: 0 или 1 (по умолчанию 1)
plugins=1	включение плагинов. Значения: 0 или 1 (по умолчанию 1)

Работа Yum через прокси сервер

option / command	descriprion
proxy="http://server:3128"	Для всех пользователей: добавить в секцию [main] в /etc/yum.conf
proxy_username=user proxy_password=pass	при необходимости указать пароль, добавить
export http_proxy="http://server:3128"	указать прокси для отдельного пользователя

Некоторые полезные плагины

plugin	description
yum-plugin-changelog	Добавляет опцию командной строки для просмотра ченжлого перед/после обновлениями
yum-plugin-fastestmirror	выбирает более быстрые репозитории из списка зеркал
yum-plugin-keys	добавляет команды keys, keys-info, keys-data, keys-remove, которые позволяют работать с ключами.
yum-plugin-versionlock	блокировать указанные пакеты от обновления, команда yum versionlock
yum-plugin-verify	добавление команд yum verify-all, verify-multilib, verify-rpm для проверки контрольных сумм пакетов

03. Linux Boot

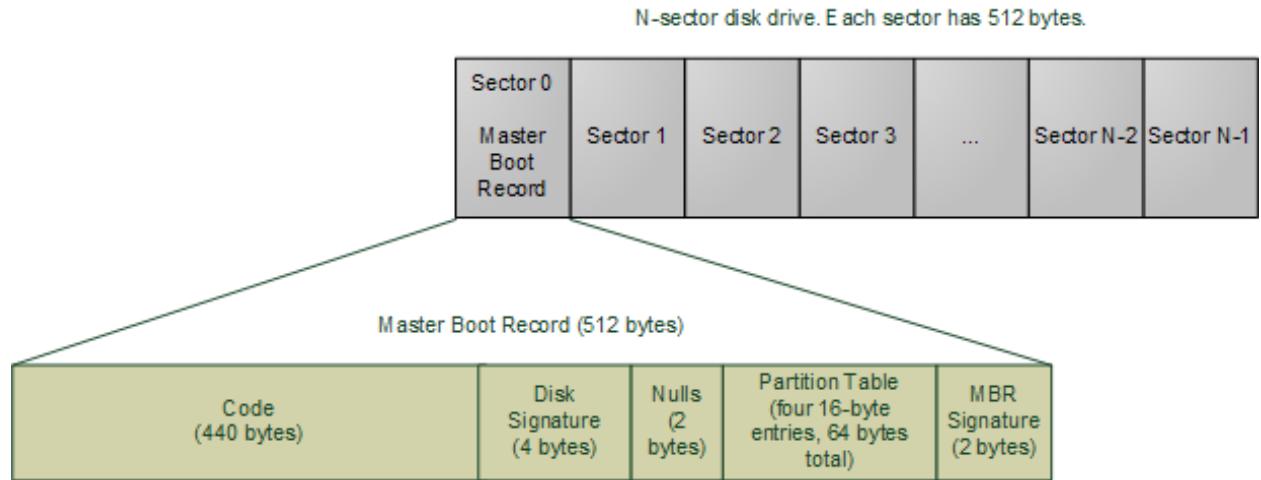


BIOS

- BIOS (Basic Input/Output System) отвечает за базовый ввод/вывод данных с устройств/на устройства.
- Делает некоторые проверки целостности устройств и Power-On Self-Test, он же «тест на адекватность себя самого»
- Ищет, загружает и выполняет программу-загрузчик ОС
- Берет загрузчик из флоппика, сидюка или жесткого диска.
- Как только загрузчик был обнаружен и загружен в память, BIOS передает управление ему.
- Короче говоря, BIOS загружает и выполняет загрузочную запись (MBR).

MBR

- MBR (Master Boot Record) - это главная загрузочная запись, хранящаяся на жестком диске
- Она размещена в 1-м секторе загрузочного диска, например /dev/hda или /dev/sda
- MBR занимает меньше, чем 512 байтов. Она состоит из трех компонентов: 1) главная загрузочная информация, «живущая» в первых 446 байтах; 2) информация о таблице разделов — в следующих 64 байтах; 3) и последние 2 байта нужны для проверки корректности mbr.
- Она содержит информацию о GRUB'e (или LILO).
- Простыми словами — MBR загружает и выполняет загрузчик GRUB.



```
dd if=/path/mbr-backup of=/dev/sda bs=446 count=1
```

Для восстановления только таблицы разделов:

```
dd if=/path/mbr-backup of=/dev/sda bs=1 skip=446 seek=466 count=66
```

MBR, но оставить таблицу разделов (может быть полезно, если вы хотите полностью переустановить операционную систему):

```
dd if=/dev/zero of=/dev/sda bs=446 count=1
```

если запароль виндовый загрузчик, то можно использовать утилиту ms-sys для его установки или воспользоваться виндовыми утилитами

<http://ms-sys.sourceforge.net/>

[https://wiki.archlinux.org/index.php/Master_Boot_Record_\(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9\)](https://wiki.archlinux.org/index.php/Master_Boot_Record_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9))

GRUB

- GRUB (Grand Unified Bootloader) - загружает и выполняет образы ядра и initrd.
- Если в системе больше чем одно ядро, граб позволяет выбрать какое из них грузить
- Параметры загрузки ядра тоже указываются тут
- GRUB понимает, что такая файловая система (LILO - нет)
- Конфигурационный файл Grub обычно лежит по пути /boot/grub/grub.conf (так же /etc/grub.conf может быть символьной ссылкой на него)
- конфигурационный файл содержит путь к ядру и образу initrd

Kernel

- Ядро монтирует файловую систему в соответствии с настройкой «root» в файле grub.conf
- Выполняет программу /sbin/init
- Поскольку init - это первый процесс, запущенный ядром Linux, поэтому она имеет идентификатор процесса (PID) №1. Можно убедиться выполнив ps -ef | grep init
- initrd - это Initial RAM Disk, он же временный диск в оперативной памяти
- initrd используется самим ядром в качестве временной корневой файловой системы, пока kernel не загрузится в реальную примонтированную файловую систему. Этот временный диск также содержит необходимые для загрузки драйверы, позволяющие получить доступ к разделам дисков и другому оборудованию

INIT

Смотрит в файл /etc/inittab для того, чтобы определить уровень выполнения (run level).

Есть следующие уровни выполнения:

- 0 - выключение

- 1 - однопользовательский режим, без поддержки сети
- 2 - многопользовательский режим, без поддержки сети
- 3 - многопользовательский режим (стандарт для серверов)
- 4 - использовался только в Slackware Linux для многопользовательского графического режима. В остальных дистрибутивах не сконфигурирован
- 5 - многопользовательский режим с сетью и иксами (стандарт для рабочих станций и букв)
- 6 - reboot

Уровень выполнения программ (Runlevel)

Исходя из настроек по умолчанию, система будет выполнять файлы в соответствии с нижеприведенными директориями.

- Выполнение уровня 0 – /etc/rc.d/rc0.d/
- Выполнение уровня 1 – /etc/rc.d/rc1.d/
- Выполнение уровня 2 – /etc/rc.d/rc2.d/
- Выполнение уровня 3 – /etc/rc.d/rc3.d/
- Выполнение уровня 4 – /etc/rc.d/rc4.d/
- Выполнение уровня 5 – /etc/rc.d/rc5.d/
- Выполнение уровня 6 – /etc/rc.d/rc6.d/

В каталоге /etc могут быть символические ссылки. Например, /etc/rc0.d залинкован на /etc/rc.d/rc0.d

В каталогах /etc/rc.d/rc*.d/ находится список программ, имя которых начинается из букв S и K.

S (startup) - используются для запуска

K (kill) - для завершения работы

Номера рядом с буквами S и K в именах программ используются для определения порядка запуска этих программ.

К примеру, **S12syslog** предназначен для запуска демона syslog, его порядковый номер **12**.

S80sendmail - для запуска демона sendmail, имеющего порядковый номер **80**. Таким образом, программа syslog будет запущена перед sendmail.

про управление уровнями загрузки деманов в различных дистрибутивах можно почитать тут: [02. Linux Distributions](#)

GRUB

Вынос конфигурации в другой файл

Так как GRUB поддерживает модульность конфигурации, можно оставить в файле **grub.cfg** только одну строку со ссылкой на другой файл, например **menu.cfg**

```
| /boot/grub/grub.cfg
| . $prefix/menu.cfg
```

и в дальнейшем вместо **grub.cfg** править только **menu.cfg**

Для автоконфигурации в этом случае можно использовать команду

```
| grub-mkconfig -o /boot/grub/menu.cfg
```

Прямая блокировка grub.cfg

Чтобы защитить файл от любых изменений, присвойте ему атрибут **immutable**

```
| chattr +i /boot/grub/grub.cfg
```

Блокировка снимается командой

```
| chattr -i /boot/grub/grub.cfg
```

Если основная конфигурация уже вынесена в другой файл, блокировку grub.cfg достаточно установить однажды и больше не снимать. Блокировка защитит файл от перезаписи скриптами установки пакетов.

Чтоб избежать конфликтов при обновлении пакета, желательно добавить путь к файлу в исключении.

Синтаксис файла конфигурации GRUB

Пример минимальной работающей конфигурации

Здесь только один пункт меню, загрузчик в корневом разделе, который передаётся ядру меткой Arch_root

```
/boot/grub/grub.cfg

set timeout=5
menuentry "Arch Linux" {
    linux /boot/vmlinuz-linux root=/dev/sdal rw
    initrd /boot/initramfs-linux.img
}
```

загружаемся со внешнего носителя

1. Изготавливаем загрузочную флешку любым способом, например, с помощью unetbootin или dd
2. Вставляем флешку и включаем компьютер.
3. Дожидаемся появления экрана grub (иногда для того, чтобы успеть его поймать, надо держать shift)
4. Перед нами появляется список вариантов загрузки.
5. Нажимаем **c** и входим в интерактивный режим.
6. Теперь требуется указать носитель, с которого будем грузиться. Обычно (hd0) — это родной жесткий диск компьютера, а флешка становится (hd1). Выяснить, как назовется флешка в вашем случае, нетрудно просто опытным путем.
Так или иначе, вводим: **root (hd1)** для GRUB Legacy или **set root=(hd1)** для GRUB2
7. Просим передать управление загрузчику на указанном диске: **chainloader +1**
8. Загружаемся! **boot**

получаем консоль root'a

1. Аналогично, добираемся до списка вариантов загрузки.
2. Выбираем нужный нам вариант.
3. Входим в режим редактирования. Здесь есть небольшие отличия между GRUB Legacy и GRUB2. В GRUB2 после нажатия клавиши **e** мы сразу попадаем в режим редактирования, а в GRUB Legacy нужно нажать **e** первый раз, выбрать строку для редактирования и еще раз нажать **e**.
4. Выбираем строку, которая начинается со слова **linux** или **kernel**.
5. Удаляем из нее слова **quiet** и **splash**, если они есть, и дописываем в конец **single init=/bin/bash** (**single 1**)
6. Если у нас GRUB2, то сразу жмем **Ctrl+X**, а если GRUBLegacy — **Esc** и потом **b**

Защита паролем

GRUB2

```
/sbin/grub2-setpassword

Enter password:
Confirm password:
```

при попытке зайти в режим редактирования, будет запрошен пароль. пользователя надо указать "root"

хэш пароля будет сохранен в user.cfg:

```
cat /boot/grub2/user.cfg

GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.0.....
```

в grub.conf это выглядит вот так

```
if [ -f ${prefix}/user.cfg ]; then
    source ${prefix}/user.cfg
    if [ -n "${GRUB2_PASSWORD}" ]; then
        set superusers="root"
        export superusers
        password_pbkdf2 root ${GRUB2_PASSWORD}
    fi
fi
```

GRUB Legacy

```
| /sbin/grub-md5-crypt
```

вводим пароль - получает его md5-хеш. копикуес его в /boot/grub/grub.conf

строку добавляем сразу под "timeout"

```
| timeout=5  
| password --md5 <password-hash>
```

чтоб совсем запредить грузить систему без ввода пароля загрузчика добавим в нудном пункте меню сразу под "title"

```
| title CentOS (2.6.18-371.el5)  
| lock  
| password --md5 <password-hash>
```

Daemon Control Systems

Systemd

- Описание
 - Возможности
 - Архитектура
 - Преимущества перед System V Init
- Анализ состояния системы
- Юниты
 - Использование юнитов
- Управление питанием
- Написание файлов юнитов
 - Обработка зависимостей
 - Типы служб
 - Редактирование предоставленных пакетами файлов юнитов
 - Замена файлов юнита
 - Drop-in snippets
 - Примеры
- Цели
 - Получение информации о текущих целях (вместо runlevel):
 - Создание пользовательской цели
 - Таблица целей
 - Изменение текущей цели
 - Изменение цели загрузки по умолчанию
- Временные файлы
 - доп инфа по временным файлам
- Таймеры
- Журнал
 - Фильтрация вывода
 - Ограничение размера журнала
 - Очистка файлов журнала вручную
 - Перенаправить журнал на /dev/tty12
 - Команда просмотра другого журнала
- Оптимизация
 - Анализ процесса загрузки. Использование systemd-analyze
 - Чтобы увидеть список запускаемых файлов юнитов, отсортированный по потраченному каждым из них на загрузку времени, выполните команду:
 - Использование systemd-bootchart
- Решение проблем
 - Изучение ошибок systemd
 - Диагностика проблем в работе определенной службы
 - Отключение журналирования аварийных дампов памяти приложений

Описание

Это системный менеджер, демон инициализации других демонов в Linux, пришедший на замену используемого ранее /sbin/init. Его особенностью является интенсивное распараллеливание запуска служб в процессе загрузки системы, что позволяло существенно ускорить запуск операционной системы.

для получения более детальной информации:

```
| man 1 systemctl
```

Возможности

Помимо простого запуска и контроля сервисов SystemD предлагает некоторые другие удобные функции, для использования которых ранее приходилось использовать дополнительные программы-демоны. Среди таких функций:

- Сокет-активация служб (заменяет inetd)
- Запуск сервисов по расписанию (заменяет CRON)
- Работа с аппаратным сторожевым таймером (заменяет WatchDog - аппаратно реализованная схема контроля над зависанием системы. Представляет собой таймер, который периодически сбрасывается контролируемой системой. Если сброса не произошло в течение некоторого интервала времени, происходит принудительная перезагрузка системы.)
- Смена корня (заменяет chroot)
- Автомонтирование разделов дисков и сетевых ресурсов

Архитектура

Systemd оперирует специально оформленными файлами конфигурации - юнитами(unit).

Каждый юнит отвечает за отдельно взятую службу, точку монтирования, подключаемое устройство, файл подкачки, виртуальную машину и т.п.

Существуют специальные типы юнитов, которые не несут функциональной нагрузки, но позволяют задействовать дополнительные возможности systemd. К ним относятся юниты типа target, slice, automount и др.

- .target - позволяет группировать юниты, воплощая концепцию уровней запуска (runlevel)
- .service - отвечает за запуск сервисов (служб), также поддерживает вызов интерпретаторов для исполнения пользовательских скриптов
- .mount - отвечает за монтирование файловых систем
- .automount - позволяет отложить монтирование файловых систем до фактического обращения к точке монтирования
- .swap - отвечает за подключение файла или устройства подкачки
- .timer - позволяет запускать юниты по расписанию
- .socket - предоставляет службам поддержку механизма сокет-активации
- .slice - отвечает за создание контейнера cgroups (control group - механизм ядра Linux, который ограничивает и изолирует вычислительные ресурсы (процессорные, сетевые, ресурсы памяти, ресурсы ввода-вывода) для групп процессов. Механизм позволяет образовывать иерархические группы процессов с заданными ресурсными свойствами и обеспечивает программное управление ими.)
- .device - позволяет реагировать на подключение устройств
- .path - управляет иерархией файловой системы

Преимущества перед System V Init

- Контроль состояния службы, реакция на изменения состояния
- Сокет-активные и шина-активные службы, которые иногда приводят к лучшему распараллеливанию взаимозависимых служб.
- cgroups используется для отслеживания служебных процессов, вместо идентификаторов процессов (PID). Это означает, что демоны не будут потеряны даже после разветвления в другие процессы.

Анализ состояния системы

Список запущенных юнитов:

```
| $ systemctl  
$ systemctl list-units
```

Список неудач, - список юнитов, попытка запуска которых не удалась:

```
| $ systemctl --failed
```

Доступные файлы юнитов можно посмотреть в директориях (второй каталог имеет приоритет)

```
| /usr/lib/systemd/system/  
| /etc/systemd/system/
```

Список установленных файлов юнитов

```
| $ systemctl list-unit-files
```

Юниты

Использование юнитов

Юнитами могут быть, например, службы (.service), точки монтирования (.mount), устройства (.device) или сокеты (.socket).

При использовании systemctl обычно всегда необходимо указывать полное имя файла юнита, включая суффикс, например, `sshd.socket`.

Однако, есть несколько сокращений для указания юнита в следующих командах systemctl:

- Если суффикс не указан, systemctl предполагает, что это .service. Например, `netctl` и `netctl.service` будут трактоваться одинаково
- Точки монтирования будут преобразованы в соответствующий юнит .mount. Например, указание `/home` равнозначно `home.mount`
- Имена устройств преобразуются в соответствующий юнит .device, поэтому указание `/dev/sda2` полностью соответствует юниту `dev-sda2.device`

Для получения дополнительной информации

```
| man systemd.unit
```

Большинство указанных ниже команд также работают, если указать несколько юнитов.

command	description
<code>systemctl start</code>	
<code>systemctl stop</code>	
<code>systemctl restart</code>	
<code>systemctl reload</code>	
<code>systemctl status</code>	,
<code>systemctl is-enabled</code>	,
<code>systemctl enable</code>	
<code>systemctl disable</code>	
<code>systemctl mask</code>	,
<code>systemctl unmask</code>	
<code>systemctl help</code>	(,)
<code>systemctl daemon-reload</code>	systemd :

Управление питанием

Для управления питанием от имени непrivилегированного пользователя необходим [polkit](#). Если вы находитесь в локальной

пользовательской сессии `systemd-logind`, и нет других активных сессий, приведенные ниже команды сработают и без привилегий суперпользователя. В противном случае (например, вследствие того, что другой пользователь вошел в систему в tty), `systemd` автоматически запросит у вас пароль суперпользователя

command	description
<code>systemctl reboot</code>	
<code>systemctl poweroff</code>	()
<code>systemctl suspend</code>	
<code>systemctl hibernate</code>	
<code>systemctl hybrid-sleep</code>	(suspend-to-both)

Написание файлов юнитов

Файлы юнитов загружаются из двух мест. Вот они по приоритету от низшего к высшему

/usr/lib/systemd/system/	,
/etc/systemd/system/	,

согответственно тут можно посмотреть пример для написания своего файла юнита, а так же ман =)

```
| man systemd.service
```

Обработка зависимостей

В `systemd` зависимости могут быть указаны правильным построением файлов юнитов

Частный случай -- юниту A требуется, чтобы юнит B был запущен перед тем, как запустится сам юнит A. В этом случае добавьте строки `R` `requires=B` и `AAfter=B` в секцию `[Unit]` файла службы A

Если подобная зависимость не является обязательной, взамен указанных выше добавьте, соответственно, строки `Wants=B` и `AAfter=B`

`Wants=` и `Requires=` не подразумевают `After=`, что означает, что **если `After=` не определено, два юнита будут запущены параллельно друг другу.**

Типы служб

Существует несколько различных типов запуска служб, которые надо иметь в виду

Type=simple	(по умолчанию) предполагает, что служба будет запущена немедленно. Процесс при этом не должен разветвляться. Не используйте этот тип, если другие службы зависят от очередности при запуске данной службы. Исключение - активация сокета
Type=forking	предполагает, что служба запускается однократно и процесс разветвляется с завершением родительского процесса. Данный тип для запуска классических демонов за исключением тех случаев, когда, в таком поведении процесса нет необходимости. Следует определить <code>PIDFile=</code> , чтобы <code>systemd</code> могла отслеживать основной процесс
Type=oneshot	полезен для скриптов, которые выполняют одно задание и завершаются. Может понадобиться также установить параметр <code>RemainAfterExit=yes</code> , чтобы <code>systemd</code> по-прежнему считала процесс активным, даже после его завершения
Type=notify	Идентичен параметру <code>Type=simple</code> , но демон пошлет <code>systemd</code> сигнал о своей готовности. Эталонная реализация данного уведомления представлена в <code>libsystemd-daemon.so</code>

Type=dbus	Сервис считается в состоянии готовности, когда определенное BusName появляется в системной шине DBus
Type=idle	поведение очень похоже на Type=simple. Будет задержено выполнение службы до тех пор пока пока все активные работы не будут выполнены.

дока по типам тут: <https://www.freedesktop.org/software/systemd/man/systemd.service.html#Type=>

ну и ман

```
| man systemd.service
```

Редактирование предоставленных пакетами файлов юнитов

Можно заменить весь блок файла на новый или создать фрагмент кода, который применяется в верхней части существующего блока файла.

В обоих методах, чтобы применить изменения, нужно перезагрузить юнит

```
| systemctl daemon-reload
```

или редактировать юнит средствами systemd, тогда он будет загружен автоматически

```
| systemctl edit
```

чтобы увидеть, какие файлы юнитов были переопределены и что конкретно было изменено

```
| systemd-delta
```

посмотреть содержимое файла юнита и все кастомные врагменты (Drop-in snippets) кода

```
| systemctl cat
```

Подсветка в vim включается установкой пакета vim-systemd из стандартных реп

Замена файлов юнита

Чтобы заменить файл юнита /usr/lib/systemd/system/, создайте файл /etc/systemd/system/ и перезапустите юнит для обновления символьных ссылок

```
| systemctl reenable
```

или так. (Эта команда откроет /etc/systemd/system/юнит в вашем текстовом редакторе (копирует установленную версию, если она еще не существует) и автоматически загружает её, когда вы закончите редактирование)

```
| systemctl edit --full
```

Drop-in snippets

Чтобы создать drop-in snippets для файла юнита /usr/lib/systemd/system/, создайте каталог /etc/systemd/system/.d/ и поместить файлы .conf там, чтобы отменять или добавлять новые опции. systemd будет анализировать эти файлы .conf и применять их поверх оригинального юнита.

Самый простой способ чтобы выполнить это, представлен ниже. (команда откроет /etc/systemd/system/юнит.d/override.conf (создаст его если это потребуется) в вашем текстовом редакторе и автоматически перезапустит юнит, когда вы закончите редактирование.)

```
| systemctl edit
```

Примеры

Чтобы просто добавить дополнительную зависимость к юниту, можно создать следующий файл:

```
/etc/systemd/system/unit.d/customdependency.conf
[Unit]
Requires=new dependency
After=new dependency
```

чтобы заменить направление для юнита ExecStart (ExecStart должна быть очищена, перед новым назначением), что не относится к типу oneshot, создадим следующий файл:

```
/etc/systemd/system/unit.d/customexec.conf
```

```
[Service]
ExecStart=
ExecStart=
```

чтобы автоматически перезапустить службу:

```
/etc/systemd/system/unit.d/restart.conf
```

```
[Service]
Restart=always
RestartSec=30
```

Цели

выполняют ту же задачу, что и уровни запуска (runlevel), но действуют немного по-другому

Каждая цель поименована (т.е. имеет собственное имя, а не номер) и, как предполагается, предназначена для конкретных задач; возможно иметь в одно и то же время активными несколько таких целей.

