Introduction

The Arclight project is a hosted (a.k.a. type-1) hypervisor.

Arclight is a server virtualization management solution based on KVM. It is designed to be a easy-to-use management platform allowing users to create and manage virtual machines (VMs) on Linux servers. Arclight utilizes the Libvirt API, All of the actions you would expect from a virtualization management tool are included in the software. For example, user can create, clone and manage VMs, storage pools networks and volumes. When it comes to networking, there are multiple options available. Users create private networks for there VMs and have the option to control DHCP within the private network. In addition to private networks, VMs can also use bridged connections, connecting them directly to the network interfaces on the physical server. Manage virtual machines directly from Arclight. There is no need to install additional VNC software. [About this project]: This project is in-development and we are still adding features to it along with complete ISO package and APIs for Enterprise Usage.

- Source code
    - GitHub mirror
    - API documentation, useful for searching API.
- Issue tracker

Pre-installation Checklist

Check that your CPU supports hardware virtualization

To run Arclight, you need a processor that supports hardware virtualization. Intel and AMD both have developed extensions for their processors, deemed respectively Intel VT-x (code name Vanderpool) and AMD-V (code name Pacifica). To see if your processor supports one of these, you can review the output from this command:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

> If 0 it means that your CPU doesn't support hardware virtualization.
>
> If 1 or more it does - but you still need to make sure that virtualization is enabled in the BIOS.

Alternatively, you may execute:

```
kvm-ok
```

which may provide an output like this:

```
INFO: /dev/kvm exists KVM acceleration can be used
```

Use a 64 bit kernel (if possible)

Running a 64 bit kernel on the host operating system is recommended but not required.

1. To serve more than 2GB of RAM for your VMs, you must use a 64-bit kernel. On a 32-bit kernel install, you'll be limited to 2GB RAM at maximum for a given VM.
2. Also, a 64-bit system can host both 32-bit and 64-bit guests. A 32-bit system can only host 32-bit guests.

To see if your processor is 64-bit, you can run this command:

```
egrep -c ' lm ' /proc/cpuinfo
```

If 0 is printed, it means that your CPU is not 64-bit.

If 1 or higher, it is. Note: lm stands for Long Mode which equates to a 64-bit CPU.

Now see if your running kernel is 64-bit, just issue the following command:

```
uname -m
```

x86_64 indicates a running 64-bit kernel. If you use see i386, i486, i586 or i686, you're running a 32-bit kernel. Note: x86_64 is synonymous with amd64.

Getting Started

This chapter includes how to set up Arclight on each platform.

Installation on Ubuntu Server

Before installing software, run the `sudo apt` update command to make sure you are installing from the latest repository information.

Installing the necessary packages On the Ubuntu server, install the QEMU + KVM hypervisor using the following command:

```
sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils
virt-manager xauth
```

Install the web server, database, and necessary PHP packages to your server. Use the following command:

```
sudo apt install apache2 mysql-server php libapache2-mod-php php-mysql php-xml
php-libvirt-php
```

The built-in VNC connection requires python. To install it use the following command:

```
sudo apt install python
```

Configuring files and permissions

To use VNC to connect into your virtual machines, you will need to edit the /etc/libvirt /qemu.conf file. Be sure to allow listening on IP address 0.0.0.0 by uncommenting the line #vnc_listen = “0.0.0.0” and saving the file.

```
sudo nano /etc/libvirt/qemu.conf
```

The web server user account on Ubuntu is called www-data. This account will need to have permissions to work with libvirt. The group is called libvirtd in Ubuntu 16.04 and libvirt in Ubuntu 18.04. To do this, add the www-data user to the necessary group.

```
sudo adduser www-data libvirt
```

Change your directory location to the root directory of your web server. The default location is /var/www/html/ in Ubuntu.

```
cd /var/www/html
```

Now download the latest version of Arclight Dashboard to the web root directory.

```
wget https://github.com/Chatnaut/Arclight/archive/refs/tags/v1.0.0.tar.gz
```

Extract the downloaded package.

```
sudo tar -xzf v1.0.0.tar.gz
```

Rename the extracted directory

```
sudo mv Arclight-1.0.0 arclight
```

Change the ownership of the arclight directory to the web server user (www-data).

```
sudo chown -R www-data:www-data /var/www/html/arclight
```