Некоторые цели реализованы так, что наследуют все службы других целей, добавляя к ним свои.

имеются также цели, которые имитируют общие уровни запуска SystemV init. Переключаться между целевыми юнитами, можно используя команду `telinit RUNLEVEL`

Получение информации о текущих целях (вместо `runlevel`):

```
| systemctl list-units --type=target
```

Создание пользовательской цели

Уровни запуска, по которым расписаны конкретные задачи по умолчанию - 0, 1, 3, 5 и 6 - имеют соответствие 1:1 с конкретными целями `systemd`.

не существует хорошего способа сделать то же самое для определяемых пользователем уровней, таких как 2 и 4. Их использование предполагает, что мы создаем новый именованный целевой юнит `systemd` наподобие `/etc/systemd/system/`, который берет за основу один из существующих уровней запуска (пример `/usr/lib/systemd/system/graphical.target`), создаем каталог `/etc/systemd/system/ .wants`, а после этого - символические ссылки на дополнительные службы из директории `/usr/lib/systemd/system/`, которые мы хотим включить при загрузке.

Таблица целей

SysV	system	
0	runlevel0.target, poweroff.target	Выключить систему
1, s, single	runlevel1.target, rescue.target	
2, 4	runlevel2.target, runlevel4.target, multi-user.target	, / . 3
3	runlevel3.target, multi-user.target	. , ,
5	runlevel5.target, graphical.target	. 3
6	runlevel6.target, reboot.target	
emergency	emergency.target	

Изменение текущей цели

В `systemd` цели доступны посредством целевых юнитов

```
| systemctl isolate graphical.target
```

Данная команда изменит только лишь текущую цель и не повлияет на следующую загрузку системы. Она соответствует командам SysV Init вида `telinit 3` и `telinit 5`

Изменение цели загрузки по умолчанию

Стандартная цель - `default.target`, которая по умолчанию является псевдонимом `multi-user.target` (примерно соответствующего прежнему уровню запуска 3)

Для изменения цели загрузки по умолчанию добавьте один из следующих параметров ядра в загрузчик одну из перечисленных ниже строк:

```
systemd.unit=graphical.target  
systemd.unit=multi-user.target  
systemd.unit=rescue.target
```

Другой способ - оставить загрузчик без изменений, а изменить целевой юнит по умолчанию - `default.target` используя `systemctl`

```
| systemctl set-default multi-user.target
```

или с форсом, чтобы перезаписать ранее созданный файл

```
| systemctl set-default -f multi-user.target
```

Символическая ссылка на новый целевой юнит по умолчанию создается в директории `/etc/systemd/system/default.target`

```
| ll /etc/systemd/system/default.target  
lwxrwxrwx. 1 root root 37 Nov 4 23:09 /etc/systemd/system/default.target ->  
/lib/systemd/system/multi-user.target
```

Временные файлы

`systemd-tmpfiles` создает, удаляет и очищает непостоянные и временные файлы и каталоги

Он читает конфигурационные файлы из `/etc/tmpfiles.d/` (приоритетный) и `/usr/lib/tmpfiles.d/`, чтобы понять, что ему следует делать.

Например, демон Samba предполагает, что существует каталог `/run/samba` с корректными правами доступа. Поэтому пакет `samba` поставляется в следующей конфигурации:

```
| /usr/lib/tmpfiles.d/samba.conf  
| D /run/samba 0755 root root
```

Конфигурационные файлы также могут использоваться для записи значений при старте системы

вместо `echo USBE > /proc/acpi/wakeup` :

```
| /etc/tmpfiles.d/disable-usb-wake.conf  
| w /proc/acpi/wakeup - - - - USBE
```

доп инфа по временным файлам

```
| man 8 systemd-tmpfiles  
| man 5 tmpfiles.d
```

Таймеры

Таймер - это файл конфигурации юнита, имя которого заканчивается на `.timer`. Он расшифровывает информацию о таймере, контролируемом при помощи `systemd`, для активации в определенное время. См. статью [systemd/Таймеры](#).

Таймеры способны в значительной степени заменить функциональность `cron`. См. раздел [Замена cron](#)

Журнал

systemd имеет собственную систему ведения логов, названную журналом (journal). В связи с этим больше не требуется запускать демон **syslog**. Для чтения логов используем команду:

```
| journalctl
```

конфиг лежит тут:

```
/etc/systemd/journald.conf

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
#RateLimitBurst=1000
#SystemMaxUse=
#SystemKeepFree=
#SystemMaxFileSize=
#SystemMaxFiles=100
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#RuntimeMaxFiles=100
#MaxRetentionSec=
#MaxFileSec=1month
#ForwardToSyslog=no
#ForwardToKMsg=no
#ForwardToConsole=no
#ForwardToWall=yes
#TTYPath=/dev/console
#MaxLevelStore=debug
#MaxLevelSyslog=debug
#MaxLevelKMsg=notice
#MaxLevelConsole=info
#MaxLevelWall=emerg
```

Storage=auto	если каталог <code>/var/log/journal/</code> будет удален (или еще не существует), systemd не пересоздаст его автоматически и вместо этого будет писать свои журналы по непостоянному пути <code>/run/systemd/journal</code>
Storage=persistent	, <pre> systemctl restart systemd-journald</pre>

Фильтрация вывода

журнал хранится в двоичном формате, содержимое его сообщений не меняется. Это означает, что их можно просматривать при помощи `strings`, например, в окружении, в котором не установлен `systemd`. Пример:

```
| strings /mnt/arch/var/log/journal/af4967d77fba44c6b093d0e9862f6ddd/system.journal | grep -i
```

Показать все сообщения с момента текущей загрузки системы

```
| journalctl -b
| journalctl -b 0
```

Показать все сообщения с момента предыдущей загрузки

```
| journalctl -b -1
```

Показать все сообщения с момента следующей за предыдущей, и т.д.

```
| journalctl -b -2
```

полная справка по работе с журналом

```
| man 1 journalctl
| man 7 systemd-journal-fields
```

Показать все сообщения, начиная с какой-либо даты (и, если хотите, времени):

```
| journalctl --since="2012-10-30 18:17:16"
```

Показать все сообщения за последние 20 минут:

```
| journalctl --since "20 min ago"
```

Показывать новые сообщения:

```
| journalctl -f
```

Показать все сообщения для конкретного исполняемого файла:

```
| journalctl /usr/lib/systemd/systemd
```

Показать все сообщения для конкретного процесса:

```
| journalctl _PID=1
```

Показать все сообщения для конкретного юнита:

```
| journalctl -u netcfg
```

Показать кольцевой буфер ядра:

```
| journalctl -k
```

Показать auth.log эквивалентно фильтрации syslog facility:

```
| journalctl -f -l SYSLOG_FACILITY=10
```

полезный пост по теме

<http://0pointer.de/blog/projects/journalctl.html>

Перенос длинных строк

По умолчанию journalctl отсекает части строк, которые не вписываются в экран по ширине less запускается с опциями FRSXMK (если умрать S то строки будут переноситься на новую строку)

```
| SYSTEMD_LESS=FRSXMK journalctl
```

чтобы такое поведение использовалось по умолчанию, экспортируем переменную из файла `~/.bashrc` или `~/.zshrc`

Ограничение размера журнала

размер по умолчанию ограничен значением в 10% от объема соответствующей файловой системы.

Максимальный объем постоянного журнала можно контролировать при помощи значения `SystemMaxUse` в конфигурационном файле `/etc/systemd/journald.conf`

```
| /etc/systemd/journald.conf
```

```
| SystemMaxUse=50M
```

доп инфа в мане

```
| man journald.conf
```

Очистка файлов журнала вручную

Файлы журнала находятся в `/var/log/journal`, так что rm будет работать

или используем `journalctl`

удалить старые записи, до размера журнала 100M

```
| journalctl --vacuum-size=100M
```

очистить все журналы старше 2 недель

```
| journalctl --vacuum-time=2weeks
```

доп инфа в мане

```
| man journalctl
```

Перенаправить журнал на /dev/tty12

Создайте drop-in каталог `/etc/systemd/journald.conf.d` и создайте файл `fw-tty12.conf` с содержимым

```
/etc/systemd/journald.conf.d/fw-tty12.conf
[Journal]
ForwardToConsole=yes
TTYPath=/dev/tty12
MaxLevelConsole=inf
```

и перезапустим `systemd-journald`.

Команда просмотра другого журнала

римонтируйте диск неисправной системы, например в `/mnt` и укажите путь журнала через `-D/--directory`, например так:

```
$ journalctl -D /mnt/var/log/journal -xe
```

Оптимизация

Анализ процесса загрузки. Использование `systemd-analyze`

`systemd-analyze`, позволяет проанализировать процесс загрузки системы, чтобы можно было увидеть, какие файлы юнитов тормозят загрузку.

Соответственно, можно оптимизировать систему.

Для использования данного инструмента вам потребуется установить пакеты `python2-cairo` и `python2-gobject`.

Чтобы увидеть, сколько времени было потрачено на подготовку пространства ядра и пространства пользователя во время загрузки, просто выполните команду:

```
| systemd-analyze
```

Если вы дополните хуком `timestamp` ваш массив `HOOKS` в конфигурационном файле `/etc/mkinitcpio.conf` и пересоберете ваш образ initramfs командой `mkinitcpio -p linux`, `systemd-analyze` сколько времени затрачивается на initramfs.

Чтобы увидеть список запускаемых файлов юнитов, отсортированный по потраченному каждым из них на загрузку времени, выполните команду:

```
| systemd-analyze blame
```

файл SVG, показывающий процесс загрузки в графическом виде, наподобие Bootchart:

```
| $ systemd-analyze plot > plot.svg
```

!!!! УЗНАТЬ ЧЕМ ЕГО ОТКРЫТЬ!!!

Использование `systemd-bootchart`

вы можете использовать его для загрузки также, как и оригинальный `bootchart`. Добавьте следующие команду к строке инициализации ядра:

```
| initcall_debug printk.time=y init=/usr/lib/systemd/systemd-bootchart
```

Решение проблем

Изучение ошибок `systemd`

изучим ошибки службы `systemd-modules-load`

1. найдем службы `systemd`, которые не смогли запуститься

```
| systemctl --failed
```

```
systemd-modules-load.service loaded failed failed Load Kernel Modules
```

2. мы обнаружили проблему в службе `systemd-modules-load` и хотим узнать больше:

```
systemctl status systemd-modules-load

systemd-modules-load.service - Load Kernel Modules
  Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
  Active: failed (Result: exit-code) since So 2013-08-25 11:48:13 CEST; 32s ago
    Docs: man:systemd-modules-load.service(8).
          man:modules-load.d(5)
   Process: 15630 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited,
               status=1/FAILURE)
```

Если не видим в списке Process ID, просто перезапустите службу при помощи команды

```
systemctl restart systemd-modules-load
```

3. Для более детального изучения ошибки. Введите следующую команду с правильным Process ID (в данном примере это 15630):

```
journalctl _PID=15630

-- Logs begin at Sa 2013-05-25 10:31:12 CEST, end at So 2013-08-25 11:51:17 CEST. --
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'blacklist usblp'
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'install usblp
/bin/false'
```

4. Мы видим, что некоторые конфигурационные файлы модулей ядра имеют неверные настройки. В этом случае мы взглянем на эти настройки в каталоге `/etc/modules-load.d/`

```
ls -Al /etc/modules-load.d/

...
-rw-r--r-- 1 root root 79 1. Dez 2012 blacklist.conf
-rw-r--r-- 1 root root 1 2. Mär 14:30 encrypt.conf
-rw-r--r-- 1 root root 3 5. Dez 2012 printing.conf
-rw-r--r-- 1 root root 6 14. Jul 11:01 realtek.conf
-rw-r--r-- 1 root root 65 2. Jun 23:01 virtualbox.conf
...
```

5. Сообщение об ошибке `Failed to find module 'blacklist usblp'` должно относиться к неправильной настройке в файле `blacklist.conf`.

Закомментируем настройку, вставив `#` перед каждой опцией, найденной на шаге 3:

```
/etc/modules-load.d/blacklist.conf

# blacklist usblp
# install usblp /bin/false
```

6. Теперь попробуйте запустить `systemd-modules-load`

```
systemctl start systemd-modules-load
```

Если все прошло успешно, ничего не отобразится, если нет, смотрим ошибки нового пада

Если все хорошо, вы можете удостовериться, что служба успешно запустилась, при помощи команды:

```
systemctl status systemd-modules-load

systemd-modules-load.service - Load Kernel Modules
  Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
  Active: active (exited) since So 2013-08-25 12:22:31 CEST; 34s ago
    Docs: man:systemd-modules-load.service(8)
          man:modules-load.d(5)
   Process: 19005 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited,
               status=0/SUCCESS)
 Aug 25 12:22:31 mypc systemd[1]: Started Load Kernel Modules.
```

Диагностика проблем в работе определенной службы

Присвоим

переменной окружения SYSTEMD_LOG_LEVEL значение debug. Например, чтобы запустить демон systemd-networkd в режиме отладки:

```
| systemctl stop systemd-networkd  
| SYSTEMD_LOG_LEVEL=debug /lib/systemd/systemd-network
```

Альтернатива. Можно временно отредактировать файл службы для получения подробного вывода. Например:

```
| /usr/lib/systemd/system/systemd-networkd.service  
| [Service]  
| ...  
| Environment=SYSTEMD_LOG_LEVEL=debug  
| ...
```

Отключение журналирования аварийных дампов памяти приложений

Добавим в файл **/etc/systemd/coredump.conf** такую строку:

```
| Storage=none
```

перезагрузить конфигурацию

```
| systemctl daemon-reload
```

UNIX System V

после инициализации ядра, ядро запускает **/sbin/init** как первый процесс

init отвечает за дальнейшую загрузку системы. Для этого он запускает так называемые стартовые скрипты, которые выполняют проверку и монтируют файловых систем, запуск необходимых демонов, настройку ядра (в том числе загрузку модулей ядра согласно установленному оборудованию, настройку IP-адресов, таблиц маршрутизации и др.), запуск графической оболочки и другие действия.

процесс **init** запускается и анализирует файл **/etc/inittab**

```
$ cat /etc/inittab  
  
id:3:initdefault:  
si::sysinit:/etc/rc.d/rc.sysinit  
  
10:0:wait:/etc/rc.d/rc 0  
11:1:wait:/etc/rc.d/rc 1  
12:2:wait:/etc/rc.d/rc 2  
13:3:wait:/etc/rc.d/rc 3  
14:4:wait:/etc/rc.d/rc 4  
15:5:wait:/etc/rc.d/rc 5  
16:6:wait:/etc/rc.d/rc 6  
  
1:2345:respawn:/sbin/mingetty tty1  
2:2345:respawn:/sbin/mingetty tty2  
3:2345:respawn:/sbin/mingetty tty3  
4:2345:respawn:/sbin/mingetty tty4  
5:2345:respawn:/sbin/mingetty tty5  
6:2345:respawn:/sbin/mingetty tty6
```

В первой строке описан терминал и его конфигурация по умолчанию.

Сначала в этом файле описываются уровни инициализации.

Затем инициируются виртуальные консоли.

Запись инициализации консолей состоит из полей, разделенных двоеточием и выглядит следующим образом:

- 1 — порядковый номер консоли
- 2345 — номера уровней инициализации, для которых консоль инициализируется
- respawn — этот параметр означает, что **init** должен перезапустить обслуживающий консоль процесс после выхода из сеанса или в случае краха.
- **/sbin/mingetty tty6** — программа (с указанием параметров), которая будет обслуживать консоль.

Можно создать свой уровень инициализации (под номером 4 или 7, 8...), просто исправив файл /etc/inittab и создав необходимые ссылки в каталоге /etc/rc.d/rc*.d

По умолчанию, в системе использовано 7 уровней инициализации:

- 0 — остановка системы
- 1 — загрузка в однопользовательском режиме
- 2 — загрузка в многопользовательском режиме без поддержки сети
- 3 — загрузка в многопользовательском режиме с поддержкой сети
- 4 — не используется
- 5 — загрузка в многопользовательском режиме с поддержкой сети и графического входа в систему
- 6 — перезагрузка

В большинстве Unix/Linux систем, узнать текущий уровень инициализации можно командами:

```
$ runlevel  
$ who -r
```

Upstart

замена системы инициализации init в UNIX и Linux системах.

Первоначально была разработана для дистрибутива Ubuntu

В настоящее время признана устаревшей и для многих дистрибутивов планируется переход на systemd.

Краткий обзор

Возможности

- Задачи и службы запускаются и останавливаются по событиям
- События генерируются задачами и службами
- События могут быть приняты от любого процесса системы
- Сервисы могут быть перезапущены, если они были завершены
- Взаимодействие с демоном init посредством D-Bus (D-Bus — система межпроцессного взаимодействия, которая позволяет приложениям в операционной системе сообщаться друг с другом)
- Можно организовывать свои события

По умолчанию её используют системы Red Hat Enterprise Linux (RHEL) 6, Chrome OS и дистрибутивы Ubuntu до версии 15.04.

Процессы

рабочие процессы (jobs) делятся на:

- задачи (task jobs)
- сервисы (service jobs, могут работать в фоновом режиме)
- процессы, которые работают до тех пор, пока их не остановит пользователь

События

Это сигналы, или вызовы, инициирующие определённое действие.

Наиболее общим примером события является мониторинг процесса

- starting
- started
- stopping
- stopped

Реализация событий

Как правило, события запускаются процессами, но в случае необходимости можно запустить их вручную с помощью команды:

```
| initctl emit <event>
```

ещё одна полезная команда для работы с Upstart:

```
| init control
```

Конфигурации процесса

Чтобы настройки процесса были валидными, конфигурационный файл должен соответствовать таким требованиям:

- Он не должен быть пустым
- Не должен содержать ошибок
- Должен содержать хотя бы один блок команд (stanza)

Создадим файл testjob.conf в каталоге /etc/init/

```
| touch /etc/init/testjob.conf
```

это процесс будет записывать сообщение и текущую метку времени в лог-файл

Существует два блока, с помощью которых можно определить цель и автора процесса: `description` и `author`. С них должен начинаться файл:

```
| description "A test job file for experimenting with Upstart"  
| author "Artem Kosenko"
```

Для того чтобы процесс загружался после того, как загружаются все остальные сервисы и процессы добавим строку

```
| start on runlevel [2345]
```

Теперь нужно добавить исполняемый код.

Строка начинается с `exec`, чтобы обозначить, что данный код должен быть обработан оболочкой Bash

```
| exec echo Test Job ran at `date` >> /var/log/testjob.log
```

(если не использовать обратные кавычки `` то команда не будет выполнена, а просто будет отображена на экране как текст "date")

проверить синтаксис на ошибки:

```
| init-checkconf /etc/init/testjob.conf
```

Если команда обнаружила ошибки, исправьте их. Если ошибок нет, команда вернёт:

```
| File /etc/init/testjob.conf: syntax ok
```

управлять процессами и фоновыми сервисами (например веб-сервером)

```
| service <servicename> <control>
```

В поле `control` можно использовать такие команды:

- `restart`: перезапуск
- `start`: запуск
- `stop`: остановка сервиса
- `status`: состояние сервиса

Запустим процесс ручками

```
| service testjob start
```

проверим наш лог

```
cat /var/log/testjob.log
```

Состояние процессов и события

место хранения

/etc/init/	
~/.init/	

Рабочие процессы пользователей запускаются в их сессиях. Такие процессы не являются общесистемными

Все процессы всегда определяются в конфигурационном файле (.conf)
имя должно представлять сервис или выполняемую задачу

Каждая такая задача имеет целью запуск (start) или остановку (stop)

Между этими двумя целями находится ряд состояний задачи, который определяет текущее действие процесса в зависимости от цели

- waiting: исходное состояние процесса.
- starting: подготовка к запуску.
- pre-start: загрузка предпусковых задач.
- spawned: запуск разделов сценария.
- post-start: выполнение операций после запуска процесса.
- running: процесс полностью запущен.
- pre-stop: подготовка к остановке процесса.
- stopping: остановка процесса.
- killed: процесс остановлен.
- post-stop: чистка окружения после остановки процесса.

Процесс в состоянии **post-start** считается запущенным процессом

остаётся запущенным до состояния **pre-stop**, в котором он готовится к остановке

процесс останавливается и переходит в состояние **post-stop** (очистка системы).

Debug log

Чтобы увидеть, как процесс изменяет свои состояния, переведем приоритет лога Upstart (/var/log/upstart/) в debug

```
| initctl log-priority debug
```

СОСТОЯНИЯ И СОБЫТИЯ – НЕ ОДНО И ТО ЖЕ

Процессы сервисов

для примера установим сервер Node.js

```
| apt-get install nodejs
```

создадим для него сервис

```
| touch /etc/init/nodetest.conf
```

добавим в файл описание и автора процесса

```
| description "Service for a test node.js server"
| author "Artem Kosenko"
```

автоматический запуск и отключение этого приложения на основе Node

```
| start on filesystem or runlevel [2345]
| stop on shutdown
```

Уровень запуска в сочетании с событием **filesystem** обеспечивает запуск процесса во время загрузки сервера

Поскольку это приложение является серверным, нужно добавить в конфигурации логирование.

Чтобы зарегистрировать запуск и остановку приложения, используем действия script, pre-start script и pre-stop script

pre-start и pre-stop – это состояния процессов, но их можно добавлять в блоки

Сначала нужно добавить сам сценарий процесса.

Он получит ID процесса для фонового сервера Node, а затем запустит сценарий приложения. **Внимательно с отступами!!!**

```
script
  export HOME="/srv"
  echo $$ > /var/run/nodetest.pid
  exec /usr/bin/nodejs /srv/nodetest.js
end script
```

Приложению Node необходима переменная домашнего каталога, потому /srv экспортируется в первой строке блока. Символы \$\$ устанавливают PID процесса и создают файл для него.

pre-start и pre-stop. Дата, а также сообщения о запуске и остановке процесса, будут внесены в лог

```
pre-start script
  echo "[`date`] Node Test Starting" >> /var/log/nodetest.log
end script
```

блок pre-stop содержит строку, которая удаляет PID-файл. Эта задача будет частью процесса остановки

```
pre-stop script
  rm /var/run/nodetest.pid
  echo "[`date`] Node Test Stopping" >> /var/log/nodetest.log
end script
```

В результате файл будет иметь такой вид:

```
description "Test node.js server"
author      "Your Name"
start on filesystem or runlevel [2345]
stop on shutdown
script
  export HOME="/srv"
  echo $$ > /var/run/nodetest.pid
  exec /usr/bin/nodejs /srv/nodetest.js
end script
pre-start script
  echo "[`date`] Node Test Starting" >> /var/log/nodetest.log
end script
pre-stop script
  rm /var/run/nodetest.pid
  echo "[`date`] Node Test Stopping" >> /var/log/nodetest.log
end script
```

созраняем и закрываем!

Как сказано в строке exec, сценарий Node.js будет запущен с сервера, потому нужно создать файл nodetest.js:

```
sudo nano /srv/nodetest.js
```

Полученный сценарий выполняет следующие действия:

- Запрашивает HTTP-модуль Node.
- Создаёт веб-сервер HTTP.
- Provide a status 200 (OK) response in the Header
- Возвращает фразу Hello World.
- Прослушивает порт 8888.

При помощи следующего кода можно запустить приложение Node:

```
var http = require("http");
http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello World");
  response.end();
}).listen(8888);
```

проверка синтаксиса

```
init-checkconf /etc/init/nodetest.conf
File nodetest.conf: syntax ok
```

Еслм код не содержит ошибок, перезапустим сервер и откроем ссылку

<http://IP:8888>

На экране появится фраза Hello World.

проверим наш лог

```
| cat /var/log/nodetest.lo
```

Для запуска стандартных команд (start, stop, restart и т.д.) используется следующий синтаксис:

```
| service nodetest restart
```

Полезная линка на закуску

[Установка Apache Tomcat 8 на сервер Ubuntu 14.04](#)

Processes & Threads

/proc/

CPU info & control

Get CPU info methods

1. /proc/cpuinfo

Содержить инфу по каждому ядру в отдельности. Можно спросмотреть less или cat

```
| cat /proc/cpuinfo  
| less /proc/cpuinfo
```

считать ядра (вместе с хиперсридингом) процессора можно так

```
| cat /proc/cpuinfo | grep processor | wc -l  
| 4
```

проверяем настоящее кол-во чистых ядер

пример с 2 физическими ядрами, но с 4 потоками (вирт ядра из-за хиперсридинга)

```
| cat /proc/cpuinfo | grep 'core id'  
| core id : 0  
| core id : 1  
| core id : 0  
| core id : 1
```

пример с настоящими 4 ядрами и 4 потоками (тут все честно)

```
| cat /proc/cpuinfo | grep 'core id'  
| core id : 0  
| core id : 2  
| core id : 1  
| core id : 3
```

2. lscpu

Также инфа что и в /proc/cpuinfo но в удобном виде, выдается одним блоком, в отличии от первого варианта.
не требует никаких опций

```
| lscpu
```

3. hardinfo

графическая утилита для сбора подробной инфы о системе. среди прочего есть режим отчета с ключем -r

```
|
```

```
| hardinfo -r
```

4. lshw

без параметров показывает довольно тетальную инфу о системе в целом. но нам ужен только проц

```
| lshw -class processor
```

5. nproc

быстро покадет только цифру кол-ва процов вместе с гиперсайдингом (если есть)

```
| nproc  
4
```

6. dmidecode

инфо по железу, в том числе и по проу

```
| sudo dmidecode -t 4
```

7. cpuid

подробнаяя инфа об INTEL и AMD роцах. Обычно надо доставлять пакет

```
| cpuid
```

8. inxi

красивенький вывод базовой инфы по процу, хорош для вывода в какой-то панельки, прогреме и т.п. Обычно надо доставлять пакет.

```
| inxi -
```

CPU control methods

Эмитируем брудую деятельность =)

нагрузим наш камушек математическими задачками, например генирацией чисел. скорей всего понадобится доставить <http://mathomatic.orgserve.de/mathomatic-16.0.5.tar.bz2>, make, make install...

В арче он оказался в реце под названием mathomatic

```
| /usr/bin/matho-primes 0 9999999999 > /dev/null
```

это нагрузит на 100 одно ядро. т.е. если у нас 4 ядра, то запустим таких процесса 4 штуки и получим полную загружку всех ядер

Управление приоритетом

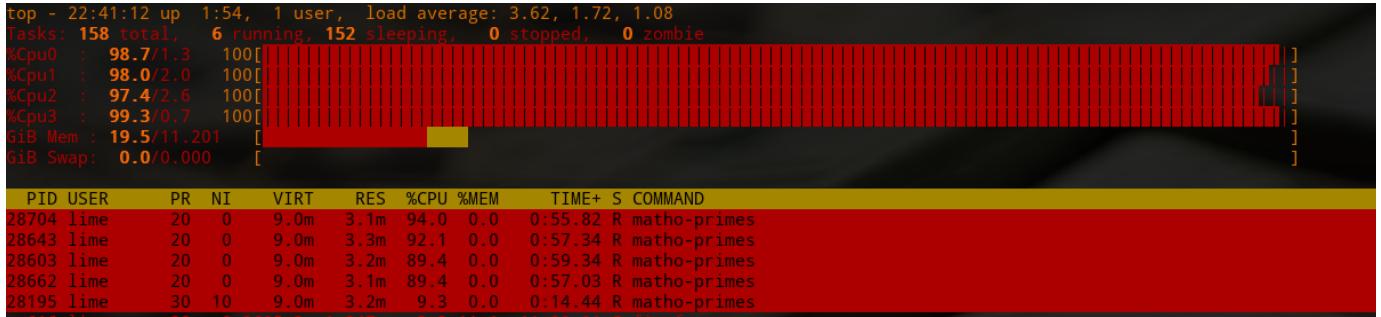
NICE, RENICE

позволяет задать приоритет запускаемой программе от -20 (наивысший приоритет волонтерства) до 19 (Наименьший приоритет)

если запустить без параметров - будет установлен приоритет 10 (ниже стандартного приоритета "0")

```
| nice /usr/bin/matho-primes 0 9999999999 > /dev/null
```

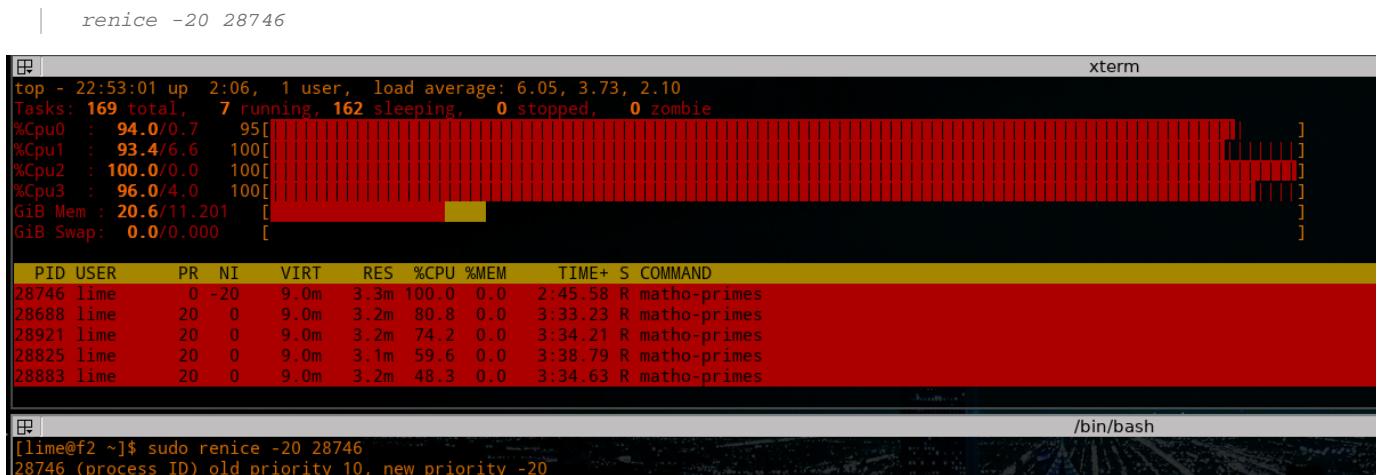
как видим запущенный с приоритетом 10 процессы "в пролете" когда проц заюзан до отказа



NI - это и есть приоритет процесса.

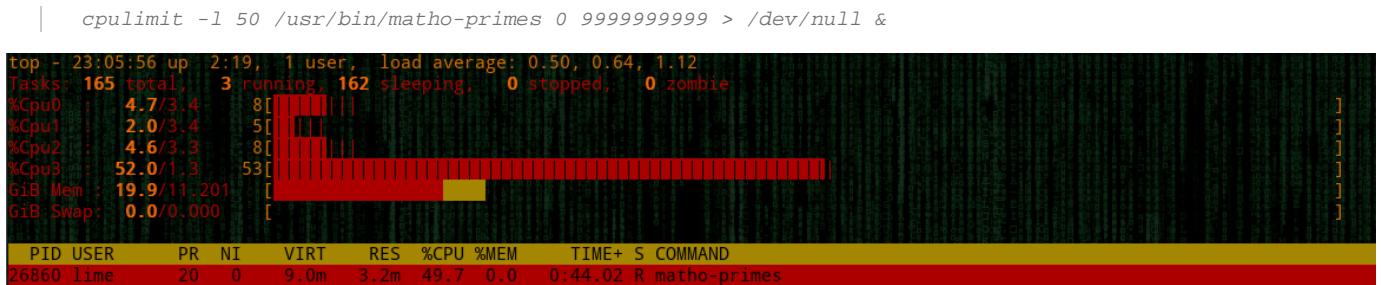
(NI = 10) менее приоритетный процесс смог "отъесть" только 9.3% процессорного времени одного из занятых на 100% ядер
(NI = 0 стандарт) более приоритетные процессы пригрузили ядра на все свободное процессорное время.

меняем приоритет на максимальный, для ранее запущенного с низким приоритетом (NI = 10) и он тут же занимает первую строку в топе:



CPULIMIT

вставляет "пали в колеса" процессу (посыпает сигналы SIGSTOP и SIGCONT) и не дает вылезать процессорное время ядра выше указанного лимита



изменить лимит для уже запущенного процесса:

```
| cpulimit -l 70 -p 28573
```

работает для процессов, которые не были запущены с помощью cpulimit изначально.

CGROUPS

Control groups (cgroups) - эта функция в Linux позволяет задать то, как ядро будет распределять указанные ресурсы для групп процессов.

Можно указать сколько процессорного времени, памяти, и bandwidth может быть использовано процессами в определенной cgroup

cgroups позволяют применить лимиты к группе процессов, а не только к одному из них как это делают nice и cpulimit (к тому же они позволяют лимитировать только процессорное время, а cgroups еще нарезать еще и память+сеть)

к примеру cgroups используется в CentOS при обновлении системы. Это значит что весь процесс загрузки и установки не оказывает особого влияния на общей производительности системы!!!

С появление Systemd синтаксис поменялся

для старых ОС <http://blog.scoutapp.com/articles/2014/11/04/restricting-process-cpu-usage-using-nice-cpulimit-and-cgroups>

для Systemd <https://www.certdepot.net/rhel7-get-started-cgroups/>

<http://lexpr.ru/node/516>

отличная дока на русском по Systemd http://www2.kangran.su/~nnz/pub/s4a/s4a_latest.pdf

CPU Limits

CPUShares=

в конфигурационном файле службы. По умолчанию это значение равно 1024. Увеличивая это число, мы даем службе больше процессорного времени, уменьшая — соответственно, меньше.

Создаем файл «ручных» настроек /etc/systemd/system/httpd.service, который включает в себя все те же опции, что и файл настроек по умолчанию /usr/lib/systemd/system/httpd.service, отличаясь от него только значением CPUShares=

```
.include /usr/lib/systemd/system/httpd.service
[Service]
CPUShares=1500
```

Первая строка обеспечивает включение в нашу конфигурацию файла с настройками по умолчанию, сделанными разработчиками Apache

Применим изменения в службе

```
systemctl daemon-reload
systemctl restart httpd.servic
```

RAM Limits

*MemoryLimit=
MemorySoftLimit=*

указывается предел потребления памяти в байтах. При этом поддерживаются суффиксы К, М, Г и Т, обозначающие соответственно, килобайт, мегабайт, гигабайт и терабайт (по основанию 1024).

```
.include /usr/lib/systemd/system/httpd.service
[Service]
MemoryLimit=1G
```

Применим изменения в службе

```
systemctl daemon-reload
systemctl restart httpd.servic
```

I/O Limits

*BlockIOWeight=
BlockIORReadBandwidth=
BlockIORWriteBandwidth=*

Задаем долю полосы I/O, вообще и отдельно чтение - отдельно запись. Значение может быть от 10 до 1000 (по-умолчанию). Параметр задает придельную скорость чтения/записи в байтах в секунду:

Уменьшить долю для службы Apache можно так:

```
.include /usr/lib/systemd/system/httpd.service
[Service]
BlockIOWeight=500
```

Можно задать такое значение отдельно для каждого устройства:

```
.include /usr/lib/systemd/system/httpd.service
[Service]
BlockIOWeight=/dev/disk/by-id/ata-SAMSUNG_MMCRE28G8MXP-0VBL1_DC06K01009SE009B5252
```

точное название устройства знать не обязательно — достаточно указать интересующий нас каталог:

```
.include /usr/lib/systemd/system/httpd.service
[Service]
BlockIOWeight=/home/lennart 750
```

NET Limits

пример юнита с лимитами

```
[Unit]
Description=Foo

[Service]
Type=forking
CPUShares=1500
MemoryLimit=3M
BlockIOWeight=500
ExecStart=/1.sh

[Install]
WantedBy=multi-user.target
```

Просмотреть дерево групп процессов

```
| systemd-cgls
```

Посмотреть какая группа сколько ресурсов жрет (TOP)

```
| systemd-cgtop
```

Прибрить все что относится к определенной группе

можно использовать -s option чтобы указать нужный сигнал (SIGTERM, SIGINT, SIGSTOP или SIGTERM по-умолчанию).

```
| systemctl kill <cgroup_name>
```

/var/run/

04. Linux Storage & RAM

[[MBR vs GPT]]

MBR vs GPT

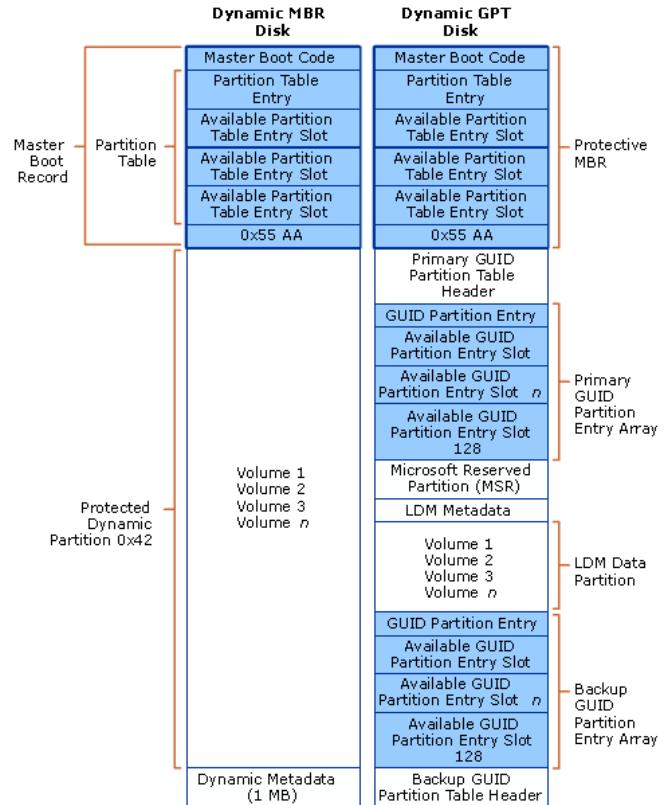
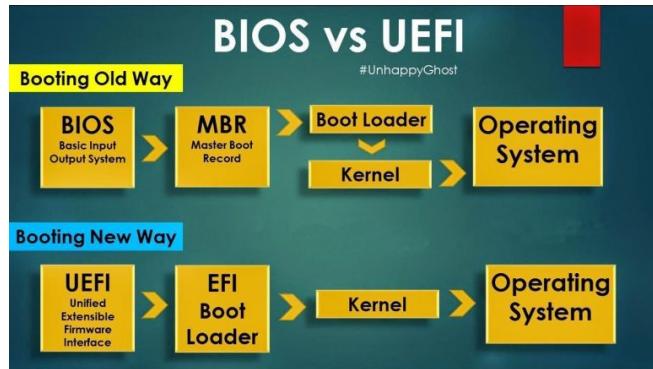
MBR (master boot record)

- up to 2 TiB ($2^{32} \times 2^9$ (512) bytes per sector).
- 4 основных раздела
- работает и с BIOS и с UEFI
- нет собственного мультизагрузчика
- fdisk, cfdisk - основные утилиты для разметки диска

GPT (GUID Partition Table)

- up to 8 ZiB (2^{64} sectors $\times 2^9$ (512) bytes per sector) Для лучшего представления это как 940 миллионов дисков по 10 ТБ каждый
- 2^{64} разделов возможно, но винда поддерживает до 128 разделов, линукс до 256 разделов
- работает только с UEFI
- есть собственный мультизагрузчик
- данные о разделах хранятся в 2x местах
- parted - основная утилита для разметки диска

BIOS and UEFI boot



parted (gpt разметка диска)

В parted нет отдельной команды для применения изменений, поэтому все изменения будут применяться налету, сразу после выполнения команд!

получаем справку в интерактивном режиме

```
| (parted) help
| (parted) help
```

Пример разбивки диска

выбираем диск, с которым будем работать

```
| # parted
```

выбираем нужный диск

```
| select /dev/sda
```

выбираем размерность в которой будем работать (s, B, kB, MB, GB, TB, compact, cyl, chs, %, kiB, MiB, GiB, TiB)

```
| (parted) unit TB
```

смотрим какие разделы имеем (будут отображены в выбранной размерности)

```
(parted) print
```

не знаю что это

```
| (parted) mklabel gpt
```

создадим раздел с названием "primary" размером 3TB и "secondary" размером 0.5TB

```
| (parted) mkpart primary 0TB 3TB  
| (parted) mkpart secondary 3TB 3.5TB
```

выходим и накатываем файловую систему на свеженарезанный диск

```
| (parted) quit  
# mkfs.xfs /dev/sda1
```

удалить разрез под номером 3

```
| (parted) rm 3
```

Примеры односторонних скриптов для разметки

Пример 1

Создать 1 раздел на весь диск

```
| parted -s -a optimal /dev/sdb mklabel gpt -- mkpart primary ext4 0 -1
```

Пример 2

создадим 5 разделов с именами "/" (500MiB), "/boot"(10GiB), "/usr/local"(10GiB), "/var"(5GiB), "/home(все что осталось, т.е. 49.5GiB)"
Важно чтоб после разделителя команд " " был пробел! В данном примере пробел находится в начале следующей строки

```
| sudo parted -s -a optimal /dev/sdb mklabel gpt -- mkpart /boot 0 500MiB\  
| mkpart / 500MiB 10.5GiB\  
| mkpart /usr/local 10.5GiB 25.5GiB\  
| mkpart /var 25.5GiB 30.5GiB\  
| mkpart /home 30.5GiB -1\  
| print
```

тоже самое но в одну строку

```
| sudo parted -s -a optimal /dev/sdb mklabel gpt -- mkpart /boot 0 500MiB mkpart / 500MiB  
| 10.5GiB mkpart /usr/local 10.5GiB 25.5GiB mkpart /var 25.5GiB 30.5GiB mkpart /home 30.5GiB -1  
| print
```

в результате видим вывод "print"

```
| Warning: The resulting partition is not properly aligned for best performance.  
| Model: ATA VBOX HARDDISK (scsi)  
| Disk /dev/sdb: 85.9GB  
| Sector size (logical/physical): 512B/512B  
| Partition Table: gpt  
| Disk Flags:  
  
| Number Start End Size File system Name Flags  
| 1 17.4kB 524MB 524MB xfs /boot  
| 2 524MB 11.3GB 10.8GB /  
| 3 11.3GB 27.4GB 16.1GB /usr/local  
| 4 27.4GB 32.7GB 5369MB /var  
| 5 32.7GB 85.9GB 53.1GB ext4 /home
```

или lsblk

```
$ lsblk  
sdb 8:16 0 80G 0 disk  
sdb1 8:17 0 500M 0 part  
sdb2 8:18 0 10G 0 part  
sdb3 8:19 0 15G 0 part  
sdb4 8:20 0 5G 0 part  
sdb5 8:21 0 49.5G 0 part
```

Вопросы

1. когда размечаю диск скриптом и указываю тип файловой системы, то его не видно на выводе "print"

```
sudo parted -s -a optimal /dev/sdb mklabel gpt -- mkpart /boot ext4 0 512\ print

Number  Start   End     Size    File system  Name   Flags
 1      17.4kB  512MB  512MB          /boot
```

2. когда переразмечаю диск, то файловая система не перетирается на диске и её видно в выводе "print"

3. как задать поле "flags" и для чего?

4. поле "system" будет видно только если там установлена ОС?

LVM

- Общие сведения о LVM
 - Документация
 - Структура (уровни построения) LVM
 - Последовательность организации LVM
 - Местонахождение логических томов в /dev
 - Получение информации о LVM
- Примеры манипуляций с LVM
 - Изменение размера логического тома LVM

Общие сведения о LVM

Документация

Управление разделами LVM (список команд)

Увеличение размера логических томов

Создание снимков

в "выпадайке" огромное количество полезной инфы по LVM. Наиболее полезное я перенесу сюда.