Creating a database

We will need a MySQL database for Arclight Dashboard to work with. To log into MySQL use the following command:

```
sudo mysql -u root
```

Once logged in, create a new database. We will name it arclight.

```
CREATE DATABASE arclight;
```

Now create a user for Arclight Dashboard to use. You could use the root user and password, but that is never advised. We will create a new user named arclight. Be sure to change the password value.

```
CREATE USER 'arclight'@'localhost' IDENTIFIED BY 'password';
```

Change the permissions of the new user to have full access to the database tables.

```
GRANT ALL PRIVILEGES ON arclight.* to 'arclight'@'localhost';
```

The new privileges should be applied, but sometimes you will need to flush the privileges so that they can be reloaded into the MySQL database. To do this use the following command:

```
FLUSH PRIVILEGES;
```

To exit MySQL, type quit or use the EXIT; statement.

```
EXIT;
```

Connecting to Arclight Dashboard

You will need to restart your server before you can use the hypervisor. This way the server restarts with all the necessary hypervisor packages loaded and the user groups applied `sudo reboot`.

Once rebooted, use a web browser to navigate to your server's IP address or domain name. Add /arclight to the end of the URL. For example: http://192.168.1.2/arclight

Installation on CentOS Server

This guide follows a fresh installation of the CentOS 7 minimal server. Before installing packages be sure to update repository information using the following command:

```
yum update -y
```

Installing the necessary packages of QEMU + KVM by using the following command:

```
yum install qemu-kvm libvirt -y
```

The PHP Libvirt extension is located in the Enterprise Linux repository. To setup this repository use the following command:

```
yum install epel-release -y
```

Install the web server, database, and necessary PHP packages to your server. Use the following command:

```
yum install httpd mariadb-server mariadb php php-mysql php-xml php-libvirt -y
```

You will need to start and enable the Apache web server and Maria database. To do this use the following commands:

```
systemctl start mariadb
systemctl enable mariadb
systemctl start httpd
systemctl enable httpd
```

Configuring files and permissions

To use VNC to connect into your virtual machines, you will need to edit the /etc/libvirt /qemu.conf file. Be sure to allow listening on IP address 0.0.0.0 by uncommenting the line #vnc_listen = “0.0.0.0” and saving the file.(If nano is not installed you can install it with yum install nano, or just simply use vi instead of nano).

```
nano /etc/libvirt/qemu.conf
```

The web server user account on CentOS is called apache. This account will need to have permissions to work with libvirt. We can do this by adding the apache user to the libvirt group. To do this, use the following command:

```
usermod -a -G libvirt apache
```

Change your directory location to the root directory of your web server. The default location is /var/www/html/ in Ubuntu.

```
cd /var/www/html
```

The minimal installation of CentOS does not come with wget to download files. You will also need git to perform software updates. Install the, using the following command:

```
yum install wget git -y
```

Now download the latest version of Arclight Dashboard to the web root directory.

```
wget https://github.com/arclight/arclight/archive/v1.0.0.tar.gz
```

Extract the downloaded package.

```
sudo tar -xzf v1.0.0.tar.gz
```

Rename the extracted directory

```
sudo mv arclight-1.0.0 arclight
```

Change the ownership of the arclight directory to the web server user (www-data).

```
chown -R apache:apache /var/www/html/arclight
```

In order for PHP to be able to save configuration files we will need to run the following command:

```
chown -t httpd_sys_rw_content_t /var/www/html/arclight/ -R
```

The CentOS firewall will block incoming http and https traffic. Also the VNC connection uses port 6080. To allow the web traffic use the following commands:

```
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https
firewall-cmd --permanent --add-port=6080/tcp
systemctl restart firewalld
```

SeLinux will block the qemu connection through the web browser. Modify the /etc/sysconfig /selinux file. The default value of the SELINUX=enforcing. Change it to SELINUX=permissive.

```
nano /etc/sysconfig/selinux
```

Creating a database

We will need a MySQL database for Arclight Dashboard to work with. To log into MySQL use the following command:

```
sudo mysql -u root
```

Once logged in, create a new database. We will name it arclight.

```
CREATE DATABASE arclight;
```