Структура (уровни построения) LVM

1. Физический том
2. Группа томов на одном или нескольких физических томах
3. Логические тома в группе томов

Последовательность организации LVM

1. Пометить тома для LVM

```
| # pvcreate /dev/sdX1
```

2. Создаем группу томов "G1"

```
| # vgcreate G1 /dev/sdX1 /dev/sdY3 ...
```

3. Создаем логические тома root и home (где 8G - размер тома, а G1 - название группы, в которой создаем том)

```
| # lvcreate -n root -L 8G G1
| # lvcreate -n home -L 10G G1
```

Местонахождение логических томов в /dev

Логические тома появляются тут:

```
|
```

```
/dev/mapper/G1-root  
/dev/mapper/G1-home
```

так же создаются ярлыки на них тут:

```
/dev/G1/root  
/dev/G1/home
```

Получение информации о LVM

```
# pvs      -      LVM  
# vgs      -  
# lvs      -
```

Примеры манипуляций с LVM

Изменение размера логического тома LVM

1. Если в группе томов неосталось свободного места, добавляем в нее любой свободный раздел жесткого диска (если место есть в группе томов, этот пункт пропускаем)

помечаем физ. том для LVM

```
# pvcreate /dev/sdZ1
```

добавляем помеченный физ. том в нашу существующую группу томов (тут G1 - это название группы)

```
# vgextend G1 /dev/sdZ1
```

2(a) Расширяем логический том (добавим к нему 35G. G1-home - это название группы и название логического тома)

```
# lvextend -r -L +35G /dev/mapper/G1-home
```

2(b) Уменьшаем логический том (отпилим от него 80G. G1-home - это название группы и название логического тома)

```
# lvreduce -r -L -80G /dev/mapper/G1-home
```

3. Переопределить размер файловой системы (если при изменение размера логического тома был использован ключ -г, то ресайз будет проведен сразу, и этот пункт не нужен)

```
# resize2fs /dev/mapper/G1-home
```

ext2, ext3, ext4

Сравнение

Файловая система	Максимальный размер файла	Максимальный размер тома	Журналируемая
ext2	16GiB - 2TiB	2TiB - 32TiB	нет
ext3	16GiB - 2TiB	2TiB - 32TiB	да
ext3	16GiB - 16TiB	1EiB	да

ext3

Журналируемая файловая система

Имеет три режима журналирования:

1. writeback: в журнал записываются только метаданные файловой системы, то есть информация о её изменениях. Не может гарантировать целостности данных.
2. ordered: то же, что и writeback, но запись данных в файл производится гарантированно до записи информации об изменении этого файла. Немного снижает производительность, также не может гарантировать целостности данных (хотя и увеличивает вероятность их сохранности при дописывании в конец существующего файла)
3. journal: полное журналирование как метаданных ФС, так и пользовательских данных. Самый медленный, но и самый безопасный

режим; может гарантировать целостность данных при хранении журнала на отдельном разделе (а лучше — на отдельном жёстком диске)

Указывается режим журнализации для mount

```
| mount /dev/hda6 /mnt/disc -t ext3 -o data=<
```

либо давить эту опцию в файле /etc/fstab

Из ext2 в ext3

переведем раздел из ext2 в ext3 без форматирования и резервного копирования

```
| tune2fs -f < >
```

Опция '-j' команды 'tune2fs' создает журнал ext3 на существующей ext2 файловой системе. После перевода файловой системы ext2 в ext3, нужно так же должны внести изменения в записи файла /etc/fstab

так же можно отключить fsck в /etc/fstab, т.к. целостность файловой системы гарантируется откатом в журнале.

Опции монтирования

- defaults – устанавливает стандартные параметры монтирования:
 - rw (чтение/запись),
 - uid (разрешается использование битов SUID и SGID),
 - dev (разрешается создавать файлы устройств символьного или блочного доступа),
 - exec (разрешается выполнение программ),
 - auto (разрешается команда mount -a),
 - nouser (монтирувать файловую систему может только суперпользователь),
 - async (асинхронный ввод-вывод);
- nodev – запрещается создавать файлы устройств символьного или блочного доступа;
- noexec – запрещается выполнять программы и сценарии;
- ro – доступ к файловой системе только для чтения;
- user – файловую систему могут монтировать рядовые пользователи. При этом автоматически включаются опции, которые можно переопределить явным образом:
 - noexec
 - nosuid
 - nodev
- nosuid – не разрешается выполнение программ с установленным битом SUID и/или SGID;
- noatime – не разрешается обновлять информацию о времени доступа к файлам и каталогам. Эта опция доступна в ядре версии от 2.2 и выше.

xfs

- Особенности файловой системы
- Ограничения
- Установка
- Копирование разделов
- Резервное копирование
 - Создание полной резервной копии
 - Создание инкрементальной резервной копии
 - Восстановление из полной резервной копии
 - Восстановление из инкрементальных резервных копий (кумулятивный)
 - Просмотр резервных копий в интерактивном режиме
- Заморозка файловой системы
- Восстановление файловой системы

Особенности файловой системы

- 64-битная
- Изменение размера «на лету» (только увеличение)
- Дефрагментация «на лету».
- Отложенное выделение места. При записи файла для него выделяется место в памяти, а на диске выделяется место только при

- записи файла на диск. Таким образом под файл оптимально выделяется место на диске, что уменьшает фрагментацию.
- Размещение в нескольких линейных областях (по умолчанию — 4 шт.) т. н. «allocation groups» (увеличивает производительность путём выравнивания активности запросов как к разным дискам на RAID-массивах типа «stripe», так и при асинхронном обращении к файловой системе на обычном диске.)
 - API ввода-вывода реального времени (для приложений жёсткого или мягкого реального времени, например, для работы с потоковым видео).
 - Инструменты резервного копирования и восстановления (xfsdump и xfsrestore).
 - «Индексные блоки» inode выделяются динамически (по мере надобности) и неиспользуемые inode могут освобождаться (высвобождая место для хранения данных).
 - Невозможно уменьшить размер существующей файловой системы.**
 - Восстановление удалённых файлов в XFS — очень сложный процесс («Raise Data Recovery for XFS» для ОС Windows.)**
 - Возможность потери данных во время записи при сбое питания, так как большое количество буферов данных хранится в памяти при том, что метаданные записываются в журнал (на диск) оперативно. Это характерно и для других файловых систем с журналированием метаданных.**

Ограничения

Максимальный размер файла	8 EiB (эксабайт 2^{60}) - 1 байт
Максимальная длина имени файла	255 байт
Максимальный размер тома	16 EiB

Установка

возможен понадобится доставить группу пакетов для управления файловой системой

```
| sudo yum install xfsprogs
| sudo yum install -y xfsdump
```

Копирование разделов

1. размонтируем раздел перед копированием

```
| umount /dev/vg01/lv01
```

2. скопируем содержимое тома lv01 сразу на lv02 и lv03 (ключом -L включем логирование копирования)

```
| xfs_copy -L /var/log/copy.log /dev/vg01/lv01 /dev/vg01/lv02 /dev/vg01/lv03
```

будет проведено единоразовое чтение и запись сразу на два раздела. Ну и все это будет задокументировано в логе.

Резервное копирование

Создание полной резервной копии

xfsdump

```
| sudo xfsdump -1 0 -f /vagrant/dump-L0 /
```

-1 0	(от 0 до 9) уровень бекапа. 0 - полный, 1-9 инкрементальный
-f /vagrant/dump-L0	путь к файлу, в котором будет храниться бекап
/	путь к смонтированному разделу, с которого делается бекап

Создание инкрементальной резервной копии

```
| sudo xfsdump -l 1 -f /vagrant/dump-L1 /
| sudo xfsdump -l 2 -f /vagrant/dump-L2 /
```

Восстановление из полной резервной копии

xfsrestore

```
-S
-L
-I
-r
-i
( ls, cd, add, delete, extract ..)
```

находим информацию по нужному бекапу

```
| xfsrestore -I
```

ресторимся

```
| xfsrestore -f /vagrant/dump-L0 -L L0 /mnt
```

Восстановление из инкрементальных резервных копий (кумулятивный)

Восстановится из полной резервной копии, потом по-очереди из всех инкрементных резервных копий (не пропуская ни одну)

```
| xfsrestore -f /vagrant/dump-L0 -L L0 -r /mnt
```

получим восстановленные файлы + каталог xfsrestorehousekeepingdir в корне (в этой папке хранятся данные об этой резервной копии) приклеиваем к ней поочередно наши инкрементальные бекапы

```
| xfsrestore -f /vagrant/dump-L1 -r /mnt
| xfsrestore -f /vagrant/dump-L2 -r /mnt
```

чтоб после окончания восстановления следует удалить каталог xfsrestorehousekeepingdir, иначе обычное восстановление не будет работать

Примечание:

Если не удалить все файлы перед восстановлением и не использовать кумулятивный режим восстановления, то будут накатаны файлы, которые есть в полном (или инкрементальном) бекапе, без учета изменений.
т.е. все что создавалось будет восстановлено, а то что удалялось - все равно останется не удаленным.

Просмотр резервных копий в интерактивном режиме

```
| xfsrestore -f /vagrant/dump-L0 -i /mnt
```

Заморозка файловой системы

При заморозке файловой системы, файлы можно прочесть, но все операции записи будут "висеть" до тех пор пока файловая система не будет разморожена

xfs_freeze -f /mnt	заморозить файловую систему, смонтированную в /mnt
xfs_freeze -u /mnt	разморозить

Восстановление файловой системы

Отмонтируем раздел и выполним восстановление с подробным выводом

```
| umount /dev/sda3  
| xfs_repair -v /dev/sda
```

NFS, CIFS-UTILS

NFS

Установка утилиты

как для сервера так и для клиента достаточно установить пакет nfs-utils
крайне желательно на всех серверах настроить сервис синхронизации времени ntp в противном случае могут возникнуть нежелательные задержки в работе nfs

```
| yum install nfs-utils
```

Сервер

Файлы

/etc/exports

```
| man exports
```

Этот файл (/etc/exports) содержит различные разделяемые каталоги и права на них на сервере.

Несколько примеров:

```
| /files *(ro,sync) # access to anyone  
| /files 192.168.0.2(rw,sync) # IP 192.168.0.2  
| /files 192.168.1.1/24(rw,sync) # 192.168.1.1 192.168.1.255
```

Чтобы изменения вступили в силу без перезапуска демона, выполните команду:

```
| exportfs -arv
```

Чтобы сделать разделённый NFS каталог открытым и с правом записи, можно использовать опцию all_squash в комбинации с опциями a nonuid и anongid.

Например, чтобы установить права для пользователя 'nobody' в группе 'nobody', можно сделать следующее:

```
| ; 192.168.0.100, rw gid 99  
| /files 192.168.0.100(rw,anonuid=99,anongid=99)
```

Это также означает, что nobody:nobody должен быть владельцем разделённой директории:

```
| # chown -R nobody:nobody /files
```

/etc/hosts.allow

разрешить доступ для IP 192.168.0.101

```
| nfsd: 192.168.0.101/255.255.255.255  
| rpcbind: 192.168.0.101/255.255.255.255  
| mountd: 192.168.0.101/255.255.255.255
```

разрешить доступ всем из подсети 192.168.0.0/24

```
| nfsd: 192.168.0.0/255.255.255.0  
| rpcbind: 192.168.0.0/255.255.255.0  
| mountd: 192.168.0.0/255.255.255.0
```

разрешить доступ для всех

```
| nfsd: ALL  
| rpcbind: ALL  
| mountd: ALL
```

Демоны

для старта NFS сервера надо запустить их

```
systemctl start rpcbind nfs-server  
systemctl enable rpcbind nfs-server
```

для старых систем (именно в этом порядке они всегда должны быть запущены)

```
# /etc/rc.d/rpcbind start      (: /etc/rc.d/portmap start)  
# /etc/rc.d/nfs-common start   (: /etc/rc.d/nfslock start)  
# /etc/rc.d/nfs-server start   (: /etc/rc.d/nfssd start)
```

Монтирование NFS шары

/etc/fstab

смонтирано на чтение и запись для всех пользователей, тип соединения <hard>, и оно может быть прервано <intr>

```
//192.168.1.182:/NFSShare1 /home/robert/share/    nfs  
username=nfsUser,password=nfsPassword,rw,hard,intr 0      0
```

CIFS-UTILS

Установка итилита

```
$ sudo yum install cifs-utils
```

Монтирование винтовой шары

/bin/bash

с запросом пароля

```
$ sudo mount -t cifs -o username=myUser //HOST_IP/SHARE_NAME /mnt/
```

без запроса пароля

```
$ sudo mount -t cifs -o username=myUser,password=myPassword //HOST_IP/SHARE_NAME /mnt/
```

/etc/fstab

монтирование винтовой шары под пользователем с ID=1001 и группой с ID=1001, права 770 как для папок так и для директорий, и кодировкой utf8

```
//192.168.1.181/windowsShare1 /home/robert/share/  cifs  
username=myWindowsUser,password=myWindowsPassword,iocharset=utf8,sec=ntlm,uid=1001,gid=1001,file_mode=0770,dir_mode=0770,noperm 0
```

монтируем винтовую шару с полным доступом для root:users и только для чтения для всех других (id пользователя не задано, поэтому будет использовано по-умолчанию, а это root)

```
//192.168.1.181/windowsShare1 /share/1/  cifs  
username=myWindowsUser,password=myWindowsPassword,iocharset=utf8,sec=ntlm,gid=100,file_mode=0774,dir_mode=0774,noperm 0 0
```

вариант монтирование с доменной авторизацией

```
//FILESERVER/SHARE$      /mnt/SHARE     cifs     username=domain\user,password=mypassword 0 0
```

доменная авторизация, без явного указания паролей

```
# cat /etc/fstab | grep SHARE  
//FILESERVER/SHARE$      /mnt/SHARE     cifs     credentials=/root/.smbcredentials 0 0
```

```
# cat /root/.smbcredentials  
username=domain\user  
password=mypassword  
  
# chmod 600 /root/.smbcredentials
```

Включение разных SMB версий

```
# cat /etc/fstab | grep SHARE  
//FILESERVER/SHARE$ /mnt/SHARE cifs  
uid=105,gid=107,dir_mode=0770,file_mode=0770,credentials=/root/.smbcredentials,vers=3.02 0 0  
  
PS C:> Get-SmbSession | select Dialect  
Dialect  
-----  
3.02
```

SWAP

Создать новый SWAP file

Создать 2Gb SWAP file

```
dd if=/dev/zero of=/swapfile bs=1M count=2048  
  
mkswap /swapfile  
swapon /swapfile
```

Убеждаемся, что swap-файл подключился правильно

```
swapon -s  
  
Filename Type Size Used Priority  
/dev/dm-1 partition 2097144 76252 -1
```

fstab

```
cat /etc/fstab | grep swap  
  
/swapfile none swap sw 0 0
```

Используется или нет

```
free -m
      total        used        free      shared    buffers     cached
Mem:       5852       5736        116          0       173     4917
-/+ buffers/cache:       644       5207
Swap:      2047         74      1973
```

I/O в реальном времени

```
vmstat 5

procs -----memory----- --swap-- -----io---- --system--
-----cpu-----
r b swpd   free   buff  cache   si   so   bi   bo   in   cs us sy id
wa st
 0 0 76252 121952 177596 5034588   0   0   5    7   0   0   1   1
97 1 0
 0 0 76252 118432 177600 5035644   0   0   0   18  315  219   0   0
100 0 0
 0 0 76252 118456 177604 5035652   0   0   0    6   91   59   0   0
100 0 0
```

Уровни свапинга

0-100 (дeфoлт 60). Чем выше значение, тем больше страниц попадает в свап.
10 - значит что если память будет занята на 90%, то только тогда начнет использоваться свап

Временно изменить уровень свапинга:

```
echo 10 > /proc/sys/vm/swappiness
```

Для постоянного изменения :

```
vim /etc/sysctl.conf

vm.swappiness = 10
```

RAM

Очистка кэша

Очистка кэша PageCache:

```
sync; echo 1 > /proc/sys/vm/drop_caches
```

Очистка inode и dentrie:

```
sync; echo 2 > /proc/sys/vm/drop_caches
```

Очистка inode и dentrie и PageCache:

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

можно добавить в крон...

Скорость очистки кэша

По умолчанию установлен параметр 100. Если его уменьшить ядро будет реже удалять страницы и это приведет к очень быстрому увеличению кэша. При нуле страницы вообще не будут удаляться. Если значение больше 100, размер кэша будет увеличиваться медленнее и неиспользуемые страницы будут сразу удаляться.

```
echo 1000 > /proc/sys/vm/vfs_cache_pressure
```

это очень сильно снижает производительность вашей системы, потому что вместо кэша данные будут читаться из диска.

05. Networking

iptables tutorial

пожалуй лучшее что попадалось:

<https://www.opennet.ru/docs/RUS/iptables/>

IP - utility

Конфигурация интерфейса

Добавить IP

Добавить IP 92.168.33.13 и маской /24 на поднятом интерфейсе eth1

опции можно сокращать!!! все эти команды имеют одинаковый эффект

```
# ip address add 192.168.33.13/24 dev  
eth1  
  
# ip addr add 192.168.33.13/24 dev  
eth1  
  
# ip a a 192.168.33.13/24 dev eth1
```

Удалить IP

|

```
# ip address delete 192.168.33.13/24  
dev eth1  
  
# ip addr del 192.168.33.13/24 dev  
eth1  
  
# ip a d 192.168.33.13/24 dev eth1
```

Включить интерфейс

```
# ip link set eth1 up  
  
# ip l s eth1 up
```

Отключить интерфейс

```
# ip link set eth1 down  
  
# ip l s eth1 down
```

Маршрутизация

Просмотр таблицы маршрутизации

```
# ip route show  
  
# ip r s
```

Добавить роут

```
ip route add 192.168.20.0/24 via  
192.168.33.10 dev eth1  
  
ip r a 192.168.20.0/24 via  
192.168.33.10 dev eth1
```

Удалить роут

```
ip route delete 192.168.20.0/24  
  
ip r d 192.168.20.0/24
```

Добавить дефолтный роут

```
# ip route add default via  
192.168.33.10  
  
# ip r a default via 192.168.33.10
```

Удалить дефолтный роут

```
# ip route delete default  
  
# ip r d default
```

iftop

Программа предоставляющая информацию об активных сетевых соединениях, скорость сетевой закачки/отдачи, является аналогом широко известного top по части использования сети: слушает трафик на указанном интерфейсе и отображает таблицу текущего использования по парам хостов.

Вывести весь трафик интерфейсе eth0:

```
| # iftop -i eth0
```

или если мы хотим увидеть прохождение трафика в байтах (по умолчанию iftop выводит статистику в битах).

```
| # iftop -i eth0 -B
```

Утилита iftop поддерживает, так называемый, pcap-filter синтаксис, используемый в пакетном фильтре и с помощью флага -f:

Трафик между локальным интерфейсом и хостом 8.8.8.8

```
| # iftop -i eth0 -f "dst 8.8.8.8"
```

Трафик идущий по ssh (можно использовать номер порта или же название протокола)

```
| # iftop -i eth0 -f "dst port 22"
```

Трафик по http

```
| # iftop -i eth0 -f "dst port http"
```

Трафик DNS

```
| # iftop -i eth0 -f "dst port domain"
```

Трафик ICMP

```
| # iftop -i eth0 -f "icmp"
```

Можно задавать сложные фильтры, в соответствии с синтаксисом pcap-filter:

Отображать трафик по ssh с хоста XXXXXX:

```
| # iftop -i eth0 -f "port ssh and host
```

Отображать весь трафик, кроме широковещательных запросов:

```
| # iftop -i eth0 -f "not ether host ff:ff:ff:ff:ff:ff"
```

Iperf

кроссплатформенная консольная клиент-серверная программа, предназначена для тестирования пропускной способности интернет канала между двумя хостами

В самом простом варианте для запуска iperf нужно воспользоваться 2 командами

```
| $iperf -s
```

Для запуска iperf сервера на одном хосте который будет выступать сервером т.е. принимать трафик.

```
| $ iperf -c <$IPERF_SERVER_IP>
```

По умолчанию используется TCP порт 5001, тестирование проходит в течение 10 секунд.

iperf и можно использовать с любым портом (ключ -p нужно указать как на сервера так и на клиенте) и для проверки скорости по TCP (по умолчанию) так и по UDP.

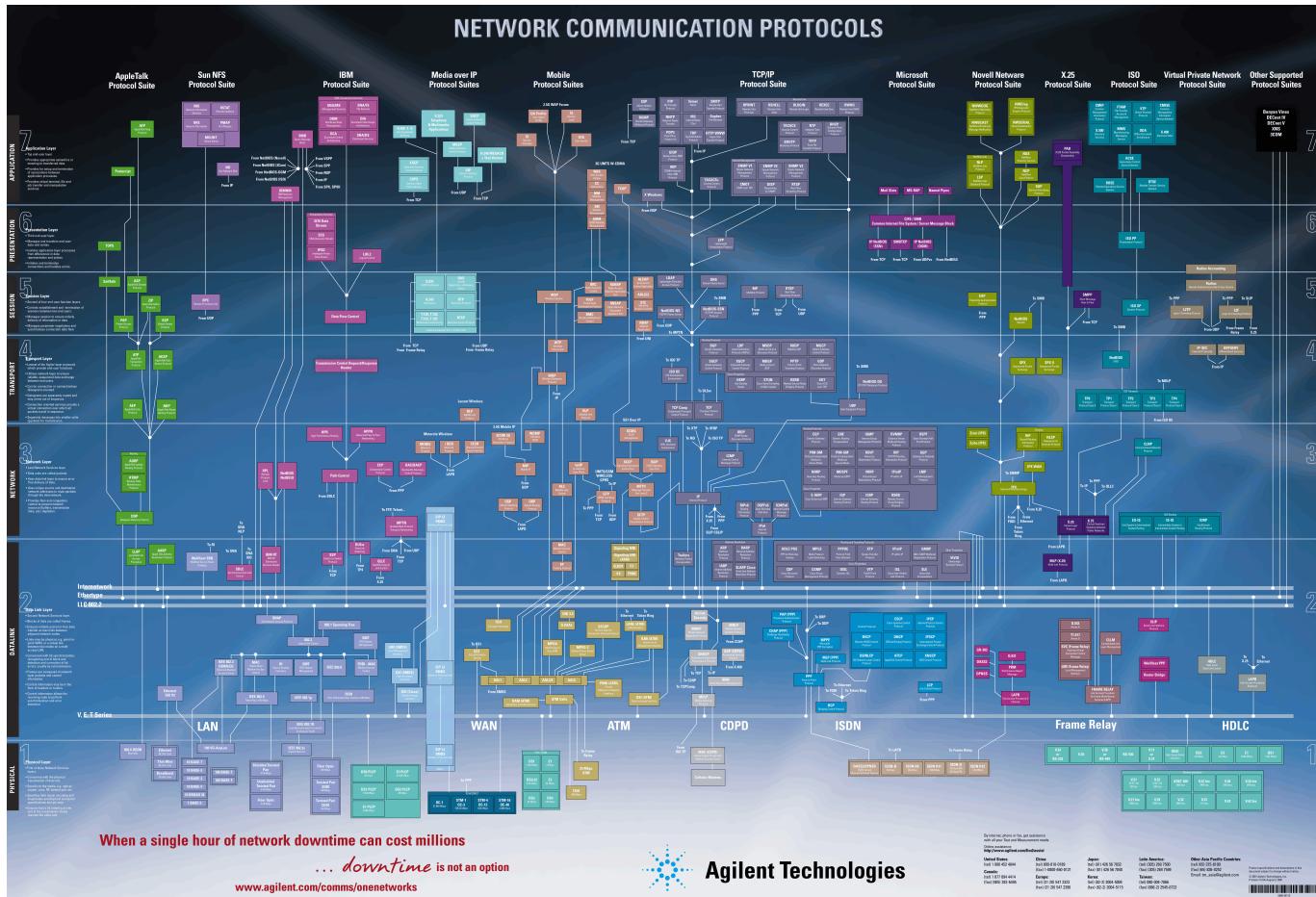
Так же имеет графический интерфейс (java app)

LINKS

[Полезная статья о захвате пакетов](#)

OSI Model (1-7 lvl)

Модель OSI				
Уровень (layer)		Тип данных (PDU)	Функции	Примеры
Host layers	7. Прикладной (application)	Данные	Доступ к сетевым службам	HTTP, FTP, SMTP
	6. Представительский (представления) (presentation)		Представление и шифрование данных	ASCII, EBCDIC, JPEG
	5. Сеансовый (session)		Управление сеансом связи	RPC, PAP
	4. Транспортный (transport)	Сегменты (segment)/ Дейтаграммы (datagram)	Прямая связь между конечными пунктами и надежность	TCP, UDP, SCTP
Media layers	3. Сетевой (network)	Пакеты (packet)	Определение маршрута и логическая адресация	IPv4, IPv6, IPsec, AppleTalk
	2. Канальный (data link)	Биты (bit)/ Кадры (frame)	Физическая адресация	PPP, IEEE 802.2, Ethernet, DSL, ARP, L2TP
	1. Физический (physical)	Биты (bit)	Работа со средой передачи, сигналами и двоичными данными	USB, витая пара, коаксиальный кабель, оптический кабель

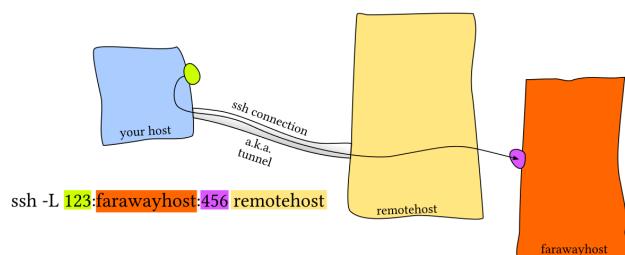
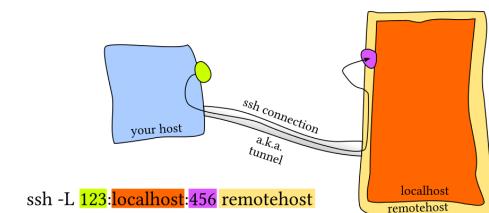


06. SSH advanced

SSH Port Forwarding

Local

```
ssh -L 123:localhost:446
      remotehost
ssh -L 123:farawayhost:456
      remotehost
```



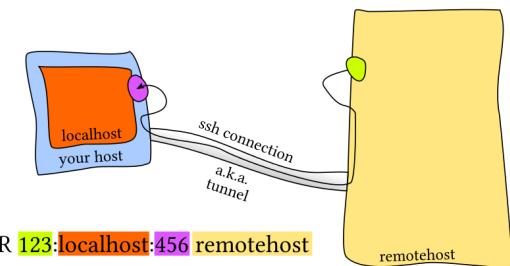
~/.ssh/config

```
Host remotehost
  HostName remotehost
  Port 22
  User username
  ForwardAgent yes
  LocalForward 123
localhost:456

Host remotehost
  HostName remotehost
  Port 22
  User username
  ForwardAgent yes
  LocalForward 123
farawayhost:456
```

Remote

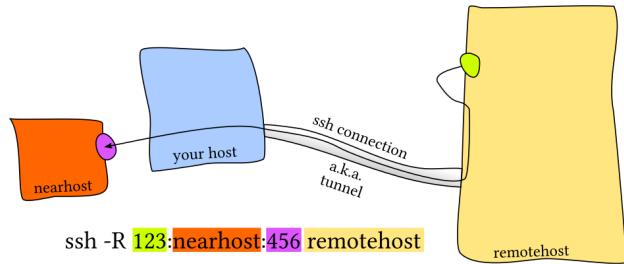
```
ssh -R 123:localhost:456
remotehost
ssh -R 123:nearhost:456
remotehost
```



~/.ssh/config

```
Host remotehost
  HostName remotehost
  Port 22
  User username
  ForwardAgent yes
  RemoteForward 123
localhost:456

Host remotehost
  HostName remotehost
  Port 22
  User username
  ForwardAgent yes
  RemoteForward 123
nearhost:456
```



Dynamic (SOCKS Proxy)

```
ssh -D 1080 remotehost  
ssh -D myipaddress:1080  
remotehost
```

~/.ssh/config

```
Host remotehost  
    HostName remotehost  
    Port 22  
    User username  
    ForwardAgent yes  
    DynamicForward 1080  
  
Host remotehost  
    HostName remotehost  
    Port 22  
    User username  
    ForwardAgent yes  
    DynamicForward  
    myipaddress:1080
```

APP > 1080:localhost > 22:remotehost:ANY (ANY port that app uses)

Config file

~/.ssh/config

```
### jump server ###  
Host host-jump  
    HostName host-jump.com  
    Port 22  
    User user123  
    ForwardAgent yes  
  
Host host1  
    HostName host1  
    ProxyJump host-jump  
    Port 22  
    User user123  
    ForwardAgent yes  
    IdentityFile ~/.ssh/id_rsa
```

Run ssh agent and add ssh keys

~/.bashrc

```
eval $(ssh-agent)  
ssh-add ~/.ssh/id_rsa  
ssh-add ~/.ssh/id_rsa2
```

SSH-agent forwarding to sudo (root user)

Кейс: 'sudo scp /file user@remout-host' не видит ssh-agent's keys

```

# visudo
Defaults>root      env_keep+=SSH_AUTH_SOCK

# ssh-agent bash
# ssh-add

```

07. Bash scripting

awk

Эмуляция GREP

```

$ awk '/^root/ {print} /etc/passwd
root:x:0:0:root:/root:/bin/bash

```

/^root/	находим строку начинающуюся на < root >
{print}	выполняем команду < print > для найденной строки
/etc/passwd	файл, с которым работаем

Эмуляция HEAD

```

$ awk '{print}NR==10{exit}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/usr/bin/nologin
daemon:x:2:2:daemon:/:/usr/bin/nologin
mail:x:8:12:mail:/var/spool/mail:/usr/bin/nologin
ftp:x:14:11:ftp:/srv/ftp:/usr/bin/nologin
http:x:33:33:http:/srv/http:/usr/bin/nologin
uuidd:x:68:68:uuidd:/:/usr/bin/nologin
dbus:x:81:81:dbus:/:/usr/bin/nologin
nobody:x:99:99:nobody:/:/usr/bin/nologin
systemd-journal-gateway:x:191:191:systemd-journal-gateway:/:/usr/bin/nologin

```

{print}	делаем вывод на экран всего
NR==10	десятая строка с момента вывода
{exit}	делаем выход (сразу после 10 строки)
/etc/passwd	файл, с которым работаем

Эмуляция wc -l (подсчет строк)

```
$ awk 'END{print NR}' /etc/passwd  
35
```

END	сработает, когда будет достигнут конец файла
print NR	выводим на экран < NR > (количество строк)
/etc/passwd	файл с которым работаем

Работа со столбцами

вывод конкретного столбца

```
$ ll / | awk '/^l/{print $8}'  
bin  
lib  
lib64  
sbin
```

ll /	выводим содержимое каталога и перенаправляем его в < awk >
/^l/	находим строки, начинающиеся на < l >, т.е. ярлыки
{print \$8}	выводим 8й столбец (тот что с именем файла в моем случае)

меняем символ, разделения столбцов

```
$ awk -F: '$1=="root"{print $7}' /etc/passwd  
/bin/bash
```

-F:	назначаем символ разделитель строк < : > (по умолчанию это пробел или табуляция)
\$1=="root"	для той строки, где < 1й > столбец имеет значение < root >
{print \$7}	вывести < 7й > столбец
/etc/passwd	файл с которым работаем

sed

Использовать как grep

```
sed -n /KEY/p /my/file
```

-n	не выводить все обрабатываемые строки
/KEY/	искомое слово
p	отобразить строки, совпадающую с шаблоном
/my/file	путь к файлу

Использовать как head

```
sed 10q /my/file
```

10q выполнить в 10 строке quit

редактировать файл с созданием резервной копии

(оригинал будет изменен, а резервная копия будет находиться рядом с оригиналом)

```
sed -i.orig s/^alias/#alias/ /my/file
```

-i[SUFFIX]	будет создана резервная копия с окончанием [SUFFIX]
s/XXX/YYY/	найти XXX и заменить на YYY
^	начало строки

тоже самое, но в другом порядке

```
sed -i.orig /^alias/s/^/#/ /my/file
```

-i.orig	сохраняем резервную копию и тем же именем + [.orig] окончание
/^alias/	ищем строку, начинающуюся со слова alias
s///	команда замены. В нашем случае меняем ^ (начало строки) на #

удалить все строки начинающиеся с комментария < #>

```
sed /^#/d /my/file
```

<code>^#</code>	начало строки
<code>d</code>	команда удаления строки, совпадшей с тем что в <code>//</code>

меняем `< / >` на `< @ >` (любой другой символ, который встречается первым после команды `< s >`)

```
sed /PATH=/s@:@:/usr/local/bin:@ /my/file
```

<code>/PATH=</code>	меняем в строке со словом <code>< PATH= ></code>
<code>s@:@:/usr/local/bin:@</code>	первое найденное <code>< : ></code> на <code>< :/usr/local/bin: ></code>
<code>/my/file</code>	путь к файлу

Несколько строк сценария (последовательность действий)

перед каждой строкой надо ставить ключь `< -e >`

```
sed -e /^#/d -e /^$/d /my/file
```

<code>-e /^#/d</code>	1ой строкой находим все строки начинающиеся с <code># ></code> и удаляем их командой <code>< d ></code>
<code>-e /^\$/d</code>	2ой строкой находим все пустые строки и тоже их удаляем
<code>/my/file</code>	путь к файлу

Добавляем строки после совпадения с шаблоном

```
sed -i '/searchText/a\Line1\nLine2' /my/file
```

<code>-i</code>	сразу вносим изменения в файл
<code>searchText</code>	искомый текст (именно после него будет осуществлена вставка)
<code>a\</code>	оператор добавления строк
<code>Line1</code>	текст первой добавляемой строки
<code>\n</code>	перенос строки
<code>Line2</code>	текст второй добавляемой строки
<code>/my/file</code>	путь к файлу

08. Internet services

DNS - BIND9

- Команды для работы с DNS
 - nsupdate
 - rndc
 - обновление одной зоны
 - обновление всех зон сервера
- КЕЙС 1
 - NS1.EPAM
 - NS2.EPAM (NS2.OLEG.EPAM, NS2.SERGEY.EPAM)
 - NS1.OLEG.EPAM
 - NS1.SERGEY.EPAM

Команды для работы с DNS

nsupdate

добавление / удаление записей "налету"

A

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
> show
> send
> quit
```

PTR

```
# ns update
> update delete 100.0.168.192.in-addr.arpa PTR
> update add 100.0.168.192.in-addr.arpa 86400 PTR
> show
> send
> quit
```

rndc

обновление зоны, после внесения изменений (вручную) в файл зоны

обновление одной зоны

zone refresh

```
# rndc reload oleg.epam.
zone refresh queued
```

обновление всех зон сервера

server reload

```
# rndc reload
server reload successful
```

КЕЙС 1

OS	CentOS 7
config: Main	/etc/named.conf
config: Options	/etc/named/named.options
config: Master Zone	/etc/named/named.master.zones
config: Slave Zone	/etc/named/named.slave.zones
config: Forward Zone	/etc/named/named.forward.zones
folder: Main Zones	/var/named/masters/
folder: Slave Zones	/var/named/slaves/

Конфигурационные файлы для удобства были разделены и добавлены в основной файл инклюдами

NS1.EPAM

VIEW LAN					
IP	Host Name	Master Zones	Slave Zones	Forward Zones	Recursion
10.10.10.10	ns1.artem.epam.	artem.epam. file: artem.epam.zone.local	-	-	127.0.0.1
VIEW WAN					
IP	Host Name	Master Zones	Slave Zones	Forward Zones	Recursion
192.168.3.10	ns1.epam.	epam. file: epam.zone	-	oleg.epam.	no
	ns1.artem.epam	artem.epam. file: artem.epam.zone		sergey.epam.	

NS2.EPAM (NS2.OLEG.EPAM, NS2.SERGEY.EPAM)

VIEW LAN					
IP	Host Name	Master Zones	Slave Zones	Forward Zones	Recursion
10.10.10.20	ns2.artem.epam.	-	artem.epam. file: artem.epam.zone.local		
192.168.3.20	ns2.oleg.epam.		oleg.epam. file: slaves/oleg.epam.zone		
192.168.3.20	ns2.sergey.epam.		sergey.epam. file: slaves/sergey.epam.zone		
192.168.3.20	ns2.artem.epam.		artem.epam. file: artem.epam.zone		

OS	CentOS 7
config: Main	/etc/named.conf
config: Zones	/etc/named/named.conf.zones
config: Options	/etc/named/named.conf.options
folder: Zones' files	/var/named/

```

/etc/named.conf

logging {
    channel default_debug
    {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.root.key";

include "/etc/named/named.conf.options";
include "/etc/named/named.conf.zones";
;
```

IP	Host Names	Master Zones	Slave Zones	Forward Zones	Recursion
192.168.3.20	ns2.epam.	-	epam. file: slaves/epam.zone	-	127.0.0.1 192.168.3.110 192.168.3.120
192.168.3.20	ns2.oleg.epam.		oleg.epam. file: slaves/oleg.epam.zone		
192.168.3.20	ns2.sergey.epam.		sergey.epam. file: slaves/sergey.epam.zone		
192.168.3.20	ns2.artem.epam.		artem.epam. file: artem.epam.zone		

/etc/named/named.conf.options

```
options {
    listen-on port 53 {
        127.0.0.1; 192.168.33.10;
        10.10.10.10; };
    directory
        "/var/named";
    dump-file
        "/var/named/data/cache_dump.db";
    statistics-file
        "/var/named/data/named_stats.txt";
    memstatistics-file
        "/var/named/data/named_mem_stats.txt";
    dnssec-enable yes;
    dnssec-validation yes;
    bindkeys-file
        "/etc/named.iscdlv.key";
    managed-keys-directory
        "/var/named/dynamic";
    pid-file
        "/run/named/named.pid";
    session-keyfile
        "/run/named/session.key";
    allow-query     { any; };
};
```

/etc/named/named.conf.zones

```
acl "allowed-lan-hosts" {
    127.0.0.1;
    10.10.10.0/24;
};

acl "disabled-lan-hosts" {
#    !10.10.10.1/32;
};

###  VIEW FROM THE LOCALNET
### 
view LAN {

    match-clients {
        allowed-lan-hosts;
        disabled-lan-hosts; };
    allow-recursion {
        allowed-lan-hosts;
        disabled-lan-hosts; };
```



```
192.168.33.20; };
    notify yes;
};

zone "oleg.epam." IN {
    type forward;
    forwarders {
192.168.33.110; };
    forward only;
};

zone "sergey.epam." IN {
    type forward;
    forwarders {
192.168.33.120; };
    forward only;
};

# my internal zones
zone "artem.epam." IN {
    type master;
    file
"artem.epam.zone.local";
};

};

###  VIEW FROM THE INTERNET
###
view WAN {
match-clients { any; };
recursion no;

# my zones
zone "epam." IN {
    type master;
    file "epam.zone";
    allow-transfer {
192.168.33.20; };
    notify yes;
};

zone "oleg.epam." IN {
    type forward;
    forwarders {
192.168.33.110; };
    forward only;
};

zone "sergey.epam." IN {
    type forward;
    forwarders {
192.168.33.120; };
    forward only;
};

zone "artem.epam." IN {
    type master;
```

```
    allow-transfer {  
192.168.33.20; };  
    file "artem.epam.zone";  
    notify yes;  
};
```

```
};
```

```
/var/named/epam.zone
```

```
@  
1D IN SOA ns1.epam.  
dnsmaster.epam. (  
  
54 ; serial (yyyymmdd##)  
  
3H ; refresh  
  
15M ; retry  
  
1W ; expiry  
  
1D ) ; minimum ttl  
  
@  
1D IN NS ns1.epam.  
@  
1D IN NS ns2.epam.  
  
@  
1D IN A 192.168.33.10  
ns1  
1D IN A 192.168.33.10  
ns2  
1D IN A 192.168.33.20  
  
oleg.epam. IN NS  
ns1.oleg.epam.  
ns1.oleg.epam. IN A  
192.168.33.110  
ns2.oleg.epam. IN A  
192.168.33.20  
  
sergey.epam. IN NS  
ns1.sergey.epam.  
ns1.sergey.epam. IN A  
192.168.33.120  
ns2.sergey.epam. IN A  
192.168.33.20
```

```
/var/named/artem.epam.zone
```

```
@  
1D IN SOA ns1.artem.epam.  
artem.artem.epam. (  
  
    54      ; serial (yyyymmdd##)  
  
    3H      ; refresh  
  
    15M     ; retry  
  
    1W      ; expiry  
  
    1D )   ; minimum ttl  
  
@  
1D IN NS  
ns1.artem.epam.  
@  
1D IN NS  
ns2.artem.epam.  
  
@  
1D IN A      192.168.33.10  
ns1  
1D IN A      192.168.33.10  
ns2  
1D IN A      192.168.33.20  
www  
1D IN A  
192.168.33.100
```

```
/var/named/artem.epam.zone.local
```

```
@  
1D IN SOA ns1.artem.epam.  
artem.artem.epam. (  
  
    57      ; serial (yyyymmdd##)  
  
    3H      ; refresh  
  
    15M     ; retry  
  
    1W      ; expiry  
  
    1D )   ; minimum ttl  
  
@  
1D IN NS  
ns1.artem.epam.  
@  
1D IN NS  
ns2.artem.epam.  
  
@  
1D IN A          10.10.10.10  
ns1  
1D IN A          10.10.10.10  
ns2  
1D IN A          10.10.10.20  
www  
1D IN A          10.10.10.100
```

NS1.OLEG.EPAM

IP	Host Name	Master Zones	Slave Zones	Forward Zones	Recursion
192.168.3.110	ns1.oleg.epam.	oleg.epam.	-	-	127.0.0.1

NS1.SERGEY.EPAM

IP	Host Name	Master Zones	Slave Zones	Forward Zones	Recursion
192.168.3.120	ns1.sergey.epam.	sergey.epam.	-	-	127.0.0.1

09. Linux Java stack

\$JAVA_HOME

Задаем переменную \$JAVA_HOME

```
[lime@f2 ~]$ export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which java))))  
  
[lime@f2 ~]$ export JAVA_HOME=$(readlink -f /usr/bin/javac | sed "s:/bin/javac::")
```

10. Virtualization and Containers

Docker

Installation

```
# yum install docker  
# systemctl enable docker  
# systemctl start docker
```

проверяем что все работает как надо

Run docker

```
# docker run hello-world  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
...
```

Search for Images

```
$ sudo docker search linux
```

Download an image

```
$ sudo docker pull ubuntu
```

List of images on this host

```
$ sudo docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED
ubuntu	latest	d355ed3537e9	2 weeks ago

Запуск

```
bash in docker
# docker run -it ubuntu /bin/bash
root@cd8d8016ab9c:/#
```

при этом если выйти из шела докера, то он закроется

Список контейнеров

```
Список активных контейнеров
# docker ps

CONTAINER ID        IMAGE       COMMAND      CREATED
STATUS              PORTS
dae88b8a1978        ubuntu      "/bin/bash"
Up 7 seconds        9 seconds ago
blissful_montalcini
```

```
Список всех контейнеров
# docker ps -a

CONTAINER ID        IMAGE       COMMAND      CREATED
STATUS              PORTS
daa88b8a1978        ubuntu      "/bin/bash"
Up 2 minutes         2 minutes ago
blissful_montalcini
717e6394e966        d355ed3537e9  "bash"
Exited (130) 6 minutes ago   6 minutes ago
eager_jennings
602782fdb874        d355ed3537e9  "bash"
Exited (130) 6 minutes ago   6 minutes ago
elated_volhard
```

```
Список недавно созданных контейнеров
# docker ps -l

CONTAINER ID        IMAGE       COMMAND      CREATED
STATUS              PORTS
dae88b8a1978        ubuntu      "/bin/bash"
Up 3 minutes         3 minutes ago
blissful_montalcini
```

Остановка контейнера

Остановить контейнер

```
# docker stop container-id
```

```
# docker stop dae88b8a1978  
dae88b8a1978
```

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	

Docker. Changes

Подключиться к уже запущенному контейнеру

Docker exec

```
| docker exec -it <CONTAINER_ID> bash
```

Docker. Common

Images

Provisionning

```
docker search <NAME>      #  
  
docker pull image/name:label      #  
docker pull registryhost:registryport/image/name:label #  
  
docker push registryhost:registryport/image/name  #      (      tag )
```

Lifecycle

```
docker images #  
docker import #      tarball(?)  
docker images #      Dockerfile  
docker commit #      ( , )  
docker rmi  #  
docker load  #      .tar  STDIN ( )  
docker save  #      .tar  STDOUT ( , , )
```

Info

```
docker history <IMAGE>      #
docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG] #
```

```
# docker run -d --name squid ubuntu-squid3 squid -N
# docker kill <CONTAINER_ID>
# docker start squid
```

Docker. Diagnostic Commands

- Список контейнеров
- Список процессов контейнера
- Нагрузка

Список контейнеров

Active				
# docker ps	CONTAINER ID	IMAGE	COMMAND	CREATED
	STATUS	PORTS	NAMES	
	c50d5f33c8c4	ubuntu-squid3	"squid -N"	10 minutes
ago	Up 10 minutes	0.0.0.0:9992->3128/tcp	unruffled_mirzakhani	
	e12411ed42d3	ubuntu-squid3	"squid -N"	16 minutes
ago	Up 16 minutes	0.0.0.0:9994->3128/tcp	vigorous_davinci	

Last

Last				
# docker ps -l	CONTAINER ID	IMAGE	COMMAND	CREATED
	STATUS	PORTS	NAMES	
	c50d5f33c8c4	ubuntu-squid3	"squid -N"	10 minutes
ago	Up 10 minutes	0.0.0.0:9992->3128/tcp	unruffled_mirzakhani	

All

```
# docker ps -a

CONTAINER ID        IMAGE               COMMAND             CREATED            NAMES
STATUS              PORTS              NAMES
c50d5f33c8c4        ubuntu-squid3      "squid -N"         10 minutes       unruffled_mirzakhani
ago     Up 10 minutes           0.0.0.0:9992->3128/tcp
e12411ed42d3        ubuntu-squid3      "squid -N"         16 minutes       vigorous_davinci
ago     Up 16 minutes           0.0.0.0:9994->3128/tcp
8f651618123c        ubuntu-squid3      "squid -N"         About an hour ago
hour ago  Exited (137) About an hour ago
squid
4f8a378cafe4        ubuntu-squid3      "squid -N"         About an hour ago
hour ago  Exited (137) About an hour ago
compassionate_euclid
...
...
```

Список процессов контейнера

Show "top" of the container with id <CONTAINER ID>

```
# docker top <CONTAINER_ID>

UID          PID    PPID   C
STIME        TTY    TIME   CMD
13           28975  28959  0
23:21        pts/3   00:00:00 squid -N
13           29046  28975  0
23:21        ?      00:00:00
(logfile-daemon) /var/log/squid/access.log
```

Нагрузка

Load of one container

```
# docker stats <CONTAINER_ID>

CONTAINER          CPU %           MEM USAGE / LIMIT      MEM %
NET I/O           BLOCK I/O        PIDS
e12411ed42d3      0.00%           152MiB / 11.2GiB    1.32%
4.64kB / 0B        0B / 16.9kB      2
```

Load of all containers

```
# docker stats
```

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %
NET I/O	BLOCK I/O	PIDS	
c50d5f33c8c4	0.00%	152MiB / 11.2GiB	1.33%
4.31kB / 0B	0B / 21kB	2	
e12411ed42d3	0.00%	152MiB / 11.2GiB	1.32%
4.64kB / 0B	0B / 16.9kB	2	

SDLC approaches

SDLC (Software development lifecycle)

01. Software lifecycle

- 1.
- 2.
- 3.
4. ,
- 5.
- 6.