Now create a user for Arclight Dashboard to use. You could use the root user and password, but that is never advised. We will create a new user named arclight. Be sure to change the password value.

```
CREATE USER 'arclight'@'localhost' IDENTIFIED BY 'password';
```

Change the permissions of the new user to have full access to the database tables.

```
GRANT ALL PRIVILEGES ON arclight.* to 'arclight'@'localhost';
```

The new privileges should be applied, but sometimes you will need to flush the privileges so that they can be reloaded into the MySQL database. To do this use the following command:

```
FLUSH PRIVILEGES;
```

To exit MySQL, type quit or use the EXIT; statement.

```
EXIT;
```

Connecting to Arclight Dashboard

You will need to restart your server before you can use the hypervisor. This way the server restarts with all the necessary hypervisor packages loaded and the user groups applied `sudo reboot`.

Once rebooted, use a web browser to navigate to your server's IP address or domain name. Add /arclight to the end of the URL. For example: http://192.168.1.2/arclight

Encrypt Arclight

This chapter includes how to encrypt Arclight console either using Let's Encrypt or by Self-signed certificate

Encrypting Arclight with Let’s Encrypt

As a security recommendation, it is always a good practice to encrypt the data sent across the Internet. You can encrypt both your arclight connection as well as the VNC console connection to your virtual machines. With the Apache web server on Ubuntu you can enable HTTPS traffic using the following command:

```
sudo a2enmod ssl
```

If you are using a domain name, you can use a Certificate Authority such as Let’s Encrypt to create a free validated SSL certificate. To get started we will need to create an Apache site configuration file for your domain. We will using the domain mydomain.com for this example. The new config file should end with the .conf extension and be located in the `/etc/apache2/sites-available/` directory. To create a new file for your domain use the following command, and be sure to change the domain name:

```
sudo nano /etc/apache2/sites-available/mydomain.com.conf
```

We will just be adding just the minimum information in the configuration file. The first line below `<VirtualHost *:80>` tells Apache that this configuration file will be used for HTTP traffic. When we configure Let’s Encrypt, the HTTPS connection (port 443) will be configured automatically. The second line ServerName mydomain.com tells Apache what domain name it should be listening for to apply this configuration. The third line DocumentRoot `/var/www/html/arclight/` indicates the root location of the web site files and that should be the filepath for your files.

```
<VirtualHost *:80>
 ServerName mydomain.com
 DocumentRoot /var/www/html/arclight/
</VirtualHost>
```

Once you add the above information to the configuration file and save it, we will then need to enable the configuration file in Apache using the a2ensite command. To do that run the following command, be sure to use your domain name:

```
sudo a2ensite mydomain.com
```

When Apache is only used for the arclight it would be a good idea to disable the default configuration file that comes with the install of Apache. To do that use the command:

```
sudo a2dissite 000-default.conf
```

You will need to restart/reload the Apache web server to apply the configuration changes. Use the following command:

```
sudo systemctl reload apache2
```

To automate the Let's Encrypt certificate using Apache we will need to install the python3-certbot-apache package. Use the following command:

```
sudo apt install python3-certbot-apache
```

To create the SSL Certificate and Apache configuration file run the following command, changing your domain name. You will be asked for an email address and you will be given an option to either redirect all traffic to the HTTPS protocol or not.

```
sudo certbot --apache -d mydomain.com
```

Now login to your Arclight Dashboard. Go to the settings page and add the location of the Let's Encrypt certificate file and key file and submit your changes. Below is the location created for mydomain.com Certificate file: `/etc/letsencrypt/live/mydomain.com/fullchain.pem` Key file: `/etc/letsencrypt/live/mydomain.com/privkey.pem` The permissions for the certificates are tied to the root user. There will need to be a permission change on the /etc/letsencrypt/live folder as well as /etc/letsencrypt/archive. We can change the permission to 755 (rwxr-xr-x) to allow the Arclight to be able to read the information. Run the following commands:

```
sudo chmod 755 /etc/letsencrypt/live
```

```
sudo chmod 755 /etc/letsencrypt/archive
```

You can either decide to restart your server or restart the python process tied to noVNC to apply the certificate and key files. If you decide to restart the service you should be able to determine which process id (PID) is using port 6080. Use the following command:

```
sudo netstat -tulpn | grep 6080
```

Then after determining the PID number, kill the process. For example, if it was PID 1386, I would use the command:

```
sudo kill 1386
```

Now logout and login to the arclight to restart the VNC connection and the new certificate should be applied.