02. Software version control systems

GIT

Основные команды

Ветки

git branch	посмотреть список веток
git branch <new_branch>	создать новую ветку <new_branch>
git checkout <branch>	переключиться на ветку <branch>
git checkout -b <new_branch>	создать новую ветку <new_branch> и переключиться в неё
git branch -d <branch>	удалить локальную ветку <branch>
git push origin :<branch>	удалить ветку <branch> на удаленном сервере

Install

```
| yum install git bash-completion
```

bash-completion - нужен чтобы команды нормально дотягивались :)

Подсветка вывода

```
git config --global  
color.diff true  
git config --global  
color.status true  
git config --global  
color.branch true  
git config --global  
color.interactive true
```

Доступно 4 типа настройки:
false - выключена

Коммиты

<code>git status</code>	просмотреть какие файлы изменены с момента последнего коммита
<code>git add <file></code>	добавить файл <code><file></code> к будущему коммиту
<code>git add -A</code>	добавить все измененные файлы к будущему коммиту
<code>git commit -m "some_comment"</code>	коммитнуть добавленные изменения с односрочным комментарием "some_comment"

true - включена, если вывод не перенаправлен
always - включена всегда

Логи

Формат вывода

<code>git log -p</code>	<code>-p</code> дельта. показать что именно было изменено в коммитах
<code>git log -p -1 --word-diff</code>	<code>--word-diff</code> покажет изменения на уровне слова (не будет писать в две строки что удалено, и что добавлено)
<code>git log --oneline</code>	<code>--oneline</code> покажет коммиты по одной строке на каждый
<code>git log --oneline --stat</code>	<code>--stat</code> покажет стату по изменениям (48 +++++--)
<code>git log --pretty=oneline</code>	<code>--pretty=</code> (доступны параметры oneline, short, full, fuller, format) меняет подробность вывода (см. таблицу с параметрами format)
<code>git log --pretty=format:"%h %an, %ar : %s"</code>	<code>--pretty=format:""</code> задаем формат вывода лога вручную (см. таблицу ниже)
<code>git log --pretty=format:"%h %s" --graph</code> <code>git log --oneline --graph</code>	<code>--graph</code> покажет график ветвлений
<code>git log --oneline --name-only</code>	<code>--name-only</code> покажет список изменённых файлов после информации о коммите.

Фильтрация

<code>git diff -- <file></code>	<code>diff -- <file></code> покажет что было изменено с момента последнего коммита
<code>git log -2</code>	<code>-2</code> (доступна любая цифра) показать только 2 последних коммита
<code>git log --since=2.weeks</code>	<code>--since=</code> <code>--after=</code> (можно указать точную дату 2 017-12-29 или использовать относительные 2 years 1 day 3 minutes ago) покажет список изменений за последние 2 недели
<code>git log --until=2017-01-30</code>	<code>--until=</code> <code>--before=</code> Ограничить коммиты теми, которые сделаны до указанной даты
<code>git log --author=<author></code>	<code>--author</code> фильтровать по автору
<code>git log --committer=<committer></code>	<code>--committer</code> фильтровать по коммитеру
<code>git log --grep=<some_text></code>	<code>--grep</code> позволяет искать по ключевым словам в сообщении
<code>git log --author=<artem> --grep=<some_text> --all-match</code>	<code>--all-match</code> найти коммиты, которые удовлетворяют первому И второму критерию

<code>git log --oneline --name-status</code>	--name-status покажет список изменённых файлов вместе с информацией о добавлении/изменении/удалении.	<code>git log -q -10 --<path></code>	-- фильтр по пути к файлу или директории. указывается в самом конце всех опций
<code>git log --abbrev-commit</code>	--abbrev-commit покажет только первые несколько символов контрольной суммы SHA-1 вместо всех 40.		
<code>git log --relative-date</code>	--relative-date покажет дату в относительном формате (например, "2 weeks ago") вместо полной даты.		

Наиболее полезные параметры формата --pretty=format

Параметр	Описание выводимых данных
%H	Хеш коммита
%h	Сокращённый хеш коммита
%T	Хеш дерева
%t	Сокращённый хеш дерева
%P	Хеши родительских коммитов
%p	Сокращённые хеши родительских коммитов
%an	Имя автора
%ae	Электронная почта автора
%ad	Дата автора (формат соответствует параметру --date=)
%ar	Дата автора, относительная (пр. "2 мес. назад")
%cn	Имя коммитера
%ce	Электронная почта коммитера
%cd	Дата коммитера
%cr	Дата коммитера, относительная
%s	Комментарий

Project related CI/CD tools

01. CM tools

Configuration Management Tools

Ansible

Directory Layout

Create Directory Layout

```
$ mkdir -p
ansible/{group_vars,host_vars,library,module_utils,filter_plugins,roles/{common/{tasks,handlers,templates,files,vars,defaults,meta,library,module_utils,lookup_plugins},icinga-master,icinga-satellite}}

$ touch ansible/production ansible/staging ansible/site.yml
ansible/icinga-master.ymk ansible/icinga-satellite.yml
ansible/roles/common/{tasks,handlers,vars,defaults,meta}/main.yml
ansible/roles/common/templates/ntp.conf.j2
ansible/roles/common/files/bar.txt ansible/roles/common/files/foo.sh
ansible/group_vars/group1 ansible/group_vars/group2
ansible/host_vars/hostname1 ansible/host_vars/hostname2

# $ tree -FCc ansible
$ tree -F ansible/
ansible/
    filter_plugins/      # if any custom filter plugins, put them here
    (optional)
    group_vars/
        group1      # here we assign variables to particular groups
        group2
    host_vars/
        hostname1    # if systems need specific variables, put them here
        hostname2
    icinga-master.ymk    # playbook for icinga-master tier
    icinga-satellite.yml
    library/      # if any custom modules, put them here (optional)
    module_utils/    # if any custom module_utils to support modules, put them
    here (optional)
    production      # inventory file for production servers
    roles/
        common/      # this hierarchy represents a "role"
        defaults/
            main.yml  # <-- default lower priority variables for this role
        files/
            bar.txt   # <-- files for use with the copy resource
            foo.sh    # <-- script files for use with the script resource
        handlers/
            main.yml  # <-- handlers file
        library/    ### roles can also include custom modules
        lookup_plugins/  ### or other types of plugins, like lookup in this
```

```
case
  meta/
    main.yml  # <-- role dependencies
  module_utils/  ### roles can also include custom module_utils
  tasks/
    main.yml  # <-- tasks file can include smaller files if
warranted
  templates/  # <-- files for use with the template resource
    ntp.conf.j2  # ----- templates end in .j2
  vars/
    main.yml  # <-- variables associated with this role
 icinga-master/  # same kind of structure as "common" was above, done
for the "icinga-master" role
```

```
icinga-satellite/
site.yml      # master playbook
staging       # inventory file for staging environment
```

Puppet

Условия

```
CentOS7      #

puppet.example.com    # puppet server
test1.example.com    #

/opt/puppetlabs/bin/puppet  # PATH , puppet
```

Install

Puppet Server. Install

Install ntpd

```
timedatectl list-timezones
sudo timedatectl set-timezone
Europe/Kiev

yum -y install ntp
ntpdate pool.ntp.org
systemctl restart ntpd &&
systemctl enable ntpd
```

Install puppet server

```
rpm -Uvh
https://yum.puppetlabs.com/pu
ppet5/puppet5-release-el-7.no
arch.rpm
yum -y install puppetserver
```

Настройка использования памяти

Puppet Agent. Install

Install ntpd

```
timedatectl list-timezones
sudo timedatectl set-timezone
Europe/Kiev

yum -y install ntp
ntpdate pool.ntp.org
systemctl restart ntpd &&
systemctl enable ntpd
```

Install puppet agent

```
rpm -Uvh
https://yum.puppetlabs.com/pu
ppet5/puppet5-release-el-7.no
arch.rpm
yum -y install puppet-agent
```

Puppet agent configuration

/etc/sysconfig/puppetserver

```
# -Xms3g -  
# -Xmx3g - ( )  
  
JAVA_ARGS="-Xms3g -Xmx3g"
```

Start puppet server

```
systemctl start puppetserver  
systemctl enable puppetserver
```

/etc/puppetlabs/puppet/puppet.conf

```
[main]  
certname = test1.example.com  
server = puppet  
environment = production  
runinterval = 1h
```

Puppet agent start

```
/opt/puppetlabs/bin/puppet  
resource service puppet  
ensure=running enable=true  
  
##  
systemctl restart puppet &&  
systemctl enable puppet
```

на этом этапе желательно уже иметь подписанный сертификат агента на стороне сервера
Puppet agent run

```
/opt/puppetlabs/bin/puppet  
agent --test #  
puppet agent -t --noop #  
  
puppet agent -t --debug  
#
```

Certificates

On puppet server

Удалить и выпустить новый сертификат

Puppet Server

```
puppet cert clean  
test1.example.com
```

Puppet Agent

```
find  
/etc/puppetlabs/puppet/ssl/  
-type f -delete  
puppet agent -t
```

Puppet Server

```
puppet cert list      #
puppet cert list
test1.example.com  #
test1.example.com
puppet cert list --all      #
```

```
puppet cert sign
test1.example.com  #
puppet cert sign --all      #
```

```
puppet cert clean
test1.example.com  #
test1.example.com
puppet ca destroy
test1.example.com  #      (
)
```

```
puppet cert list
test1.example.com  #
puppet cert sign
test1.example.com
```

Facts

```
facter -p
```

PCB. File

рекурсивное назначение прав

```
.pp
file {
  $folders:
    ensure      => 'directory',
    owner       => 'apache',
    group       => 'apache',
    mode        => '0755',
    recurse     => true,
    recurselimit => 1
}
```

создание файлов, темплитов, ссылок и дерикторий через hiera переменные

.yaml

```
profile::some_profile::folders:
  - '/data'
  - '/data/folder1'
  - '/data/folder2'

profile::some_profile::templates:
  - file: '/data/folder1/file.conf'
    erb:   'data/folder1/file.conf.erb'

profile::some_profile::files:
  - file:   '/data/folder1/file1'
    source: 'file1'

profile::some_profile::links:
  - link:  '/data/link1'
    target: '/data/folder1'
  - link:  '/data/link2'
    target: '/data/folder1/file.conf'
```

.pp

```
class profile::some_profile{
    $folders      = hiera('profile::some_profile::folders'),
    $templates    = hiera('profile::some_profile::templates'),
    $files        = hiera('profile::some_profile::files'),
    $links        = hiera('profile::some_profile::links'),
}

{
    $folders.each |$folders| {
        file {
            $folders:
                ensure      => 'directory',
                owner       => 'apache',
                group      => 'apache',
                mode        => '0755',
        }
    }
    $templates.each |$template| {
        file {"${template['file']}":
            ensure  => 'present',
            owner   => 'apache',
            group   => 'apache',
            mode    => '0644',
            content => template("profile/some_profile/${template['erb']}"),
        }
    }
    $files.each | $file| {
        file{
            "${file['file']}":
                ensure => present,
                owner  => 'root',
                group  => 'root',
                mode   => '0644',
                source =>
"puppet:///modules/profile/some_profile/${file['source']}"
        }
    }
    $links.each |$link| {
        file {
            "${link['link']}":
                ensure => 'link',
                target => "${link['target']}";
        }
    }
}
```

```
.pp
firewall {'102 forward port
80 to 82':
  table      => 'nat',
  chain      => 'PREROUTING',
  proto      => tcp,
  dport      => '80',
  destination =>
'XXX.XXX.XXX.XXX',
  todest      =>
'YYY.YYY.YYY.YYY:82',
  jump        => 'DNAT',
}
```

PCB. Packages

```

## install package from RPM
package
{'icinga-rpm-release-7-2.el7.
centos.noarch':
    provider => 'rpm',
    source   =>
"https://packages.icinga.com/
epel/icinga-rpm-release-7-latest.noarch.rpm",
    ensure   => present,
}

## install more then one
package
package { [
    'MySQL-python',
    'icinga2',

'nagios-plugins-all',
    'vim-icinga2',
    'mariadb-server',
    'mariadb',

'icinga2-ido-mysql',
    'php',
    'php-ldap',
    'php-mysql',
    'php-pgsql',
    'icingacli',
]:
    ensure => installed,
}

```

PCB. php.ini

```

.php
php::ini { '/etc/php.ini':
    expose_php      => 'Off',
    display_errors => 'On',
    date_timezone  => 'UTC',
    error_log       => syslog,
    short_open_tag => 'On'
}

```

Puppet. Cert & Curl

allow check and delete certificate

/etc/puppetlabs/puppetserver/conf.d/auth.conf

```
authorization: {
    version: 1
    rules: [
# ...
        {
            match-request: {
                path: "/puppet-ca/v1/certificate_status/"
                type: path
                method: [ get, put, delete ]
            }
            allow-unauthenticated: true
            sort-order: 200
            name: "certificate_status"
        },
        {
            match-request: {
                path: "/puppet-ca/v1/certificate_statuses/"
                type: path
                method: get
            }
            allow-unauthenticated: true
            sort-order: 200
            name: "certificate_statuses"
        },
# ...
    ]
}
```

curl requests

```
curl -w '\n' -s -H "Accept: text/json" -k -X GET  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM | tr ',' '\n'
```

```
    environment  
curl -w '\n' -s -H "Accept: text/json" -k -X GET  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM?environment=PRODUCTION | tr ',' '\n'
```

```
curl -w '\n' -s -H "Accept: text/json" -k -X DELETE  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM?environment=PRODUCTION | tr ',' '\n'
```

```
curl -w '\n' -s -H "Accept: text/json" -k -X GET  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM | tr ',' '\n'
```

```
"state": "requested"
```

```
curl -w '\n' -s -H "Accept: text/json" -k -X DELETE  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM | tr ',' '\n'
```

```
find /etc/puppetlabs/puppet/ssl/ -type f -delete
```

```
/opt/puppetlabs/bin/puppet agent --test
```

```
"state": "requested"
```

```
curl -w '\n' -s -H "Content-Type: text/json" -k -X PUT --data  
'{"desired_state": "signed"}'  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM | tr ',' '\n'
```

```
(      "state": "revoked")  
curl -w '\n' -s -H "Content-Type: text/json" -k -X PUT --data  
'{"desired_state": "revoked"}'  
https://puppet-master.EXAMPLE.COM:8140/puppet-ca/v1/certificate_status/HOS  
TNAME.EXAMPLE.COM | tr ',' '\n'
```

Puppet. Hiera

типы данных

.yaml

```
---  
# string  
data_1:  
  'aaa'  
  'bbb'  
  'ccc'
```

```
# array  
data_2:  
  - aaa  
  - bbb  
  - ccc
```

```
# hash  
data_3:  
  'AAA':  
    aaa: '111'  
    bbb: '222'  
    ccc: '333'  
  'BBB':  
    aaa: '111'  
    bbb: '222'  
    ccc: '333'
```

.pp

```
class profile::some_profile(  
  $data_1 = hiera('data_1')  
  $data_2 = hiera_array('data_2')  
  $data_3 = hiera_hash('data_3')  
)  
{  
  if $data_1 != undef { module_name { $data_1: } }  
  if $data_2 != undef { module_name { $data_2: } }  
  if $data_3 != undef { create_resources( module_name, $data_3, {} ) }  
}
```

Puppet. Ресурсы

Ресурс — это самая мелкая единица абстракции в Puppet. Ресурсами могут быть:

- файлы;
- пакеты (Puppet поддерживает пакетные системы многих дистрибутивов);

- сервисы;
- пользователи;
- группы;
- задачи cron;
- и т. д.

Синтаксис ресурсов вы можете невозбранно подсмотреть в [документации](#).
В Puppet есть возможность добавлять свои ресурсы.

пример ресурса file (В терминах Puppet здесь описан ресурс типа файл с названием (title) /tmp/helloworld.)

```
file { '/tmp/helloworld':
  ensure  => present,          #
  content => 'Hello, world!',   #      "Hello, world!"
  mode    => 0644,             #      - 0644
  owner   => 'root',           #      - root
  group   => 'root'            #      - root
}
```

Зависимости

Многие ресурсы зависят от других ресурсов. Например, для ресурса «сервис sshd» необходим ресурс «пакет sshd» и опционально «конфиг sshd»

Посмотрим, как это реализуется:

Конфигурация SSHD сервиса

```
file { 'sshd_config':
  path      => '/etc/ssh/sshd_config',
  ensure    => file,
  content   => "Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
UsePrivilegeSeparation yes
Key_regeneration_interval 3600
ServerKeyBits 768
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes",
  mode      => 0644,
  owner   => 'root',
  group   => 'root',
  require  => Package['sshd']
}
```

Установка пакета "openssh-server"

```
package { 'sshd':
  ensure => latest,
  name   => 'openssh-server'
}
```

Запуск службы "sshd"

```
service { 'sshd':
  ensure     => running,
  enable     => true,
  name       => 'ssh'
  subscribe  => File['sshd_config'],
  require    => Package['sshd']
}
```

Здесь используется онлайн-конфиг, который делает манифест некрасивым. На самом деле так почти никогда не делается, существует механизм шаблонов, основанный на ERB, и возможность просто использовать внешние файлы. Но нас не это интересует сейчас.

Варианты зависимостей

Puppet поддерживает много вариантов описания зависимостей. Подробно, как всегда, можно прочитать в [документации](#).

- **Require** означает ровно то, что ожидается. Если ресурс А зависит (require) от ресурса Б, то Puppet сначала обработает ресурс Б, а потом вернётся к ресурсу А.
- **Subscribe** даёт чуть более хитрое поведение. Если ресурс А подписан (subscribe) на ресурс Б, то Puppet сначала обработает ресурс Б, а потом вернётся к ресурсу А (поведение как у require), и далее при изменениях Б, будет заново обрабатываться А. Это очень удобно для создания сервисов, зависящих от их конфигов (как в примере выше). Если конфиг изменяется, сервер перезапускается, не нужно самому об этом беспокоиться.

Puppet. Структура каталогов

```

/etc/puppetlabs/
  code
    environments
      production
        environment.conf
        hieradata
        manifests
        modules
      modules
    mcollective
      client.cfg
      data-help.erb
      discovery-help.erb
      facts.yaml
      metadata-help.erb
      rpc-help.erb
      server.cfg
    puppet
      auth.conf
      hiera.yaml
      puppet.conf
      ssl
        ca
          ca_crl.pem
          ca_crt.pem
          ca_key.pem
          ca_pub.pem
          inventory.txt
        private
          requests
        serial
        signed
          icinga.example.com.pem
          puppet.example.com.pem
          puppet-slave.example.com.pem
      certificate_requests
      certs
        ca.pem
        puppet.example.com.pem
      crl.pem
      private
        private_keys
          puppet.example.com.pem
      public_keys
        puppet.example.com.pem
    puppetserver
      conf.d
        auth.conf
        global.conf
        puppetserver.conf
        web-routes.conf
        webserver.conf
      logback.xml
      request-logging.xml
      services.d
        ca.cfg
    pxp-agent
      modules

```

99. Other

curl ipecho.net/plain ; echo	get external IP address

Arch Linux

Installation

LVM+loader

```

# parted /dev/sda
(parted) mklabel gpt
(parted) unit MiB          # MB, GB - 100; MiB, Gib - 1024
(parted) mkpart root 0 512 # (e   ,    )
(parted) set 1 boot on    #   "boot"  1
(parted) mkpart lvm 512 -1 # -1
(parted) print
(parted) q
# mkfs.fat -F32 /dev/sda1
# vgcreate G1 /dev/sda2      # volume group (vg) c G1.
# lvcreate -n root -L 25GiB # logical volume (lv) c root
# lvcreate -n home -l 100%FREE G1 # ,       (   )
# mkfs.xfs /dev/G1/root
# mkfs.xfs /dev/G1/home
# lsblk -f
# mount /dev/G1/root /mnt
# mkdir /mnt/{boot,home}
# mount /dev/sda1 /mnt/boot
# mount /dev/G1/home /mnt/home

# pacstrap /mnt base base-devel
# genfstab -L /mnt >> /mnt/etc/fstab # -L ; -U UUID ( ? )
# arch-chroot /mnt/

# sed -i 's/#en_US.UTF/en_US.UTF/' /etc/locale.gen
# locale-gen
##### localectl set-locale LANG="en_US.UTF-8"
# echo myhostname > /etc/hostname
pacman -S dialog wpa_supplicant refind-efi
##### mkdir -p /boot/EFI/Boot
# cp /usr/share/refind/refind_x64.efi /boot/EFI/Boot/bootx64.efi
# cp -r /usr/share/refind/drivers_x64/ /boot/EFI/Boot/
# echo 'extra_kernel_version_strings linux-git,linux-lts,linux;' >
/esp/EFI/Boot/refind.conf
# passwd
# useradd -m -G wheel -s /bin/bash myusername
# passwd myusername
# pacman -S vim
# visudo  # %wheel ALL=(ALL) NOPASSWD: ALL      #
# exit

```

install same packages from existing arch installation

```

# pacman -Qqen > /mnt/tmp/pkglist.txt
# arch-chroot /mnt/
# pacman -S - < pkglist.txt

```

AWS

awscli

install

on HREL7

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm  
rmp -i epel-release-latest-7.noarch.rpm  
vim /etc/yum.repos.d/epel.repo # enable 0 -> 1  
yum install python2-pip  
pip install awscli  
  
aws configure  
...
```

copy from s3

copy from s3

```
aws s3 cp --recursive s3://BACKET_NAME/MY_FOLDER /local_folder/
```

Bareos

Bareos. Bconsole commands

Full restore

```
( jobstatus=T)  
*list job=nl-edc-artemk-test.meteogroup.net jobstatus=T  
  
*restore jobid=18218 client=nl-edc-artemk-test.meteogroup.net  
restoreclient=nl-edc-artemk-test.meteogroup.net where=/  
restorejob=RestoreFilesLin all done yes  
  
 cwd is: /  
 $ mark *
```

Информация

Jobs

Abbr.	Description	Weight
C	Created, not yet running	15
R	Running	15
B	Blocked	15
T	Completed successfully	10
E	Terminated with errors	25
e	Non-fatal error	20
f	Fatal error	100
D	Verify found differences	15
A	Canceled by user	90
I	Incomplete job	15
L	Committing data	15
W	Terminated with warnings	20
I	Doing data despooling	15
q	Queued waiting for device	15
F	Waiting for Client	15
S	Waiting for Storage daemon	15
m	Waiting for new media	15
M	Waiting for media mount	15
s	Waiting for storage resource	15

```

echo "list jobs jobstatus=R" | bconsole
,
()
echo "list jobs jobstatus=C" | bconsole

echo "list jobs jobstatus=f" | bconsole

```

j	Waiting for job resource	15
c	Waiting for client resource	15
d	Waiting on maximum jobs	15
t	Waiting on start time	15
p	Waiting on higher priority jobs	15
i	Doing batch insert file records	15
a	SD despooling attributes	15

Кейсы

Перезапустить все упавшие джобы бекава и удалить записи о них

```

    ($11=="B" , , "R")
echo "list jobs jobstatus=f" | bconsole | awk '$11=="B"{print $4}' |
sort | uniq | while read NAME; do echo "run job=$NAME yes" | bconsole;
done

echo "list jobs jobstatus=f" | bconsole | awk '$11=="B'{print $2}' | sed
s/,//g | while read ID; do echo "delete jobid=$ID" | bconsole; done

```

BASH SCRIPTS

DNS

Get all MX records and their IPs for list of domains

mx_check.sh

```
#!/bin/bash
DOMAIN_LIST=$1
cat $DOMAIN_LIST |
while read DOMAIN;
do
  dig +short MX $DOMAIN |
  while read WEIGHT MX
  do
    dig +short A $MX |
    while read A
    do
      echo -e "\"$DOMAIN\" , \"$WEIGHT\" , \"$MX\" , \"$A\" "
    done
  done
done
done
```

Example

```
$ echo -e gmail.com "\n" i.ua | ./mx_check.sh
"gmail.com", "10", "alt1.gmail-smtp-in.l.google.com.", "74.125.23.26"
"gmail.com", "40", "alt4.gmail-smtp-in.l.google.com.", "74.125.69.27"
"gmail.com", "20", "alt2.gmail-smtp-in.l.google.com.", "74.125.28.27"
"gmail.com", "30", "alt3.gmail-smtp-in.l.google.com.", "74.125.198.26"
"gmail.com", "5", "gmail-smtp-in.l.google.com.", "64.233.165.26"
"i.ua", "11", "mx14.i.ua.", "213.186.119.6"
"i.ua", "25", "mx12.i.ua.", "91.198.36.66"
"i.ua", "10", "mx20.i.ua.", "51.15.142.90"
"i.ua", "15", "mx5.i.ua.", "91.198.36.14"
"i.ua", "15", "mx6.i.ua.", "91.198.36.95"
"i.ua", "10", "mx13.i.ua.", "213.186.119.6"
"i.ua", "10", "mx10.i.ua.", "213.186.119.6"
"i.ua", "10", "mx18.i.ua.", "163.172.152.120"
"i.ua", "10", "mx16.i.ua.", "163.172.153.94"
"i.ua", "10", "mx7.i.ua.", "213.186.122.10"
"i.ua", "10", "mx19.i.ua.", "51.15.129.137"
"i.ua", "20", "mx11.i.ua.", "91.198.36.62"
"i.ua", "10", "mx15.i.ua.", "163.172.154.152"
```

Fix welcome message on Ubuntu's

```
sudo chmod -x /etc/update-motd.d/*
sudo chmod +x /etc/update-motd.d/99-footer
sudo rm /etc/motd
sudo ln -s /etc/motd.tail /etc/motd
```

HP-ILo

power on – Turn the server on

power off – Turn the server off

power reset – Reset the server

restart ILO

```
</>hpiLO-> cd /map1  
</map1>hpiLO-> reset
```

Icinga2. Cluster configuration.

OS

CentOS 7

Hosts

1. icinga-master.example.com
2. icinga-satellite.example.com
3. test-node.example.com

Ansible

Install Master

```
master.yaml
```

```
---
```

```
- hosts: all
```



```
vars:
```

```
  path:
```

```
    PATH: /sbin:/bin:/usr/sbin:/usr/bin
```



```
tasks:
```

```
## SELINUX
```

```
  - name: setenforce 0
```

```
    selinux:
```

```
      policy: targeted
```

```
      state: permissive
```



```
## REPO
```

```
  - name: Enable yum repos
```

```
    yum:
```

```
      name: "{{ item }}"
```

```

        state: present
    with_items:
    -
https://packages.icinga.com/epel/icinga-rpm-release-7-latest.noarch.rpm
    - epel-release

## UNSTALL
- name: Install packages
  yum:
    name: "{{ item }}"
    state: present
  with_items:
    - vim
    - MySQL-python
    - icinga2
    - nagios-plugins-all
    - vim-icinga2
    - mariadb-server
    - mariadb
    - icinga2-ido-mysql
    - httpd
    - php
    - php-ldap
    - php-mysql
    - php-pgsql
    - icingacli
    - icingaweb2

## PHP.INI
- name: Set date.timezone for CLI
  become: true
  lineinfile:
    dest: /etc/php.ini
    regexp: "date.timezone ="
    line: "date.timezone = UTC"

## CREATE DB
- name: Enable and Start MariaDB
  systemd:
    name: mariadb
    state: started
    enabled: True

- name: 'MariaDB: create databases'
  mysql_db:
    name: "{{item}}"
    state: present
  with_items:
    - icinga
    - icingaweb2
  register: db_created

```

```

- name: 'MariaDB - import database schema'
  mysql_db:
    name: icinga
    state: import
    collation: True
    target: /usr/share/icinga2-ido-mysql/schema/mysql.sql
  when: db_created.changed

## COPY ido-mysql.conf
- name: 'MariaDB: copy ido-mysql.conf'
  copy:
    src: icinga2-files/ido-mysql.conf
    dest: /etc/icinga2/features-available/ido-mysql.conf
    owner: icinga
    group: icinga
    mode: 0640
  register: ido_mysql_conf

## COPY api-users.conf
- name: copy api-users.conf
  copy:
    src: icinga2-files/api-users.conf
    dest: /etc/icinga2/conf.d/api-users.conf
    owner: icinga
    group: icinga
    mode: 0640
  register: api_users

## DB USERS
- name: 'MariaDB: create user "icinga"'
  # when: db_created.changed
  mysql_user:
    name: icinga
    password: 12345
    # priv: "icinga.*:GRANT,SELECT,INSERT,UPDATE,DELETE,DROP,CREATE
VIEW,INDEX,EXECUTE"
    priv: "icinga.*:ALL"
    state: present
- name: 'MariaDB: create user "icingaweb2"'
  # when: db_created.changed
  mysql_user:
    name: icingaweb2
    password: 12345
    priv: "icingaweb2.*:ALL"
    state: present
  register: db_created.changed

## SETUP ICINGA API

```

```
- name: icinga2 api setup
  command: icinga2 api setup
  when: db_created.changed
  become: true
  environment: "{{path}}"

## CREATE ICINGA TOKEN
- name: Create icinga token
  command: icingacli setup token create
  become: true
  environment: "{{path}}"
- name: Show icinga token
  command: icingacli setup token show
  become: true
  environment: "{{path}}"
  register: icinga_token

## ENABLE/RESTART HTTPD, MARIADB, ICINGA2
- name: Systemd - Enable and Start httpd
  systemd:
    name: "{{item}}"
    state: restarted
    enabled: True
  with_items:
    - httpd
    - mariadb
    - icinga2
```

```
- debug: msg="{{icinga_token.stdout}}"
```

Install Satellite

sattelite.yaml

```
---
```

```
- hosts: all
```

```
vars:
```

```
    path:
        PATH: /sbin:/bin:/usr/sbin:/usr/bin
```

```
tasks:
```

```
## SELINUX
```

```
- name: setenforce 0
  selinux:
    policy: targeted
    state: permissive
```

```
## REPO
```

```
- name: Enamle yum repos
  yum:
    name: "{{ item }}"
    state: present
  with_items:
    - https://packages.icinga.com/epel/icinga-rpm-release-7-latest.noarch.rpm
      - epel-release
```

```
## UNSTALL
```

```
- name: Install packages
  yum:
    name: "{{ item }}"
    state: present
  with_items:
    - vim
    - MySQL-python
    - icinga2
    - nagios-plugins-all
    - vim-icinga2
    - mariadb-server
    - mariadb
    - icinga2-ido-mysql
    - httpd
```

```

    - php
    - php-ldap
    - php-mysql
    - php-pgsql
    - icingacli

## PHP.INI
- name: Set date.timezone for CLI
  become: true
  lineinfile:
    dest: /etc/php.ini
    regexp: "date.timezone ="
    line: "date.timezone = UTC"

## CREATE DB
- name: Enable and Start MariaDB
  systemd:
    name: mariadb
    state: started
    enabled: True

- name: 'MariaDB: create databases'
  mysql_db:
    name: "{{item}}"
    state: present
  with_items:
    - icinga
    - icingaweb2
  register: db_created

- name: 'MariaDB - import database schema'
  mysql_db:
    name: icinga
    state: import
    collation: True
    target: /usr/share/icinga2-ido-mysql/schema/mysql.sql
  when: db_created.changed

## COPY ido-mysql.conf
- name: 'MariaDB: copy ido-mysql.conf'
  copy:
    src: icinga2-files/ido-mysql.conf
    dest: /etc/icinga2/features-available/ido-mysql.conf
    owner: icinga
    group: icinga
    mode: 0640
  register: ido_mysql_conf

## COPY api-users.conf
- name: copy api-users.conf

```

```

copy:
  src: icinga2-files/api-users.conf
  dest: /etc/icinga2/conf.d/api-users.conf
  owner: icinga
  group: icinga
  mode: 0640
  register: api_users

## DB USERS
- name: 'MariaDB: create user "icinga"'
  mysql_user:
    name: icinga
    password: 12345
    # priv: "icinga.*:GRANT,SELECT,INSERT,UPDATE,DELETE,DROP,CREATE
VIEW,INDEX,EXECUTE"
    priv: "icinga.*:ALL"
    state: present
- name: 'MariaDB: create user "icingaweb2"'
  mysql_user:
    name: icingaweb2
    password: 12345
    priv: "icingaweb2.*:ALL"
    state: present
  register: db_created.changed

## SETUP ICINGA API
- name: icinga2 api setup
  command: icinga2 api setup
  become: true
  environment: "{{path}}"

## ENABLE/RESTART MARIADB, ICINGA2
- name: Systemd - Enable and Start httpd
  systemd:
    name: "{{item}}"
    state: restarted
    enabled: True

```

```
with_items:  
  - mariadb  
  - icinga2
```

Add Satellite to Master

Master

```
[root@icinga-master ~]# icinga2 node wizard  
...  
Please specify if this is a satellite setup ('n' installs a master setup)  
[Y/n]: N  
Please specify the common name (CN) [ec2]: icinga-master.example.com  
Bind Host []:  
Bind Port []:  
done  
  
[root@icinga-master ~]# systemctl restart icinga2
```

Sattellite

```
[root@icinga-satellite ~]# icinga2 node wizard  
...  
Please specify if this is a satellite setup ('n' installs a master setup)  
[Y/n]: Y  
Please specify the common name (CN) [icinga-client]:  
icinga-satellite.example.com  
Master Common Name (CN from your master setup): icinga-master.example.com  
Do you want to establish a connection to the master from this node? [Y/n]:  
Y  
Master endpoint host (Your master's IP address or FQDN):  
icinga-master.example.com  
Master endpoint port [5665]:  
Add more master endpoints? [y/N]: N  
Host [icinga-master.example.com]:  
Port [5665]:  
Is this information correct? [y/N]: Y  
(Hint: # icinga2 pki ticket --cn 'icinga-satellite.example.com'):  
XXXXXXXXXXXXXXXXXXXXXXXXXXXX  
...  
...
```

Master

```
[root@icinga-master ~]# icinga2 pki ticket --cn  
'icinga-satellite.example.com'  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Sattellite

```
...  
Bind Host []:  
Bind Port []:  
Accept config from master? [y/N]: Y  
Accept commands from master? [y/N]: Y  
Done.
```

```
[root@icinga-satellite ~]# systemctl restart icinga2
```

Keepalived + HAProxy + Nginx

Схема взаимодействия

ha1.example.com - основной сервер для HAProxy.

192.168.33.10 WAN (eth1)
10.10.10.10 LAN (eth2)

Virtual IPs which are managed by Keepalived:
192.168.33.101/24
192.168.33.102/24
192.168.33.103/24

ha2.example.com - слейв для HAProxy

192.168.33.20 WAN
10.10.10.20 LAN

node1.example.com - нода1 бекенда (Nginx)

10.10.10.100 LAN

node2.example.com - нода2 бекенда (Nginx)

10.10.10.200 LAN

Конфиги

Keepalived

ha1.example.com

VRRP IP 224.0.0.18 VRRP (112)

Keepalived

```
# yum install -y keepalived

# iptables -I INPUT -i eth0 -d 224.0.0.0/8 -p vrrp -j ACCEPT
# iptables -I OUTPUT -o eth0 -d 224.0.0.0/8 -p vrrp -j ACCEPT
# service iptables save

# systemctl enable keepalived.service
# systemctl start keepalived.service
```

Разрешаем форвард пакетов

/etc/sysctl.d/10-keepalived.conf

```
net.ipv4.ip_forward = 1
```

Применяем

```
# sysctl -p --load=/etc/sysctl.d/10-keepalived.conf
net.ipv4.ip_forward = 1
```

/etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {
    notification_email {
        admin@example.com
    }
    notification_email_from ral@example.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.33.101/24
        192.168.33.102/24
        192.168.33.103/24
    }
}
...
...
```

Можно так же присваивать лейблы виртуальным IP (хотя и не обязательно)

/etc/keepalived/keepalived.conf

```
...
virtual_ipaddress {
    192.168.33.101/24 label WWW-1
    192.168.33.102/24 label WWW-2
    192.168.33.103/24 label FTP-1
}
...
```

Keepalived

```
# yum install -y keepalived  
# systemctl enable keepalived.service  
# systemctl start keepalived.service
```

Разрешаем форвард пакетов

/etc/sysctl.d/10-keepalived.conf

```
net.ipv4.ip_forward = 1
```

Применяем

```
# sysctl -p --load=/etc/sysctl.d/10-keepalived.conf  
net.ipv4.ip_forward = 1
```

/etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {  
    notification_email {  
        admin@example.com  
    }  
    notification_email_from ha2@example.com  
    smtp_server 127.0.0.1  
    smtp_connect_timeout 30  
    # router_id LVS_DEVEL  
}  
  
vrrp_instance VI_1 {  
    state BACKUP  
    interface eth1  
    virtual_router_id 51  
    priority 100  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass 1111  
    }  
    virtual_ipaddress {  
        192.168.33.101/24  
        192.168.33.102/24  
        192.168.33.103/24  
    }  
}
```

HAProxy

Для того чтобы HAProxy можно было биндить на виртуальные IP (которые могут быть иногда недоступны, когда срабатывает Keepalived и мы рестартует HAProxy, получаем ошибку "cannot bind socket [192.168.33.101:80]") разрешим системе бинд несуществующих интерфейсов:

```
/etc/sysctl.d/10-haproxy.conf
```

```
net.ipv4.ip_nonlocal_bind=1
```

Применим настройки

```
# sysctl -p --load /etc/sysctl.d/10-haproxy.conf
net.ipv4.ip_nonlocal_bind = 1
```

Чтоб проверить что мы избежали этой ошибки, уроним Keepalive, Рестартанем HAProxy и убедимся что он все равно слушает на НЕСУЩЕСТВУЮЩЕМ IP:

```
# systemctl stop keepalived.service
# systemctl restart haproxy.service
# systemctl status haproxy.service
```

Примечание

чтобы узнать какой сервер сейчас активен достаточно ввести "ip a" и на активном сервере будут видны поднятые виртуальные IP, а на неактивном - нет

Active server

```
# ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:d8:71:80 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 82753sec preferred_lft 82753sec
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:2e:de:2e brd ff:ff:ff:ff:ff:ff
    inet 192.168.33.10/24 brd 192.168.33.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet 192.168.33.101/24 scope global secondary eth1
        valid_lft forever preferred_lft forever
    inet 192.168.33.102/24 scope global secondary eth1
        valid_lft forever preferred_lft forever
    inet 192.168.33.103/24 scope global secondary eth1
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:c3:30:d5 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.10/24 brd 10.10.10.255 scope global eth2
        valid_lft forever preferred_lft forever
```

Passive server

```
# ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:d8:71:80 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 82856sec preferred_lft 82856sec
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:9c:1b:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.33.20/24 brd 192.168.33.255 scope global eth1
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:e1:b8:3e brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.20/24 brd 10.10.10.255 scope global eth2
        valid_lft forever preferred_lft forever
```

Ссылки по теме

About the Keepalived Configuration File

https://docs.oracle.com/cd/E37670_01/E41138/html/section_wv3_hjn_pr.html

Configuring Simple Load Balancing Using HAProxy

https://docs.oracle.com/cd/E37670_01/E41138/html/section_brt_vmb_nr.html

Making HAProxy Highly Available Using Keepalived

https://docs.oracle.com/cd/E37670_01/E41138/html/section_sm3_svy_4r.html

Configuring Simple Load Balancing Using HAProxy

https://docs.oracle.com/cd/E37670_01/E41138/html/section_brt_vmb_nr.html

Configuring HAProxy for Session Persistence

https://docs.oracle.com/cd/E37670_01/E41138/html/section_jyh_zhz_4r.html

Linux Desktop Programs

xnviewmp - программа для просмотра изображений

foxitreader - pdf

xfe - файловый менеджер

terminator - консоль

franz-bin - меседжер (все в одном)

thunderbird - почта

MySQL

mysqldump

all-databases

```
mysqldump -u <DB_USERNAME> -p --all-databases > all-databases.sql  
mysqldump -u <DB_USERNAME> -p --all-databases | gzip > all-databases.sql.gz
```

one database

```
mysqldump -u <DB_USERNAME> -p --database <DB_NAME> > <DB_NAME>.sql  
mysqldump -u <DB_USERNAME> -p --database <DB_NAME> | gzip >  
<DB_NAME>.sql.gz
```

one table

```
mysqldump -u <DB_USERNAME> -p <DB_NAME> <TABLE_NAME> >  
<DB_NAME>.<TABLE_NAME>.sql  
mysqldump -u <DB_USERNAME> -p <DB_NAME> <TABLE_NAME> | gzip >  
<DB_NAME>.<TABLE_NAME>.sql.gz
```

mysqldump slave

mysqldump on slave server

```
mysqladmin stop-slave  
mysqldump --all-databases > all-databases.sql  
mysqladmin start-slave
```

restore

restore all databases

```
mysql -u <DB_USERNAME> -p < all-databases.sql  
gunzip < all-databases.sql.gz | mysql -u <DB_USERNAME> -p
```

restore one database

```
mysql -u <DB_USERNAME> -p <DB_NAME> < <DB_NAME>.sql  
gunzip < <DB_NAME>.sql.gz | mysql -u <DB_USERNAME> -p <DB_NAME>
```

restore one table

```
mysql -u <DB_USERNAME> -p <DB_NAME> < <DB_NAME>.<TABLE_NAME>.sql  
gunzip < <DB_NAME>.<TABLE_NAME>.sql.gz | mysql -u <DB_USERNAME> -p  
<DB_NAME>
```

restore one database from full backup

```
mysql -u <DB_USERNAME> -p --one-database <DB_NAME> < all-databases.sql  
gunzip < all-databases.sql.gz | mysql -u <DB_USERNAME> -p --one-database  
<DB_NAME>
```

commands

show databases

```
mysqlshow -u USER -pPASSWORD  
mysql> SHOW DATABASES;
```

show tables in database

```
mysqlshow -u USER -pPASSWORD <DB_NAME>  
mysql> SHOW TABLES FROM <DB_NAME>;
```

Show MySQL process list

```
mysqladmin -i 1 processlist # -i (interval in second)  
mysql> SHOW PROCESSLIST;
```

```
mysql> SHOW FULL PROCESSLIST;
```

Kill process by ID

```
mysql> KILL <ID>;
```

Nagios + Git + Jenkins

Описание сервера и его роль

CentOS7, стандартная установка Nagios из реп

цель: разместить все конфиги Nagios в Git-репозитории, все изменения делать в скопированной папке у себя в хоме, проверять на ошибки код, пушить в ветку DEV.

Jenkins ловит новый коммит в DEV и тянет код на сервер в tmp каталог, проверяет на ошибки и если все ок - мержит с мастером и делает

git pull на сервере.