Encrypting Arclight with self-signed certificate

As a security recommendation, it is always a good practice to encrypt your the data sent across the Internet. You can encrypt both your arclight connection as well as the VNC connection to your virtual machines.With the Apache web server on Ubuntu you can enable https traffic using the following command:

```
sudo a2enmod ssl
```

Ubuntu has a configuration already setup to be used with a self-signed certificate. It can be activated by using the following command:

```
sudo a2ensite default-ssl.conf
```

You will need to restart/reload the Apache web server to apply the SSL connection. Use the following command:

```
sudo systemctl restart apache2
```

The VNC connection will default to using the protocol of you web connection. If you wish to use https with VNC you will need to create a certificate. By default, the noVNC app that comes with arclight looks for a cert called self.pem in the `/etc/ssl/` directory.To create the certificate for the VNC connection navigate to the `/etc/ssl/` directory.

```
cd /etc/ssl/
```

Create the certificate by using the following command:

```
sudo openssl req -x509 -days 365 -new -nodes -out self.pem -keyout self.pem
```

Now change the permissions of the self.pem file

```
sudo chmod 755 self.pem
```

If you have already used arclight, you will need to kill the existing VNC process. To determine the process to kill use netstat and determine the process number that is listening on port 6080.

```
sudo netstat -tulpn | grep 6080
```

Now kill the process. For example if the process was numbered 29226, you would kill it using the command:

```
sudo kill 29226
```

Now when you log into arclight, the VNC software will use the self-signed cert. Because it is self-signed your browser will not trust it. To trust the certification visit your URL:6080 and click the Advanced button on the screen. For example, if you were using 192.168.1.2 to view the web interface you should use https://192.168.1.2:6080.

Add Custom Storage Pools

Using arclight, you can define Libvirt storage pools in the /var, /mnt, and /media directories. This was done to prevent full access to the operating system from the Web interface. If you need to define a storage pool outside of these limitations, you can use the terminal using Libvirt to register a storage pool. In this example we will define the /home/ubuntu/ directory as a storage pool.

Define the storage pool using the pool-define-as command from virsh. We will pass in the type of storage devices which is a directory, name which we will call myHomePool, and the filepath to the storage pool.

```
virsh pool-define-as --type dir --name myHomePool --target /home/ubuntu
```

The storage pool will now show up in arclight. If you wish to view it in the terminal you can use the following command

```
virsh pool-list --all
```

The storage pool myHomePool will not be running, you can start it using arclight, or in the terminal you can use the following command to start the storage pool. Optionally you can use pool-autostart to automatically start the pool upon the system boot and use pool-autostart –disable to remove it.

```
virsh pool-start myHomePool
```

If you choose to stop the storage pool from running, you can do this in arclight or by using the pool-destroy option.

```
virsh pool-destroy myHomePool
```

Lastly if you decide to remove the storage pool you can undefine it. This will leave the directory intact on the operating system, just removing it from the list of storage pools. Again, this can be done in arclight or by using the pool-undefine option in the terminal.

```
virsh pool-undefine myHomePool
```

ISO images for KVM machines

When getting started with KVM virtual machines, one common question is how do I get ISO image files used to install the operating systems in the virtual machines. The default location that Libvirt uses as a storage pool for KVM virtual machines is the `/var/lib/libvirt/images/` directory. You will need to download the ISO files using a command such as wget. Find the URL of the ISO from from the vendor, for example http://releases.ubuntu.com/18.04.1/ubuntu-18.04.1-live-server-amd64.iso

You will need to switch your user account to the root user: Navigate to the `/var/lib/libvirt/images/` directory:

```
cd /var/lib/libvirt/images/
```

Use wget to download the file:

```
wget http://releases.ubuntu.com/18.04.1/ubuntu-18.04.1-live-server-amd64.iso
```

The ISO file will now show up in arclight.

API Documentation

Total number of functions: 159. Functions supported are:

libvirt_get_last_error()

libvirt_connect($url, $readonly, $credentials)

libvirt_node_get_info($conn)

libvirt_node_get_cpu_stats($conn, $cpunr)

libvirt_node_get_cpu_stats_for_each_cpu($conn, $time)

libvirt_node_get_mem_stats($conn)

libvirt_connect_get_machine_types($conn)

libvirt_connect_get_information($conn)

libvirt_connect_get_uri($conn)

libvirt_connect_get_hostname($conn)

libvirt_image_create($conn, $name, $size, $format)

libvirt_image_remove($conn, $image)

libvirt_connect_get_hypervisor($conn)

libvirt_connect_is_encrypted($conn)

libvirt_connect_is_secure($conn)

libvirt_connect_get_all_domain_stats($conn, $stats, $flags)

libvirt_connect_get_maxvcpus($conn)

libvirt_connect_get_sysinfo($conn)

libvirt_domain_get_counts($conn)

libvirt_domain_is_persistent($res)

libvirt_domain_set_max_memory($res, $memory)

libvirt_domain_set_memory($res, $memory) libvirt_domain_set_memory_flags($res, $memory, $flags)

libvirt_domain_get_autostart($res)

libvirt_domain_set_autostart($res, $flags)

libvirt_domain_get_metadata($res, $type, $uri, $flags)

libvirt_domain_set_metadata($res, $type, $metadata, $key, $uri, $flags)

libvirt_domain_is_active($res)

libvirt_domain_lookup_by_name($res, $name)

libvirt_domain_lookup_by_uuid($res, $uuid)

libvirt_domain_qemu_agent_command($res, $cmd, $timeout, $flags)

libvirt_domain_lookup_by_uuid_string($res, $uuid)

libvirt_stream_create($res)

libvirt_stream_close($res)

libvirt_stream_abort($res)

libvirt_stream_finish($res)

libvirt_stream_recv($res, $data, $len)

libvirt_stream_send($res, $data, $length)

libvirt_domain_lookup_by_id($conn, $id)

libvirt_domain_get_name($res)

libvirt_domain_get_uuid_string($res)

libvirt_domain_get_screenshot_api($res, $screenID)

libvirt_domain_get_screenshot($res, $server, $scancode)

libvirt_domain_get_screen_dimensions($res, $server)

libvirt_domain_send_keys($res, $server, $scancode)

libvirt_domain_send_pointer_event($res, $server, $pos_x, $pos_y, $clicked, $release) libvirt_domain_get_uuid($res)

libvirt_domain_get_id($res)

libvirt_domain_get_next_dev_ids($res)

libvirt_connect_get_capabilities($conn, $xpath)

libvirt_connect_get_emulator($conn, $arch)

libvirt_connect_get_nic_models($conn, $arch)

libvirt_connect_get_soundhw_models($conn, $arch, $flags)

libvirt_domain_new($conn, $name, $arch, $memMB, $maxmemMB, $vcpus, $iso_image, $disks, $networks, $flags)

libvirt_domain_new_get_vnc()

libvirt_domain_get_xml_desc($res, $xpath)

libvirt_domain_get_disk_devices($res)

libvirt_domain_get_interface_devices($res)

libvirt_domain_change_vcpus($res, $numCpus, $flags)

libvirt_domain_change_memory($res, $allocMem, $allocMax, $flags)

libvirt_domain_change_boot_devices($res, $first, $second, $flags)

libvirt_domain_disk_add($res, $img, $dev, $typ, $driver, $flags)

libvirt_domain_disk_remove($res, $dev, $flags)

libvirt_domain_nic_add($res, $mac, $network, $model, $flags)

libvirt_domain_nic_remove($res, $dev, $flags)

libvirt_domain_get_info($res)

libvirt_domain_create($res)

libvirt_domain_destroy($res)

libvirt_domain_resume($res)

libvirt_domain_core_dump($res, $to)

libvirt_domain_shutdown($res)

libvirt_domain_managedsave($res)

libvirt_domain_suspend($res)

libvirt_domain_undefine($res)

libvirt_domain_reboot($res, $flags)

libvirt_domain_define_xml($conn, $xml)

libvirt_domain_create_xml($conn, $xml)

libvirt_domain_memory_peek($res, $start, $size, $flags)

libvirt_domain_memory_stats($res, $flags)

libvirt_domain_update_device($res, $xml, $flags)

libvirt_domain_block_stats($res, $path)

libvirt_domain_block_resize($res, $path, $size, $flags)

libvirt_domain_block_commit($res, $disk, $base, $top, $bandwidth, $flags)

libvirt_domain_block_job_abort($res, $path, $flags)

libvirt_domain_block_job_set_speed($res, $path, $bandwidth, $flags)

libvirt_domain_get_network_info($res, $mac)

libvirt_domain_get_block_info($res, $dev)

libvirt_domain_xml_xpath($res, $xpath, $flags)

libvirt_domain_interface_stats($res, $path)

libvirt_domain_get_connect($res)

libvirt_domain_migrate_to_uri($res, $dest_uri, $flags, $dname, $bandwidth)

libvirt_domain_migrate_to_uri2($res, $dconnuri, $miguri, $dxml, $flags, $dname, $bandwidth)

libvirt_domain_migrate($res, $dest_conn, $flags, $dname, $bandwidth)

libvirt_domain_get_job_info($res)

libvirt_domain_has_current_snapshot($res, $flags)

libvirt_domain_snapshot_lookup_by_name($res, $name, $flags)

libvirt_domain_snapshot_create($res, $flags)

libvirt_domain_snapshot_get_xml($res, $flags)

libvirt_domain_snapshot_revert($res, $flags)

libvirt_domain_snapshot_delete($res, $flags)

libvirt_list_domain_snapshots($res, $flags)

libvirt_storagepool_lookup_by_name($res, $name)

libvirt_storagepool_lookup_by_volume($res)

libvirt_storagepool_list_volumes($res)

libvirt_storagepool_get_info($res)

libvirt_storagevolume_lookup_by_name($res, $name)

libvirt_storagevolume_lookup_by_path($res, $path)

libvirt_storagevolume_get_name($res)

libvirt_storagevolume_get_path($res)

libvirt_storagevolume_get_info($res)

libvirt_storagevolume_get_xml_desc($res, $xpath, $flags)

libvirt_storagevolume_create_xml($res, $xml, $flags)

libvirt_storagevolume_create_xml_from($pool, $xml, $original_volume)

libvirt_storagevolume_delete($res, $flags)

libvirt_storagevolume_resize($res, $capacity, $flags)

libvirt_storagevolume_download($res, $stream, $offset, $length, $flags)

libvirt_storagevolume_upload($res, $stream, $offset, $length, $flags)

libvirt_storagepool_get_uuid_string($res) libvirt_storagepool_get_name($res)

libvirt_storagepool_lookup_by_uuid_string($res, $uuid)

libvirt_storagepool_get_xml_desc($res, $xpath)

libvirt_storagepool_define_xml($res, $xml, $flags)

libvirt_storagepool_undefine($res)

libvirt_storagepool_create($res)

libvirt_storagepool_destroy($res)

libvirt_storagepool_is_active($res)

libvirt_storagepool_get_volume_count($res)

libvirt_storagepool_refresh($res, $flags)

libvirt_storagepool_set_autostart($res, $flags)

libvirt_storagepool_get_autostart($res)

libvirt_storagepool_build($res)

libvirt_storagepool_delete($res)

libvirt_list_storagepools($res)

libvirt_list_active_storagepools($res)

libvirt_list_inactive_storagepools($res)

libvirt_list_domains($res)

libvirt_list_domain_resources($res)

libvirt_list_active_domain_ids($res)

libvirt_list_active_domains($res)

libvirt_list_inactive_domains($res)

libvirt_list_networks($res, $flags)

libvirt_list_nodedevs($res, $cap)

libvirt_nodedev_get($res, $name)

libvirt_nodedev_capabilities($res)

libvirt_nodedev_get_xml_desc($res, $xpath)

libvirt_nodedev_get_information($res)

libvirt_network_define_xml($res, $xml)

libvirt_network_undefine($res)

libvirt_network_get($res, $name)

libvirt_network_get_bridge($res)

libvirt_network_get_active($res)

libvirt_network_get_information($res)

libvirt_network_set_active($res, $flags)

libvirt_network_get_xml_desc($res, $xpath)

libvirt_version($type)

libvirt_check_version($major, $minor, $micro, $type)

libvirt_has_feature($name)

libvirt_get_iso_images($path)

libvirt_print_binding_resources()

libvirt_logfile_set($filename, $maxsize)

Full Libvirt API Documentation is available at here.