1. засунул дефолтный конфиги в гит

```
# cd /etc/nagios
# touch conf.d/.keep #           GIT
# git init
# git add -A
# git commit -m "init"
# git remote add origin git@bitbucket.org:USERNAME/NAGIOS.git
# git push -u origin master
```

2. мувнул папку в сторону и изменил абсолютные пути на динамические

```
# mv /etc/nagios /etc/nagios_
# vim /etc/nagios_/nagios.conf
:%s/\`/etc\`/nagios\`//g
:wq
```

3. проверяю что проверка конфигов работает в "мувнутой" папке

```
# /sbin/nagios -v /etc/nagios/_nagios.cfg

Nagios Core 4.2.4
Copyright (c) 2009-present Nagios Core Development Team and Community
Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 12-07-2016
License: GPL

Website: https://www.nagios.org
Reading configuration data...
    Read main config file okay...
    Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
    Checked 8 services.
    Checked 1 hosts.
    Checked 1 host groups.
    Checked 0 service groups.
    Checked 1 contacts.
    Checked 1 contact groups.
    Checked 24 commands.
    Checked 5 time periods.
    Checked 0 host escalations.
    Checked 0 service escalations.
Checking for circular paths...
    Checked 1 hosts
    Checked 0 service dependencies
    Checked 0 host dependencies
    Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight
check
```

4. мувую папку Nagios обратно

```
# mv /etc/nagios_ /etc/nagios
```

5. копирую с гита Nagios к себе в хому, назрещаю читать мою хому не только мне

```
$ cd ~  
$ git clone origin git@bitbucket.org:USERNAME/NAGIOS.git  
$ chmod +rx /home/USERNAME
```

6. проверяю работает ли чек конфигов

```
$ sudo /sbin/nagios -v /home/USERNAME/nagios/nagios.cfg  
  
Nagios Core 4.2.4  
Copyright (c) 2009-present Nagios Core Development Team and Community  
Contributors  
Copyright (c) 1999-2009 Ethan Galstad  
Last Modified: 12-07-2016  
License: GPL  
  
Website: https://www.nagios.org  
Reading configuration data...  
    Read main config file okay...  
    Read object config files okay...  
  
Running pre-flight check on configuration data...  
  
Checking objects...  
    Checked 8 services.  
    Checked 1 hosts.  
    Checked 1 host groups.  
    Checked 0 service groups.  
    Checked 1 contacts.  
    Checked 1 contact groups.  
    Checked 24 commands.  
    Checked 5 time periods.  
    Checked 0 host escalations.  
    Checked 0 service escalations.  
Checking for circular paths...  
    Checked 1 hosts  
    Checked 0 service dependencies  
    Checked 0 host dependencies  
    Checked 5 timeperiods  
Checking global event handlers...  
Checking obsessive compulsive processor commands...  
Checking misc settings...  
  
Total Warnings: 0  
Total Errors: 0  
  
Things look okay - No serious problems were detected during the pre-flight  
check
```

Old Repos

Ubuntu 10.04

/etc/apt/sources.list

```
## EOL upgrade sources.list
# Required
deb http://old-releases.ubuntu.com/ubuntu/ lucid main restricted universe
multiverse
deb http://old-releases.ubuntu.com/ubuntu/ lucid-updates main restricted
universe multiverse
deb http://old-releases.ubuntu.com/ubuntu/ lucid-security main restricted
universe multiverse

# MySQL Percona
deb http://repo.percona.com/apt lucid main # disabled on upgrade to lucid
```

игнор просроченных ключей репы

/etc/apt/apt.conf

```
Acquire::Check-Valid-Until false;
```

Debian 6

/etc/apt/sources.list

```
deb http://archive.debian.org/debian squeeze main
deb http://archive.debian.org/debian squeeze-lts main
```

игнор просроченных ключей репы

/etc/apt/apt.conf

```
Acquire::Check-Valid-Until false;
```

CentOS 5.11

/etc/yum.repos.d/CentOS-Base.repo

```
[base]
```

```
name=CentOS-$releasever - Base
baseurl=http://vault.centos.org/5.11/os/$basearch/
gpgcheck=1
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=1

#released updates
[update]
name=CentOS-$releasever - Updates
baseurl=http://vault.centos.org/5.11/updates/$basearch/
gpgcheck=1
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=1

#packages used/produced in the build but not released
[addons]
name=CentOS-$releasever - Addons
baseurl=http://vault.centos.org/5.11/addons/$basearch/
gpgcheck=1
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=1

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
baseurl=http://vault.centos.org/5.11/extras/$basearch/
gpgcheck=1
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=1

#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
baseurl=http://vault.centos.org/5.11/centosplus/$basearch/
gpgcheck=1
enabled=0
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=2

#contrib - packages by Centos Users
[contrib]
name=CentOS-$releasever - Contrib
baseurl=http://vault.centos.org/5.11/contrib/$basearch/
gpgcheck=1
enabled=0
gpgkey=http://vault.centos.org/RPM-GPG-KEY-centos4
protect=1
priority=2
```

OWA or Office 360 mail & Linux

- Thunderbird & Office 360
- Thunderbird & OWA
 - Установка
 - Настройка почты OWA
 - Настройка календаря OWA

Thunderbird & Office 360

доставляем указанные плагины, синкаем с шита плагин под календарик от мелкомягких, и устанавливаем его ручками из собранного пакета.

<https://github.com/Ericsson/exchangecalendar>

почта:

<https://kb.wisc.edu/office365/page.php?id=28427>

Thunderbird & OWA

Установка

| \$ yaourt -S davmail

запускаю davmail.

Main Network Encryption Logging Advanced

Gateway

Exchange Protocol: Auto ▾

OWA (Exchange) URL: https://owa[REDACTED]

Local POP port: 1110 No SSL

Local IMAP port: 1143 No SSL

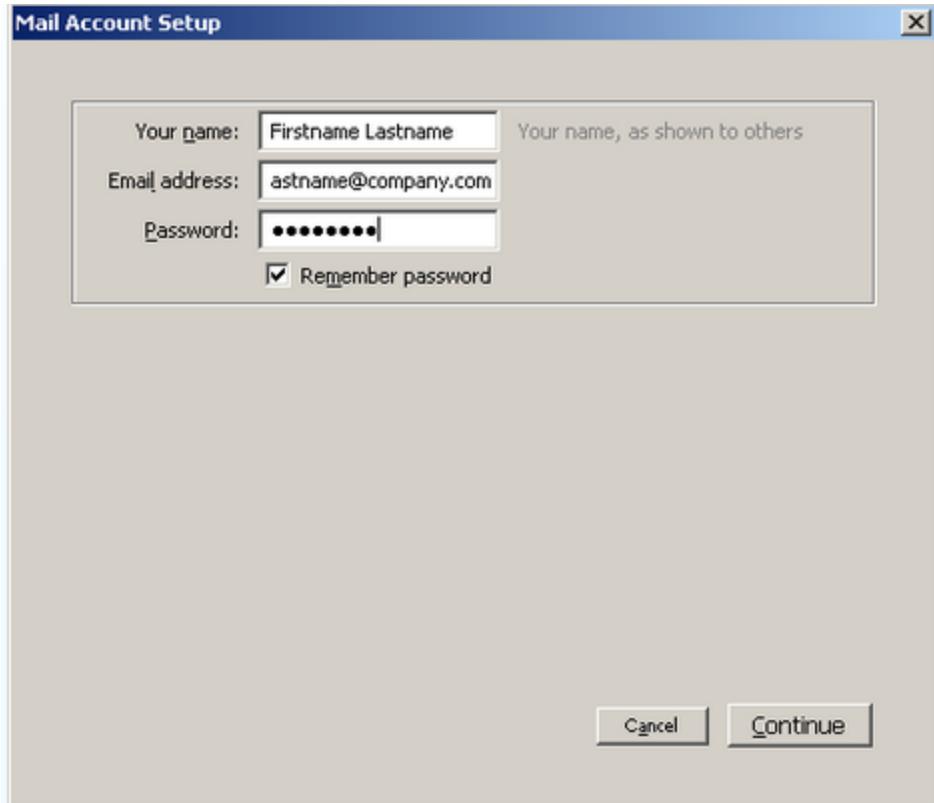
Local SMTP port: 1025 No SSL

CalDAV HTTP port: 1080 No SSL

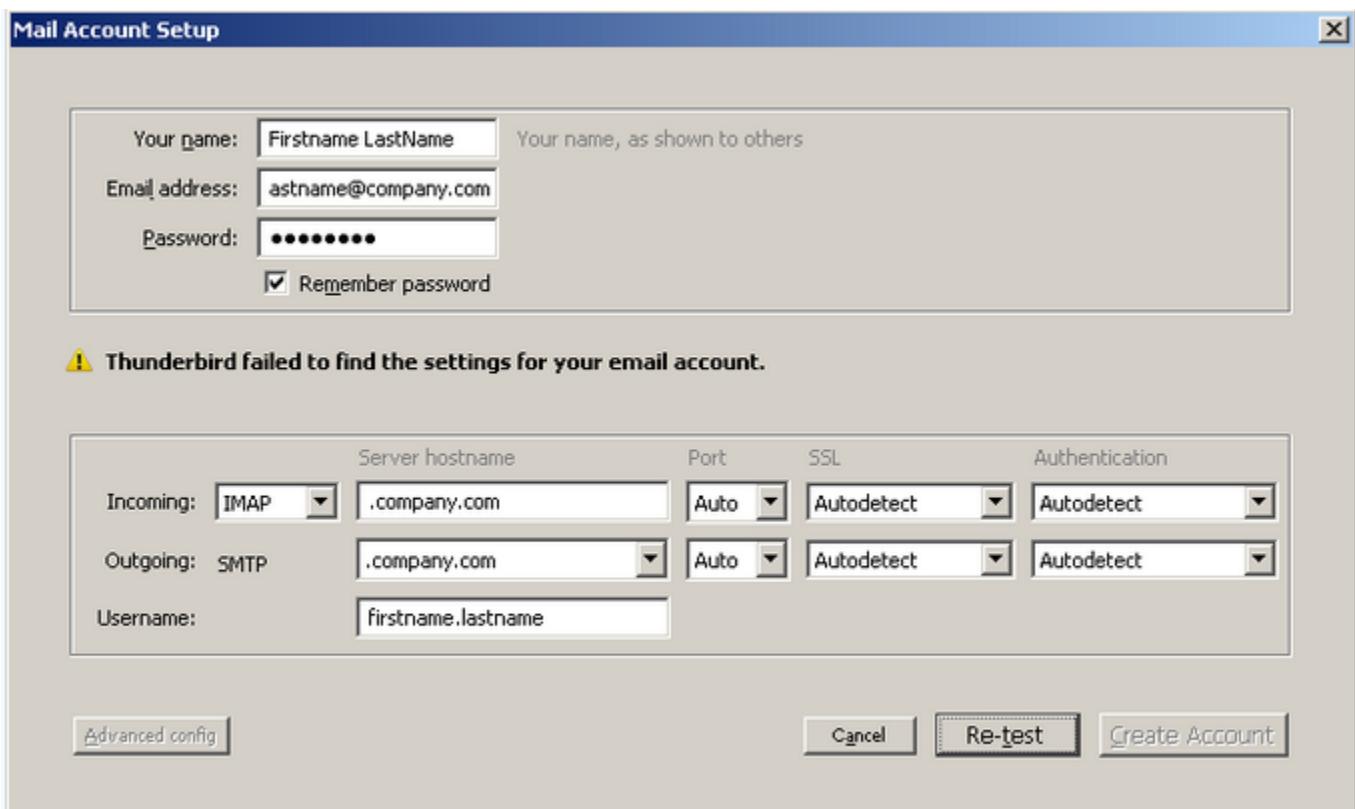
Local LDAP port: 1389 No SSL

Настройка почты OWA

создаю новый аккаунт



в дополнительных настройках указываю localhost, 1143, 1025, и без SSL и с Normal password



Mail Account Setup

Your name:	Firstname Lastname	Your name, as shown to others		
Email address:	astname@company.com			
Password:	*****			
<input checked="" type="checkbox"/> Remember password				

	Server hostname	Port	SSL	Authentication
Incoming: POP3	localhost	1110	None	Normal password
Outgoing: SMTP	localhost	1025	None	Normal password
Username:	first.lastname			

[Advanced config](#) [Re-test](#) [Create Account](#) [Cancel](#)

принимаем наш подставной SSL

Mail Account Setup



Warning!

Incoming settings: localhost does not use encryption.
▶ [Technical Details](#)

Outgoing settings: localhost does not use encryption.
▶ [Technical Details](#)

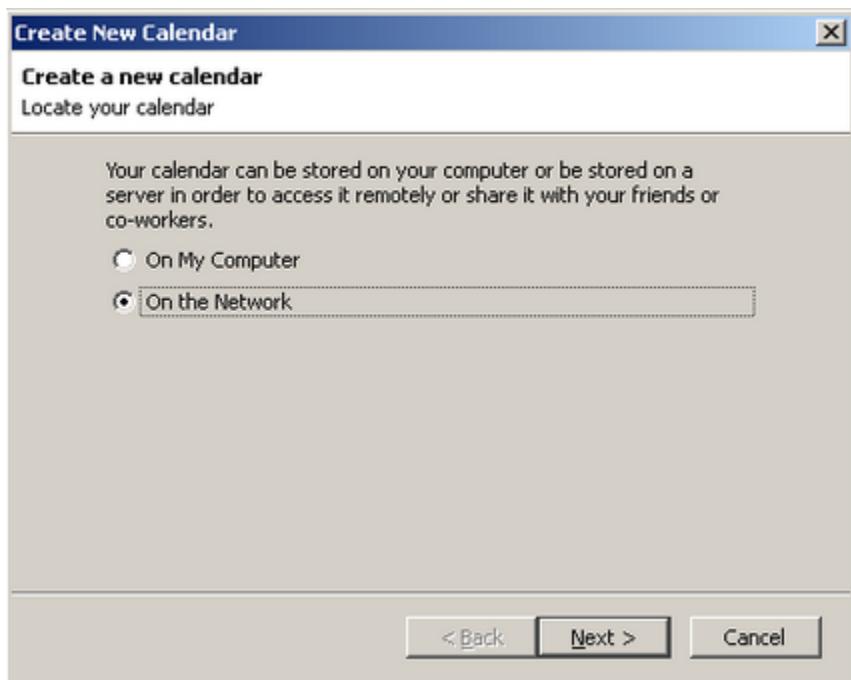
Thunderbird can allow you to get to your mail using the provided configurations. However, you should contact your administrator or email provider regarding these improper connections. See the Thunderbird FAQ for more information.

I understand the risks.

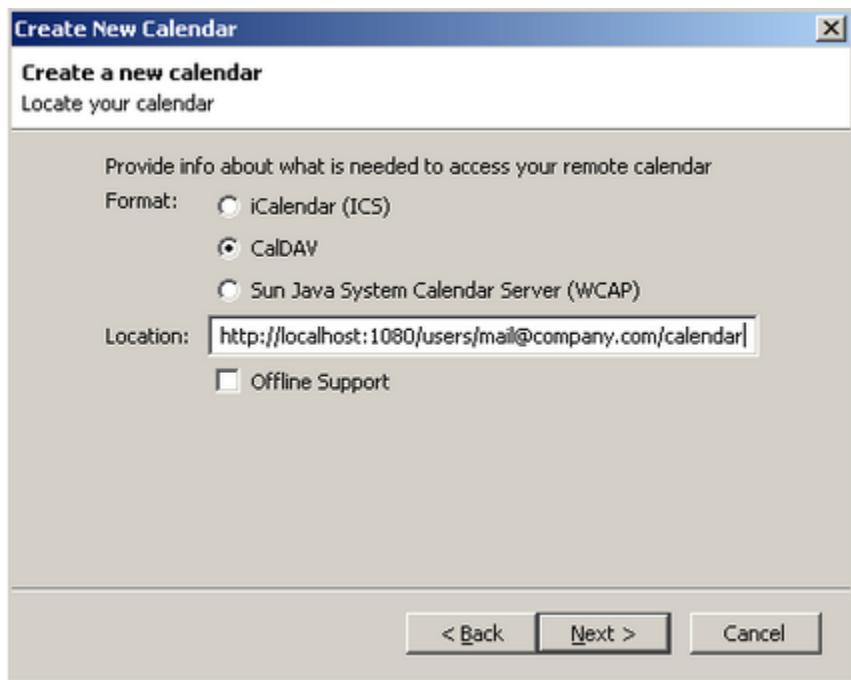
[Change Settings](#) [Create Account](#)

Настройка календаря OWA

Открываем календари и добавляем новый сетевой календарь



Добавляю calDAV с ссылкой <http://127.0.0.1:1080/users/mail@company.com/calendar>



даю имя и не выбираю никакой ящик

Create New Calendar

Create a new calendar
Customize your calendar

You can give your calendar a nickname and colorize the events from this calendar.

Name:

Color: 

Show Reminders:

E-Mail:

< Back Cancel

вводу свои учетные данные от календаря

Authentication Required

A username and password are being requested by <http://localhost:1080>. The site says: "Passerelle DavMail"

User Name:

Password:

Use Password Manager to remember this password.

OK Cancel

Create New Calendar

Create a new calendar
Calendar Created

Your calendar has been created.

< Back Cancel

BCE!

PostgreSQL + Pacemaker

Проверить мастер или слейв

f (false) - MASTER

t (true) - SLAVE

```
# sudo -u postgres psql -c "select pg_is_in_recovery();"
```

```
postgres=# select pg_is_in_recovery();
 pg_is_in_recovery
-----
 f
(1 row)
```

Проверить состояние репликации на Мастере

```
# sudo -u postgres psql -c "select client_addr, sync_state from
pg_stat_replication;"
```

Промоут Реплики до Мастера

```
# sudo -u postgres pg_ctl promote -D /var/lib/pgsql/data/
server promoting
```

Ошибка авторизации

```
$ psql
PostgreSQL error: Fatal: role "username" does not exist

$ sudo su - postgres
$ psql
postgres=#
```

Pacemaker

How Should the Configuration be Updated?

```
# cibadmin --query > tmp.xml  
# vim tmp.xml  
# cibadmin --replace --xml-file tmp.xml
```

Java key store

- Что такое Java Keystore?
 - Trust Store
 - Форматы Keystore поддерживаемые Java
 - Алиасы
- Создание keystore
- Генерация Ключей
 - Генерация при помощи openssl
 - 1. Создание пары ключей
 - 2. Конвертируем сертификат в формат
 - 3. Создание JKS файла
- Генерация при помощи keytool
 - Создание самоподписанного сертификата в новом/существующем хранилище
- Управление хранилищем ключей
 - Просмотр информации о хранилище
 - Использование Keytool для просмотра информации о сертификате

Что такое Java Keystore?

Keystore используется для хранения собственных приватных ключей и сертификатов сервера или клиента.

Для аутентификации клиента и сервера устанавливающих SSL соединение требуются приватные ключи и сертификаты. Если используется односторонняя аутентификация, то keystore нужен только на серверной стороне. При двусторонней аутентификации и клиент и сервер обмениваются сертификатами, соответственно и у сервера, и у клиента должен быть keystore с парой приватный ключ/публичный ключ + сертификат.

Trust Store

Второй тип keystore применяется для хранения trusted сертификатов. В него кладутся ключи trusted certificate authorities CA. При использовании самоподписанными сертификатами, в trusted store может лежать самоподписанный сертификат. Это тоже keystore, но в Java он называется trusted store.

Форматы Keystore поддерживаемые Java

Т.о., как описано выше, keystore — контейнер, используемый для хранения ключей и сертификатов. Java поддерживает два формата keystore:

- JKS (Java Key Store) – Java format
- PKCS12 — this is an industry standard

Тип keystore, используемый по-умолчанию, задается в Java security properties файле свойством keystore.type. Если приложение обращается к key store файлу без явного указания его типа, используется JKS формат. Java security properties файл расположен в каталоге lib внутри инсталляционного директория с Java по пути: /lib/security/java.security

Для работы с keystore в java дистрибутиве есть специальная утилита keytool. Keytool вполне достаточно для операций с ключами в Java. Однако JKS формат является проприетарным и закрытым. Поэтому часто для разнообразных конвертаций и взаимодействия со сторонними разработчиками могут использоваться утилиты, поставляемые в комплекте с библиотекой OpenSSL.

В тех случаях, когда планируется использовать ключи исключительно в Java keystore в формате JKS вполне подойдет.

Алиасы

Keystore (в JKS формате), позволяет хранить несколько пар ключей и сертификатов. Для идентификации каждой пары или отдельного сертификата используется алиас. Алиас указывается в исходном коде при доступе к соответствующему ключу или сертификату. Доступ к каждому алиасу ограничивается паролем.

Создание keystore

Процесс генерации keystore (JKS или PKS12) включает генерацию пары ключей (приватного и публичного). Затем получение от Certificate Authority (CA) подписи к публичному ключу и связанной с ним идентифицирующей информации в виде сертификата. Certificate authority генерирует сертификат на основе публичного ключа и идентификационной информации, переданной ему в виде CSR.

1) Сгенерировать пару ключей (public / private key)

В java при генерации пары ключей с помощью keytool сразу создается самоподписанный self-signed сертификат, который можно немедленно использовать для тестирования.

Следующие шаги, таким образом, нужны только для создания полноценного официального сертификата.

2) Сгенерировать запрос на получение сертификата (Certificate Signing Request (CSR)).

3) Получить CSR, подписанный доверенным CA (output of this is a certificate)

4) Импортировать сертификат, сделанный CA в ваш key store.

5) Импортировать сертификат CA в ваш truststore как trusted certificate

Генерация Ключей

Генерация при помощи openssl

1. Создание пары ключей

```
openssl req -x509 -nodes -sha256 -days 10000 -newkey rsa:2048 -keyout  
<key_name>.key -out <crt_name>.crt.
```

Данной командой мы генерируем: ключ и сертификат:

- В формате x509
- с шифрованием SHA
- Срок действия сертификата 10000 дней

2. Конвертируем сертификат в формат

```
openssl pkcs12 -export -in <crt_name>.crt -inkey <key_name>.key  
-certfile <crt_name>.crt -name "alias_name" -out <pkcs_name>.p12.
```

При конвертации нужно ввести **key-store secret** (пароль для сертификата)

3. Создание JKS файла

```
keytool -importkeystore -srckeystore <pkcs_name>.p12 -srcstoretype pkcs12 -destkeystore <jks_name>.jks -deststoretype JKS
```

При выполнении этой команды нужно будет задать **key-store password** которым будет зашифрован JKS файл

Генерация при помощи keytool

```
keytool -genkeypair \
-alias domain \
-keyalg RSA \
-keystore keystore.jks
```

Данная команда сгенерирует пару 2048-битных RSA-ключей под указанным псевдонимом (в данном случае это domain) в указанном файле хранилища (keystore.jks).

Создание самоподписанного сертификата в новом/существующем хранилище

```
keytool -genkey \
-alias domain \
-keyalg RSA \
-validity 365 \
-keystore keystore.jks
```

Данная команда создаст пару 2048-битных RSA-ключей, действительных на протяжении 365 дней, с указанным псевдонимом (domain) в заданном файле ключей (keystore.jks)

Управление хранилищем ключей

Просмотр информации о хранилище

```
keytool -list \
-keystore keystore.jks
```

Команда выводит список контрольных сумм всех сертификатов хранилища (keystore.jks) с соответствующими псевдонимами. Для более полной информации используйте ключ **-v**

keytool -list -v output example

```
keytool -list -v -keystore keystore.jks
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 1 entry
Alias name: mu-app
Creation date: Mar 14, 2017
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: O=Default Company Ltd, L=Default City, C=XX
Issuer: O=Default Company Ltd, L=Default City, C=XX
Serial number: 8b218a8a352bc9d5
Valid from: Tue Mar 14 07:21:59 CST 2017 until: Wed Mar 14 07:21:59 CST
2018
Certificate fingerprints:
MD5: F9:71:C6:70:D9:CA:65:81:06:0D:EC:CE:D5:4C:54:1D
SHA1: 25:E2:89:6C:D4:D1:29:48:C0:B7:91:22:F3:BA:AF:E8:5C:6B:F9:F9
SHA256:
AC:95:D2:F0:9F:D2:AE:A2:EB:F8:B2:BA:D8:62:2E:BA:68:60:18:4C:61:3A:9E:6C:
E1:30:30:54:EB:CB:A1:31
Signature algorithm name: SHA256withRSA
Version: 3
Extensions:
#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: BD 34 03 21 C8 47 BA EC 10 2B 04 F9 15 2D B4 5E .4.!..G...+....^
0010: AD DB 03 08 ....
]
]
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:true
PathLen:2147483647
]
#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BD 34 03 21 C8 47 BA EC 10 2B 04 F9 15 2D B4 5E .4.!..G...+....^
0010: AD DB 03 08 ....
]
]
```

```
keytool -printcert \
-file domain.crt
```

Данная команда выведет подробную информацию о файле сертификата (certificate.crt), в том числе контрольную сумму, distinguished name владельца и срок его действия: