




# Cloud Computing

**BITS Pilani**



# Cloud Infrastructure Management using OpenNebula

# Cloud ?



# That looks easy!!!

## The Cloud



Is it So?????

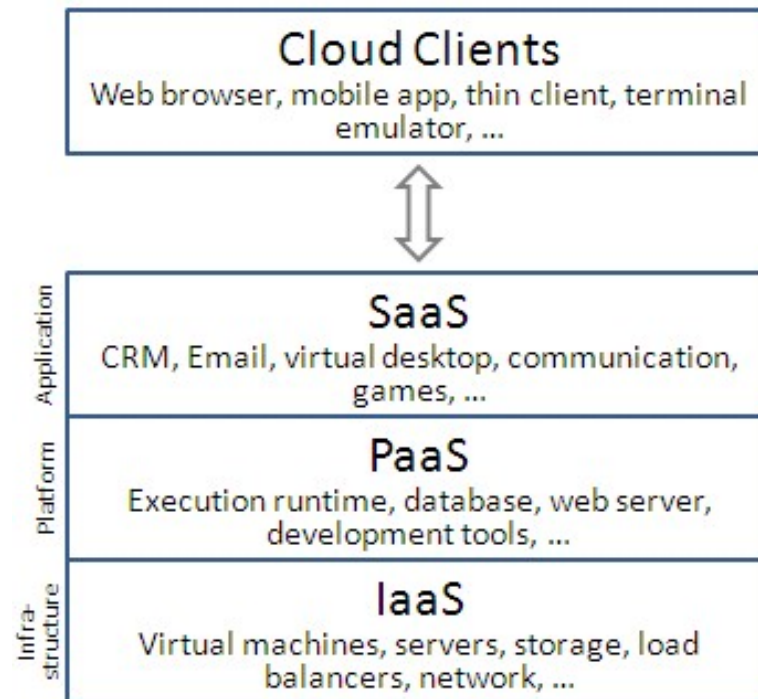
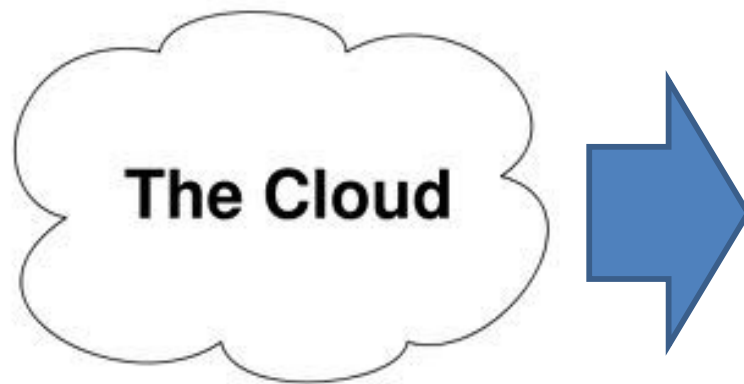


# Introduction



So CLOUD requires managing.....

But what will you manage

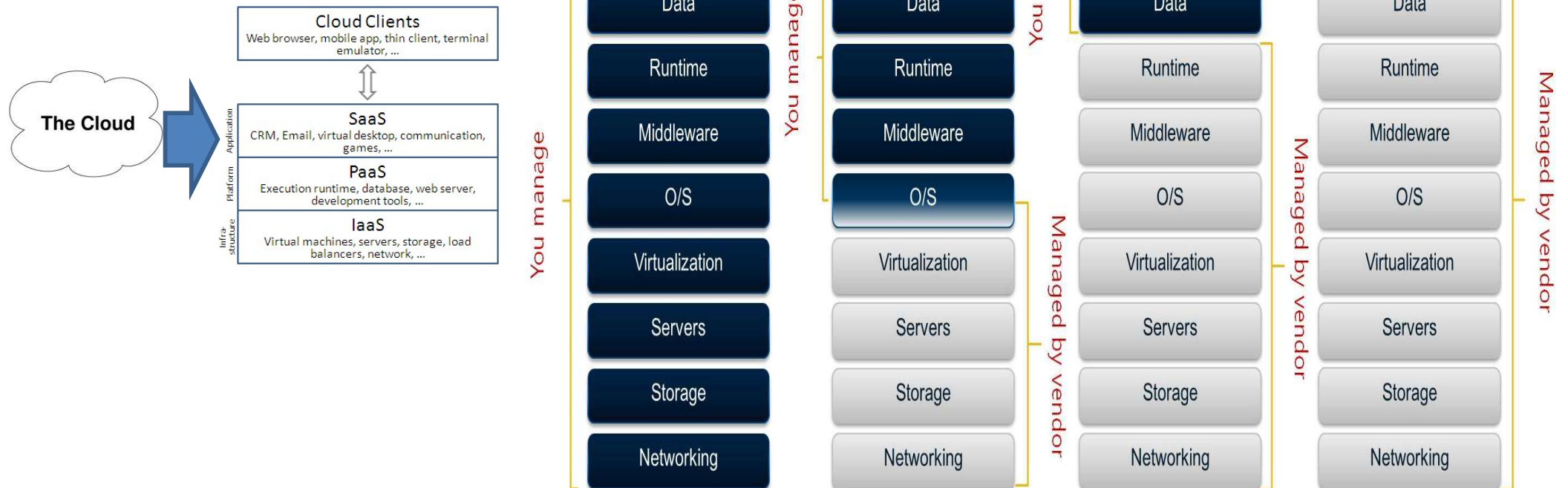


# Introduction



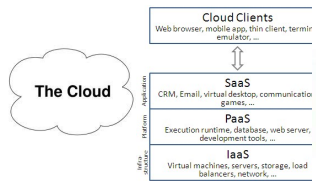
Ok, ok got to know what to manage.....

Is it so.....????

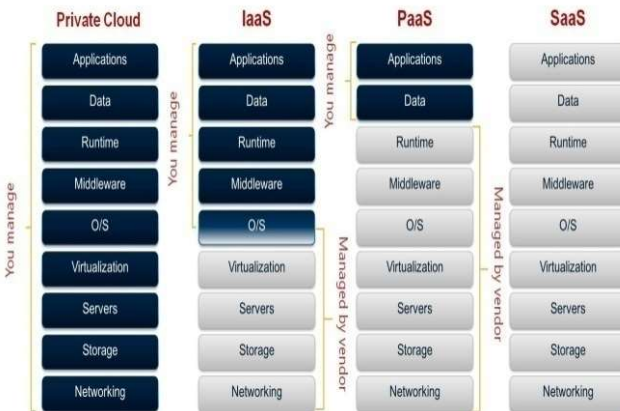




• OMG!!!!



Therefore the  
for cloud architecture is  
"efficient management"  
of resources at all the  
three layers of cloud stack

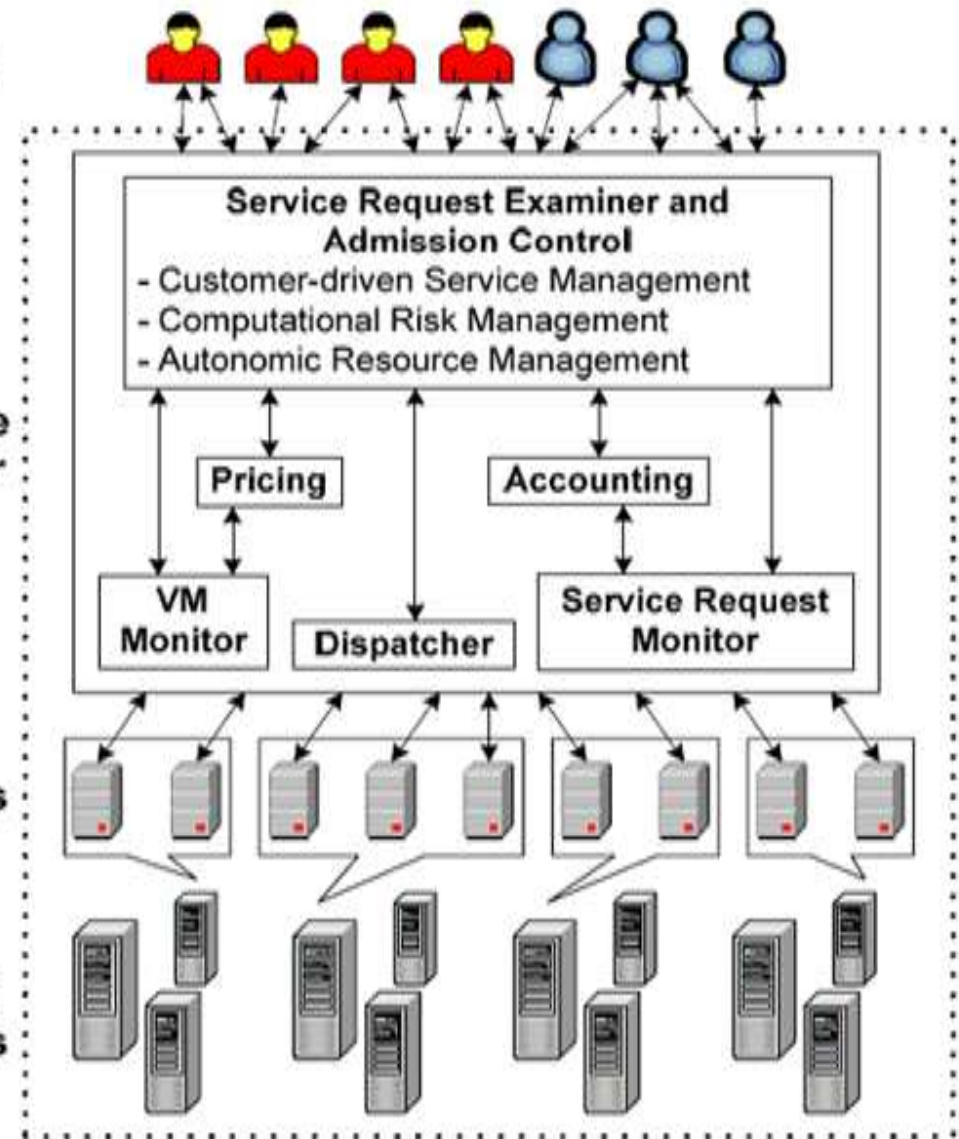


Users/  
Brokers

SLA  
Resource  
Allocator

Virtual  
Machines  
(VMs)

Physical  
Machines



---

- Cloud Distributed environment

- With large scale of systems to manage
- Support of multi-tenancy
- Management to maintain SLAs

- So there is need for automation to replace manual operations and to reduce overall cost



# Outline

---

- Virtual infrastructure managers
- Architectural view on OpenNebula
- Constructing a Private cloud
- Virtual Machines in OpenNebula
- Constructing a Hybrid cloud



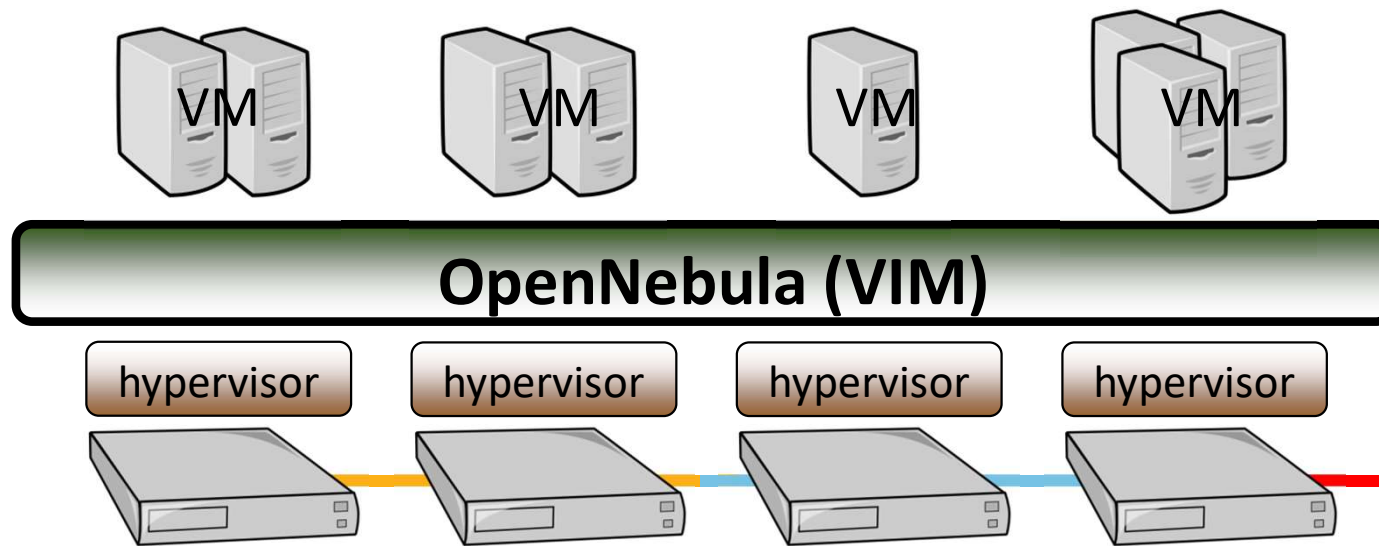


Need of the hour???

# Virtual Infrastructure Managers

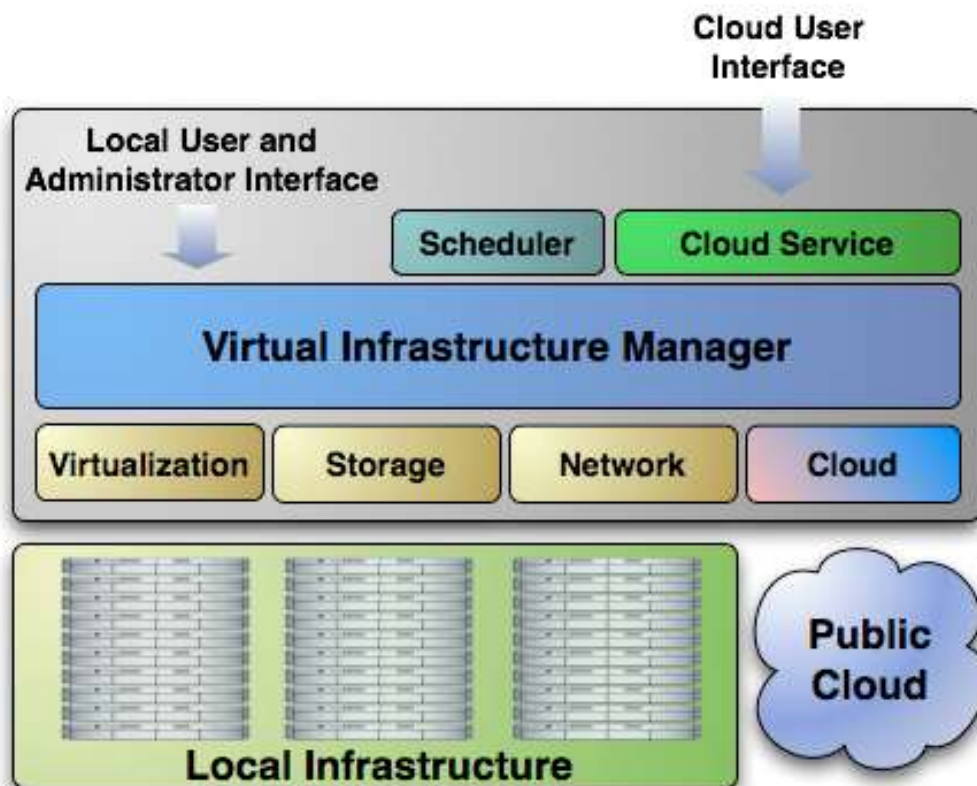
# Why a Virtual Infrastructure Manager?

- VMs are great!!...but something more is needed
  - Where did/do I put my VM? (**scheduling & monitoring**)
  - How do I provision a new cluster node? (**clone**)
  - What IP addresses are available? (**networking**)
- Provide a **uniform view** of the resource pool
- **Life-cycle management** and monitoring of VM
- The VIM should **integrate** Image, Network and Virtualization



# Extending the Benefits of Virtualization to Clusters

- Dynamic deployment and re-placement of virtual machines on a pool of physical resources
- Transform a rigid distributed physical infrastructure into a flexible and agile virtual infrastructure

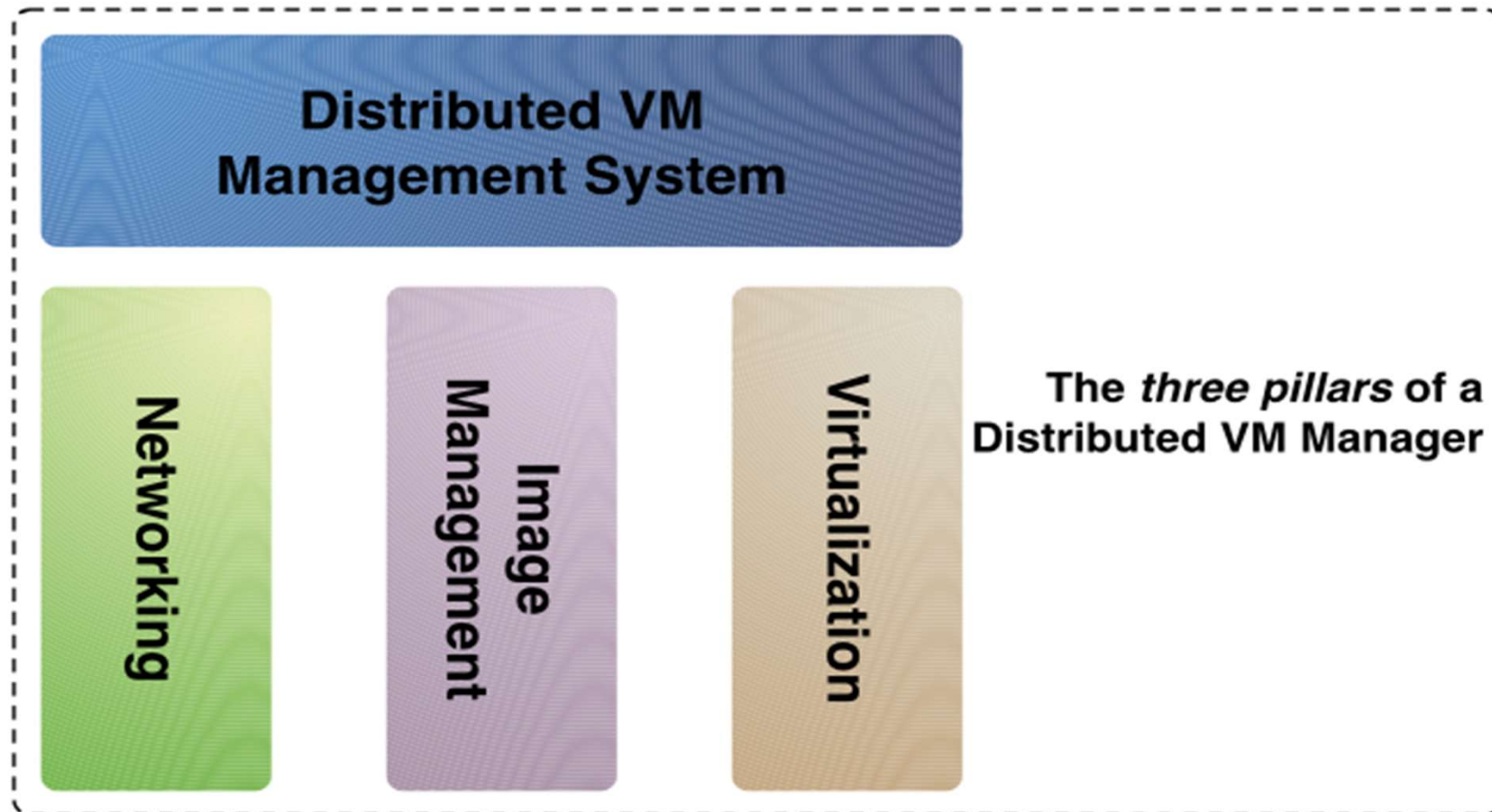


- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

# Virtual Machine Management Model

---

## Distributed VM Management Model



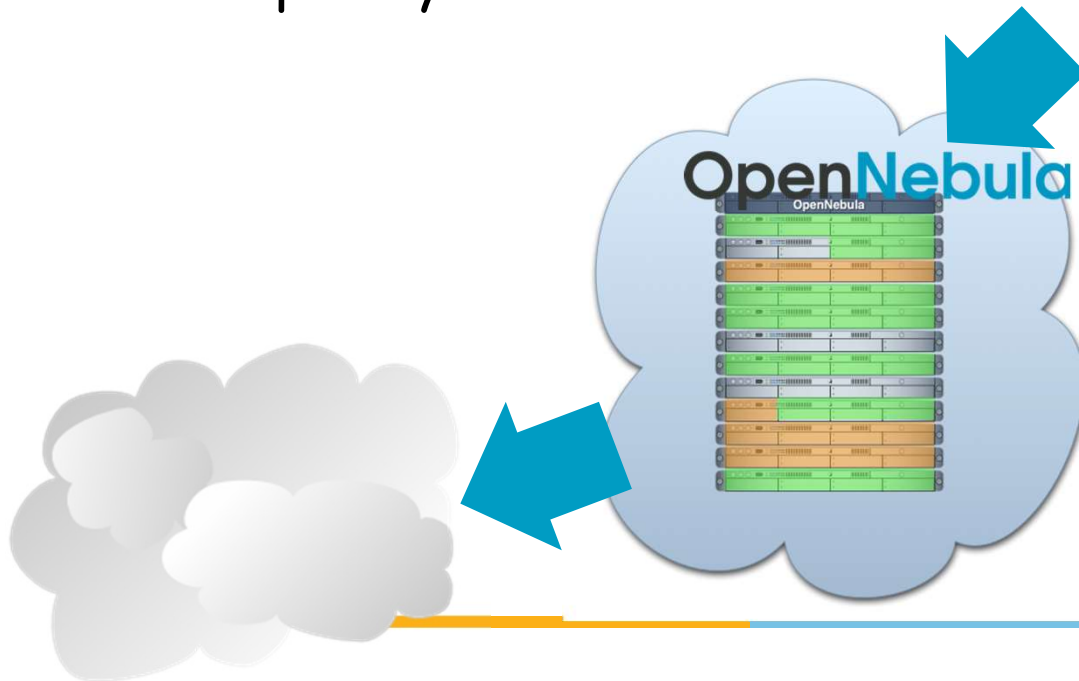


# What is OpenNebula?

# What is OpenNebula?

## Enabling Technology to Build your Cloud

- Private Cloud to simplify and optimize internal operations
- Hybrid Cloud to supplement the capacity of the Private Cloud
- Public Cloud to expose your Private to external users



# What is the OpenNebula Open-Source Project?

Building the Industry Standard Open Source Cloud Computing Tool

Lead Innovation in Enterprise-Class Cloud Computing Management

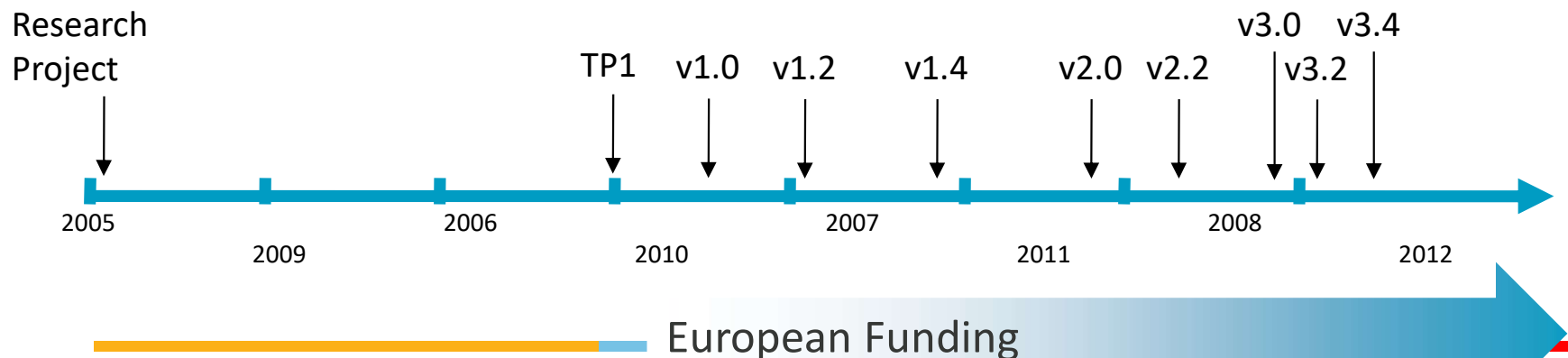
- Develop, maintain and assure the quality of OpenNebula
- Collaborate with open-source and research projects and communities
- Support the community and the ecosystem

An Active and Engaged Community

- 4,000 downloads/month
- 100 active contributors
- OSS distribution channels



From a Research Project on Scalable Management of VMs:



# The Benefits of OpenNebula

---

## For the Infrastructure Manager

- Centralized management of VM workload and distributed infrastructures
- Support for VM placement policies: balance of workload, server consolidation...
- Dynamic resizing of the infrastructure
- Dynamic partition and isolation of clusters
- Dynamic scaling of private infrastructure to meet fluctuating demands
- Lower infrastructure expenses combining local and remote Cloud resources

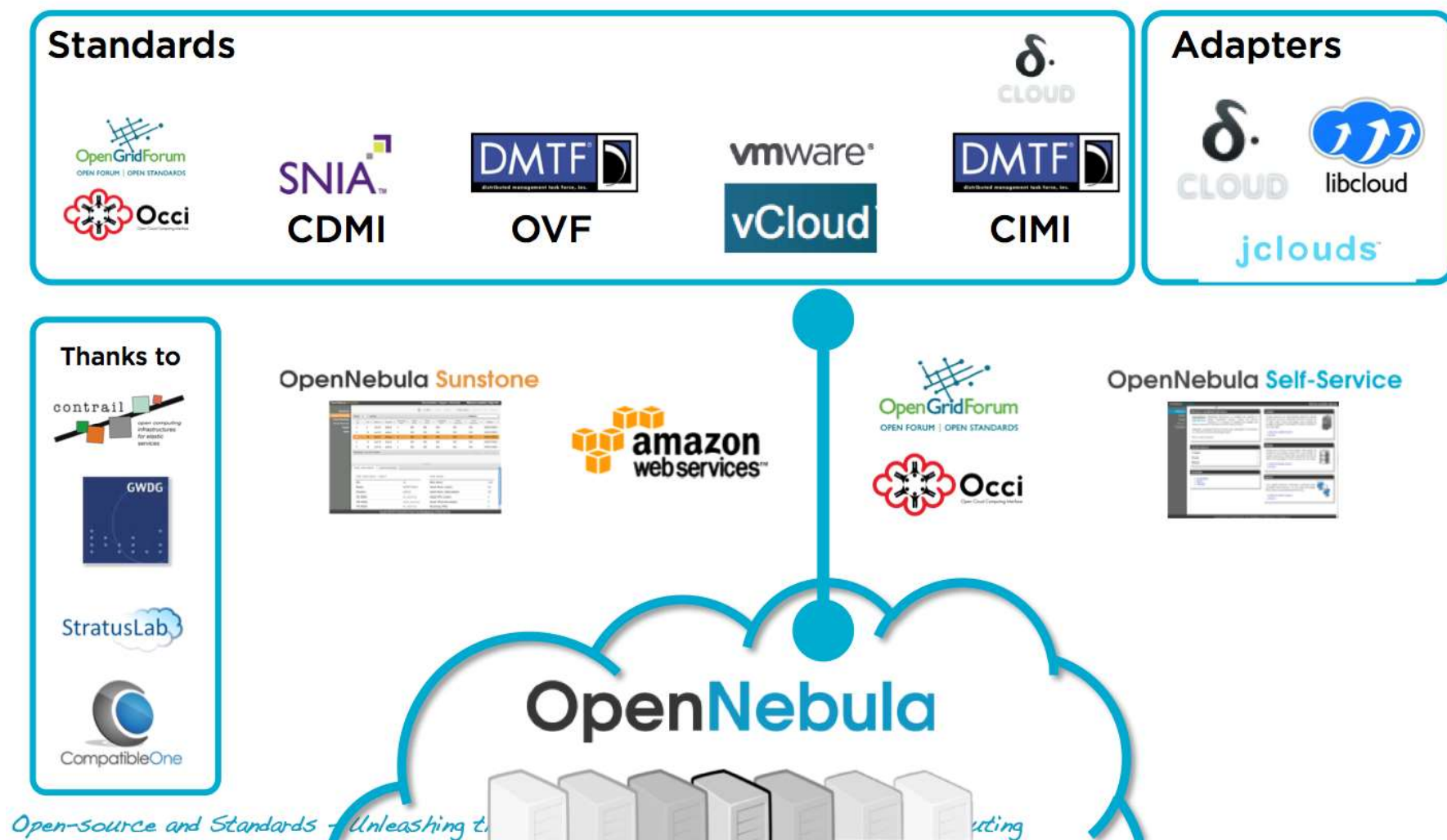
## For the Infrastructure User

- Faster delivery and scalability of services
- Support for heterogeneous execution environments
- Full control of the lifecycle of virtualized services management



# Interoperability From the Cloud Consumer Perspective

Standards (de facto and de jure) and adapters can be used to leverage existing ecosystems and ensure portability across providers....



# Interoperability from the Cloud Provider perspective

Interoperable (platform independent), innovative (feature-rich) and proven (mature to run in production).



# The Benefits for System Integrators

---

- Fits into any existing data center, due to its open, flexible and extensible interfaces, architecture and components
- Builds any type of Cloud deployment
- Open source software, Apache license
- Seamless integration with any product and service in the cloud ecosystem and management tool in the data center, such as
  - cloud providers
  - VM managers
  - virtual image managers
  - service managers
  - management tools
  - schedulers

# The main features of OpenNebula

Feature	Function
<b>Internal Interface</b>	<ul style="list-style-type: none"><li>• Unix-like CLI for fully management of VM life-cycle and physical boxes</li><li>• XML-RPC API and libvirt virtualization API</li></ul>
<b>Scheduler</b>	<ul style="list-style-type: none"><li>• Requirement/rank matchmaker allowing the definition of workload and resource-aware allocation policies</li><li>• Support for advance reservation of capacity through Haizea</li></ul>
<b>Virtualization Management</b>	<ul style="list-style-type: none"><li>• Xen, KVM, and VMware</li><li>• Generic libvirt connector (VirtualBox)</li></ul>
<b>Image Management</b>	<ul style="list-style-type: none"><li>• General mechanisms to transfer and clone VM images</li></ul>
<b>Network Management</b>	<ul style="list-style-type: none"><li>• Definition of isolated virtual networks to interconnect VMs</li></ul>
<b>Service Management and Contextualization</b>	<ul style="list-style-type: none"><li>• Support for multi-tier services consisting of groups of inter-connected VMs, and their auto-configuration at boot time</li></ul>
<b>Security</b>	<ul style="list-style-type: none"><li>• Management of users by the infrastructure administrator</li></ul>
<b>Fault Tolerance</b>	<ul style="list-style-type: none"><li>• Persistent database backend to store host and VM information</li></ul>
<b>Scalability</b>	<ul style="list-style-type: none"><li>• Tested in the management of medium scale infrastructures with hundreds of servers and VMs (no scalability issues has been reported)</li></ul>
<b>Flexibility and Extensibility</b>	<ul style="list-style-type: none"><li>• Open, flexible and extensible architecture, interfaces and components, allowing its integration with any product or tool</li></ul>

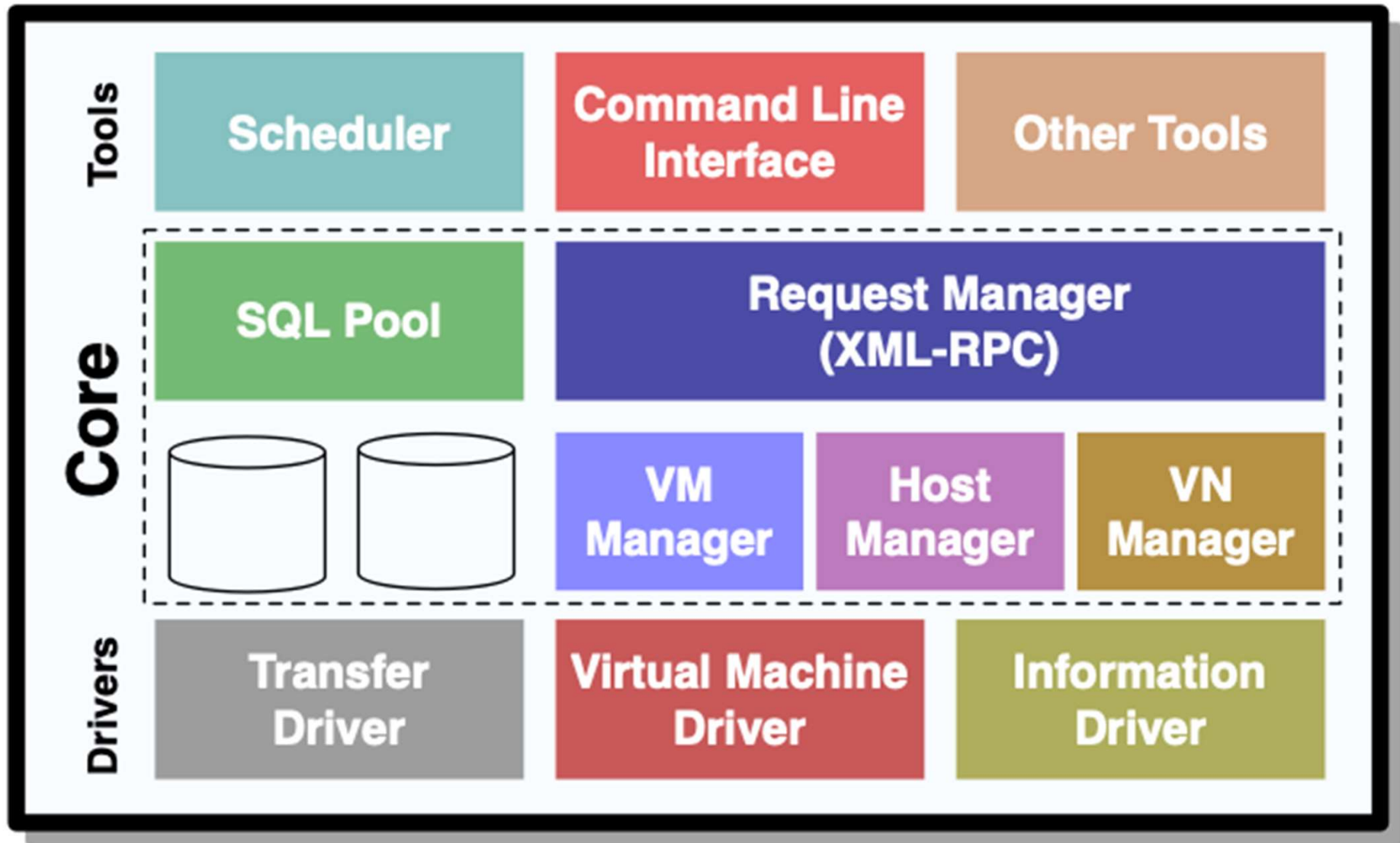
# Comparison with Similar Technologies

	Platform ISF	VMware Vsphere	Eucalyptus	Nimbus	OpenNebula
Virtualization Management	VMware, Xen	VMware	Xen, KVM	Xen	Xen, KVM, VMware
Virtual Network Management	Yes	Yes	No	Yes	Yes
Image Management	Yes	Yes	Yes	Yes	Yes
Service Contextualization	No	No	No	Yes	Yes
Scheduling	Yes	Yes	No	No	Yes
Administration Interface	Yes	Yes	No	No	Yes
Hybrid Cloud Computing	No	No	No	No	Yes
Cloud Interfaces	No	vCloud	EC2	WSRF, EC2	EC2 Query, OGF OCCI
Flexibility and Extensibility	Yes	No	Yes	Yes	Yes
Open Source	No	No	GPL	Apache	Apache



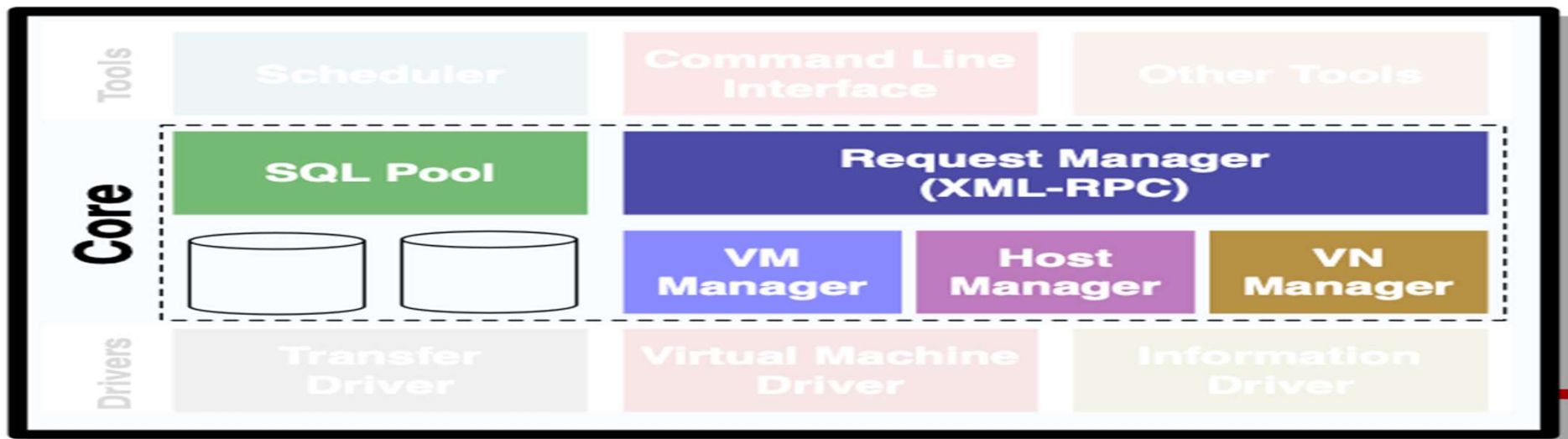
# Inside OpenNebula

# OpenNebula Architecture



# The Core

- Request manager: Provides a XML-RPC interface to manage and get information about ONE entities.
- SQL Pool: Database that holds the state of ONE entities.
- VM Manager (virtual machine): Takes care of the VM life cycle.
- Host Manager: Holds handling information about hosts.
- VN Manager (virtual network): This component is in charge of generating MAC and IP addresses.





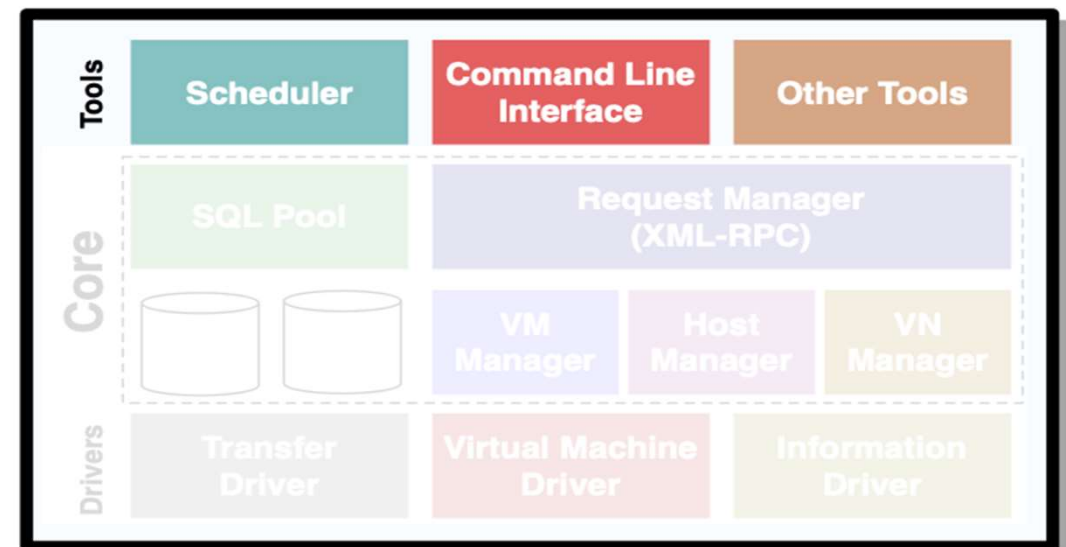
# The tools layer

## Scheduler:

- Searches for physical hosts to deploy newly defined VMs

## Command Line Interface:

- Commands to manage OpenNebula.
- onevm: Virtual Machines
  - create, list, migrate...
- onehost: Hosts
  - create, list, disable...
- onevnet: Virtual Networks
  - create, list, delete...



# The drivers layer

Transfer Driver: Takes care of the images.

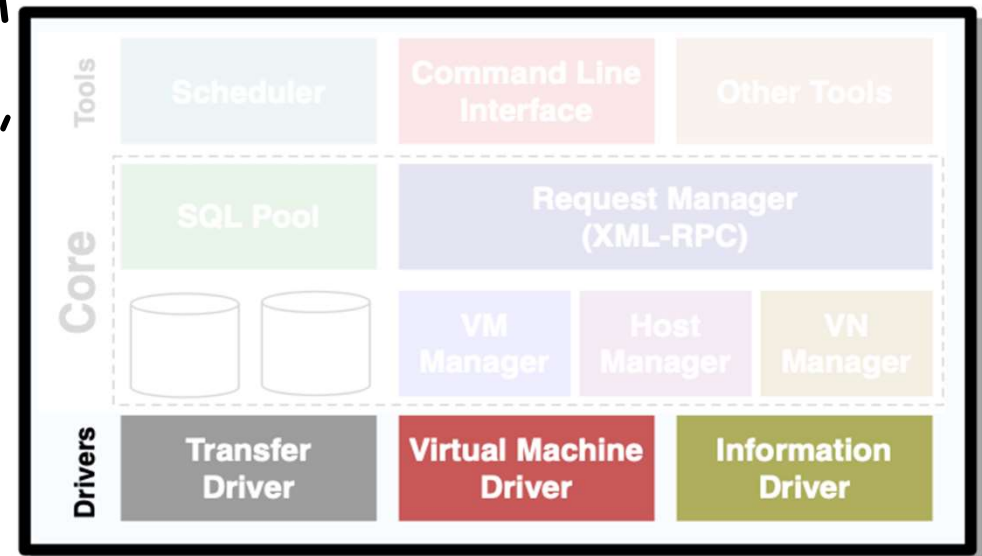
- cloning, deleting, creating swap image...

Virtual Machine Driver: Manager of the lifecycle of a virtual machine

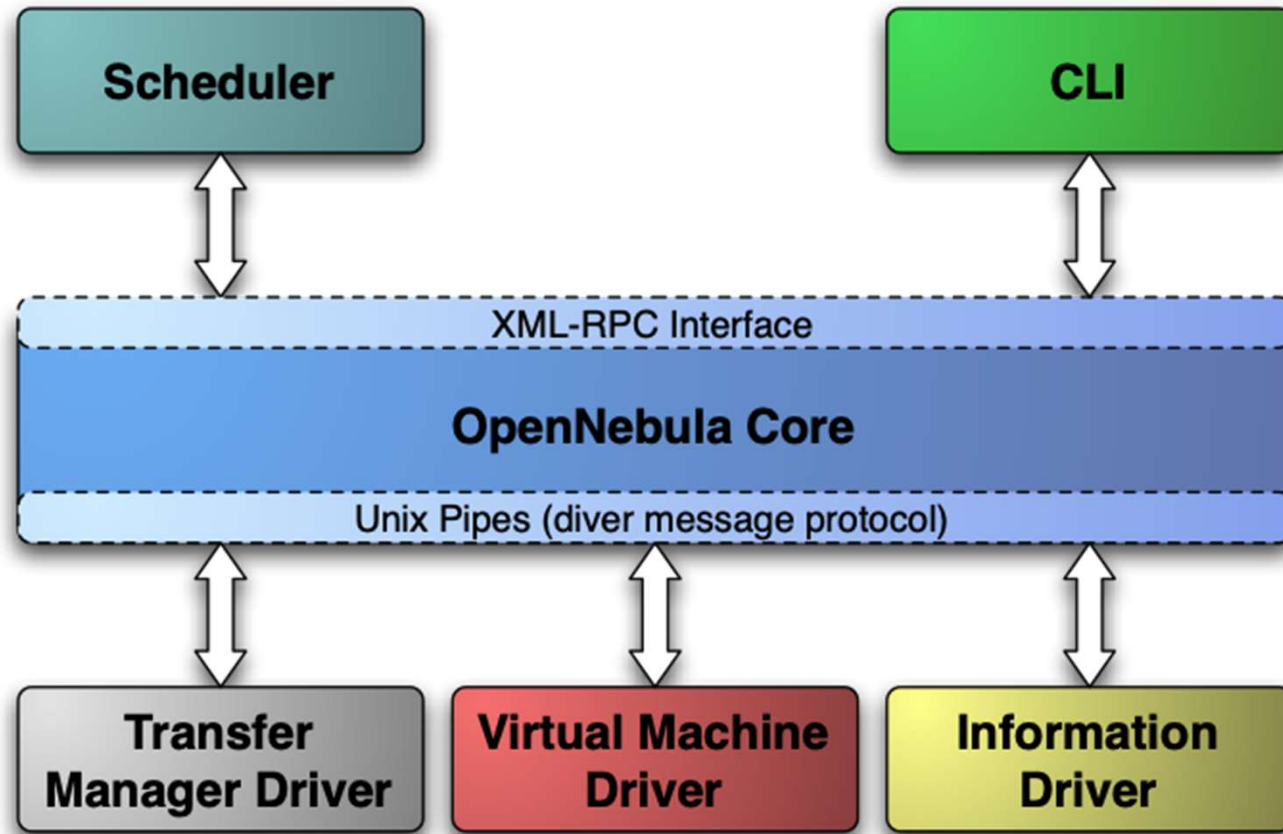
- deploy, shutdown, poll, migrate...

Information Driver: Executes scripts in physical hosts to gather information about them

- total memory, free memory, total #cpus, cpu consumed...



# Process separation

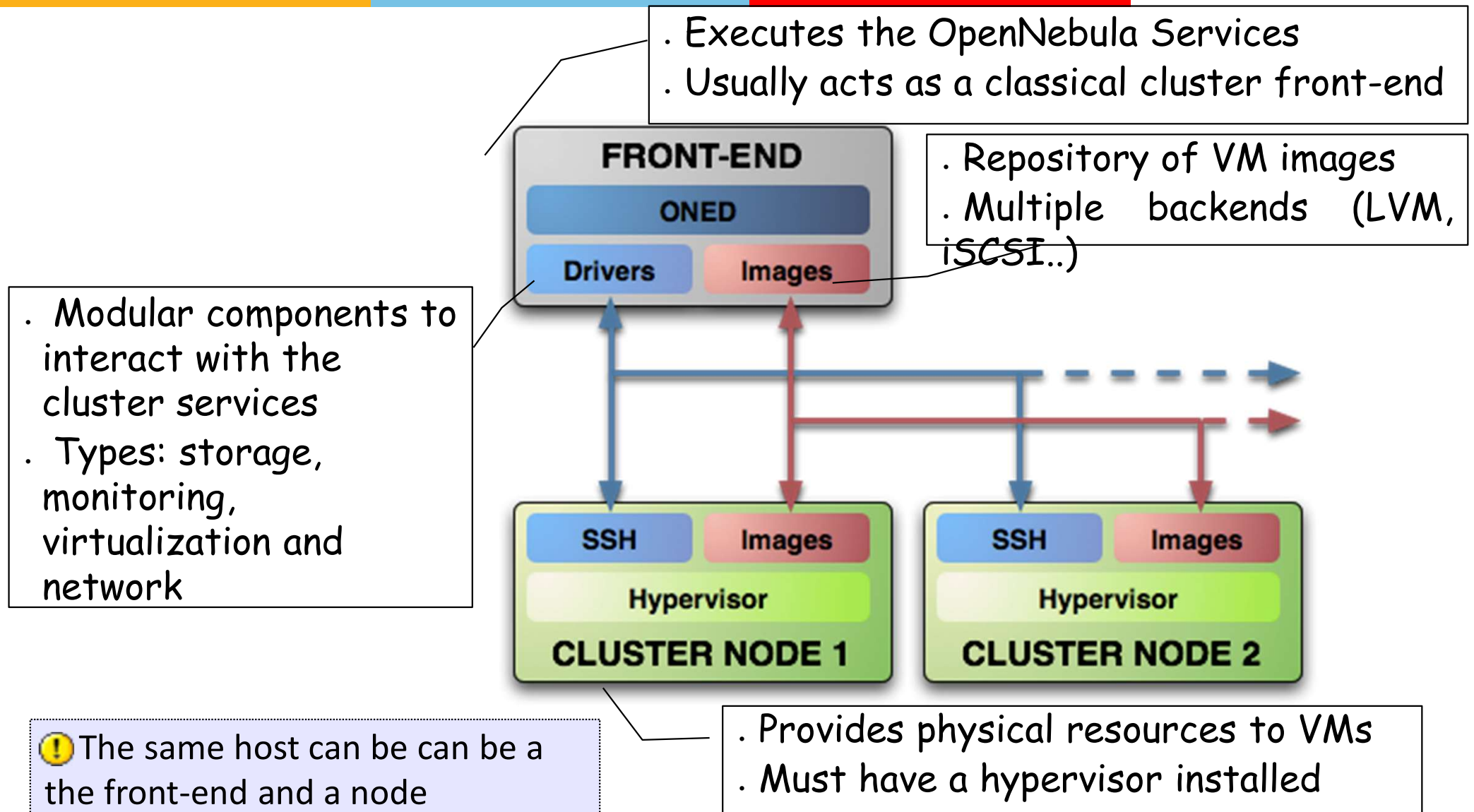


- Scheduler is a separated process, just like command line interface.
- Drivers are also separated processes using a simple text messaging protocol to communicate with OpenNebula Core Daemon (oned)

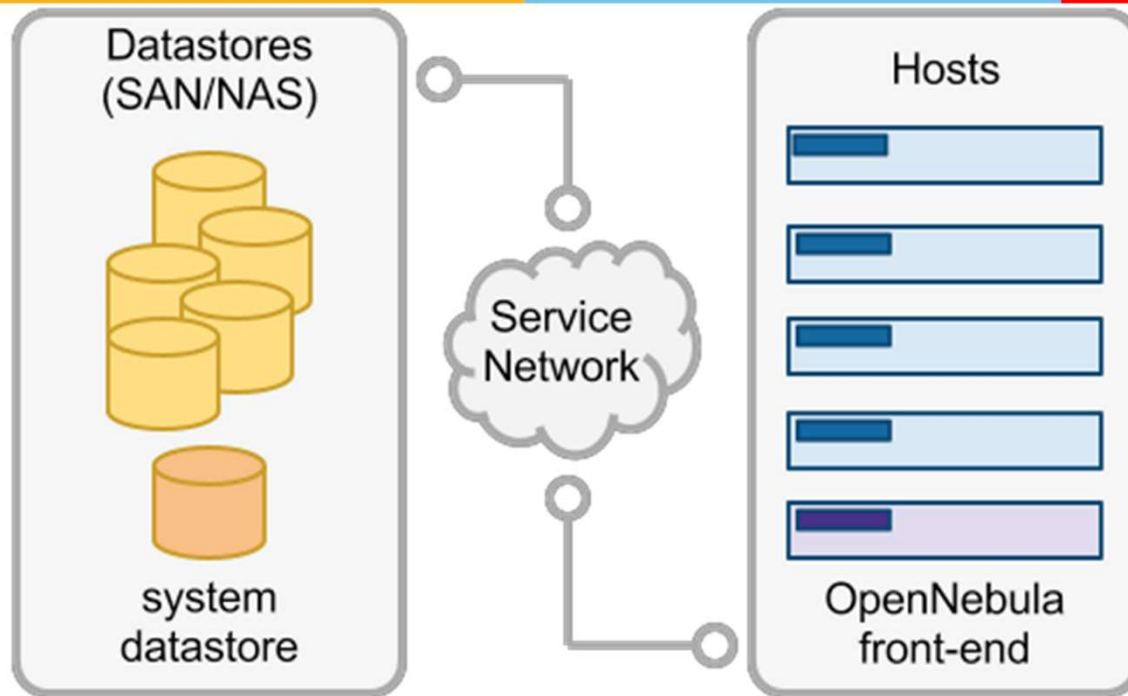


# Constructing a private cloud

# System Overview



## Complex Storage behind OpenNebula

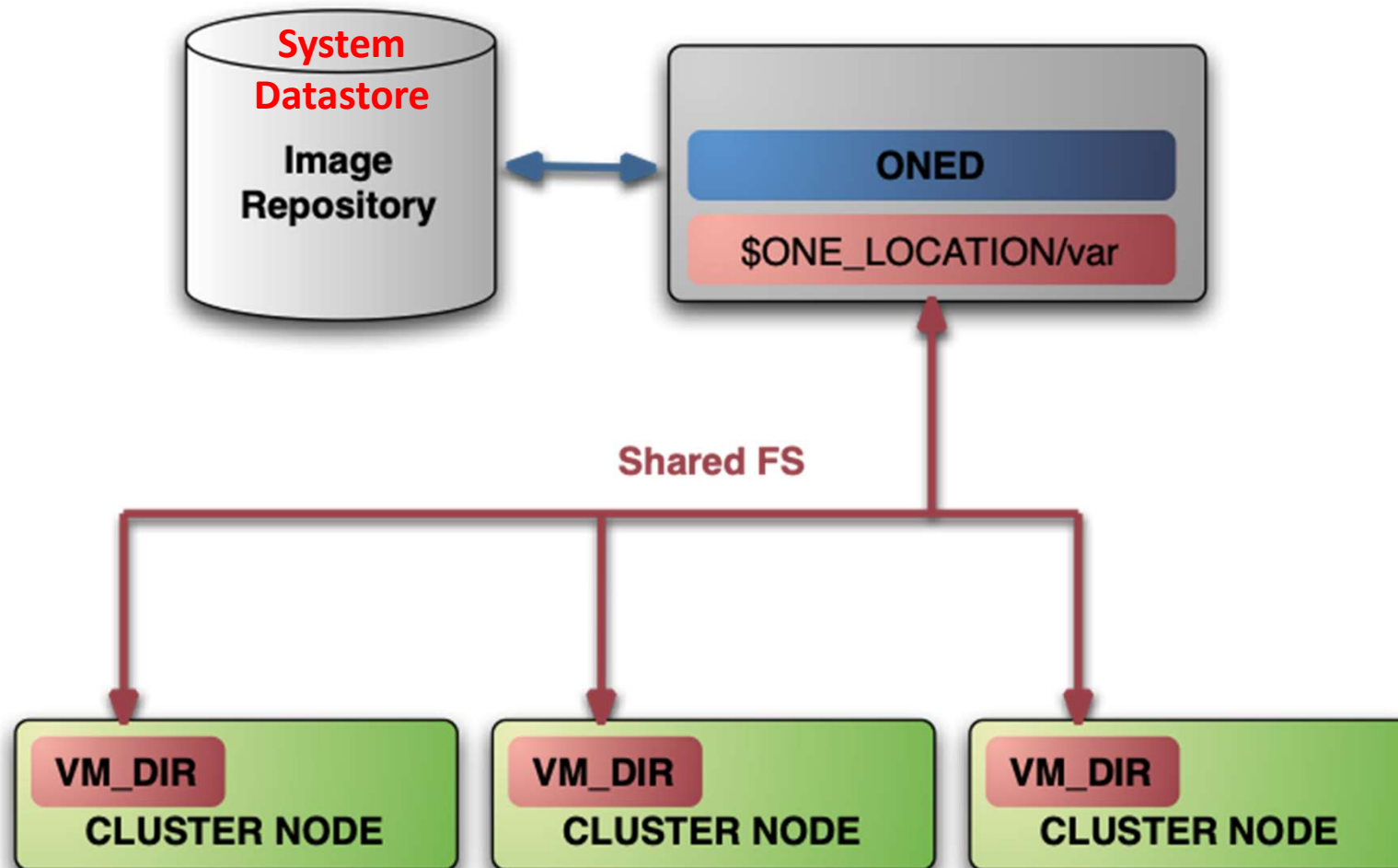


Datastore	Transfer Manager Drivers				
	shared	ssh	iscsi	qcow	vmware
System	OK	OK			
File-System	OK	OK		OK	
iSCSI			OK		
VMware	OK	OK			OK

Virtual machines and their images are represented as files

Virtual machines and their images are represented as block devices (just like a disk)

## System Datastore with Shared Transfer Manager Driver



# Preparing the storage for a simple private cloud

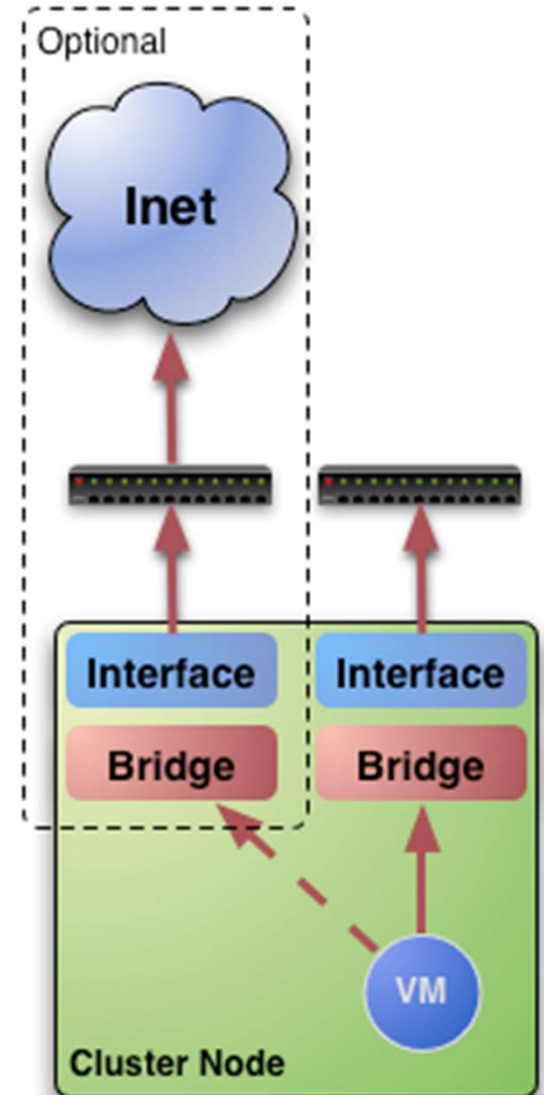
- **Image Repository (system datastore):** Any storage medium for the VM images (usually a high performing SAN)
  - OpenNebula supports multiple back-ends (e.g. LVM for fast cloning)
  - The front-end must have access to the repository
- **VM Directory:** The home of the VM in the cluster node
  - Stores checkpoints, description files and VM disks
  - Actual operations over the VM directory depends on the storage medium
  - Should be shared for live-migrations
  - You can go on without a shared FS and use the SSH back-end
  - Defaults to `$ONE_LOCATION/var/$VM_ID`

⚠ **Dimensioning the Storage...** Example: A 64 core cluster will typically run around 80VMs, each VM will require an average of 10GB of disk space. So you will need ~800GB for `/srv/cloud/one`, you will also want to store 10-15 master images so ~200GB for `/srv/cloud/images`. A 1TB `/srv/cloud` will be enough for this example setup.

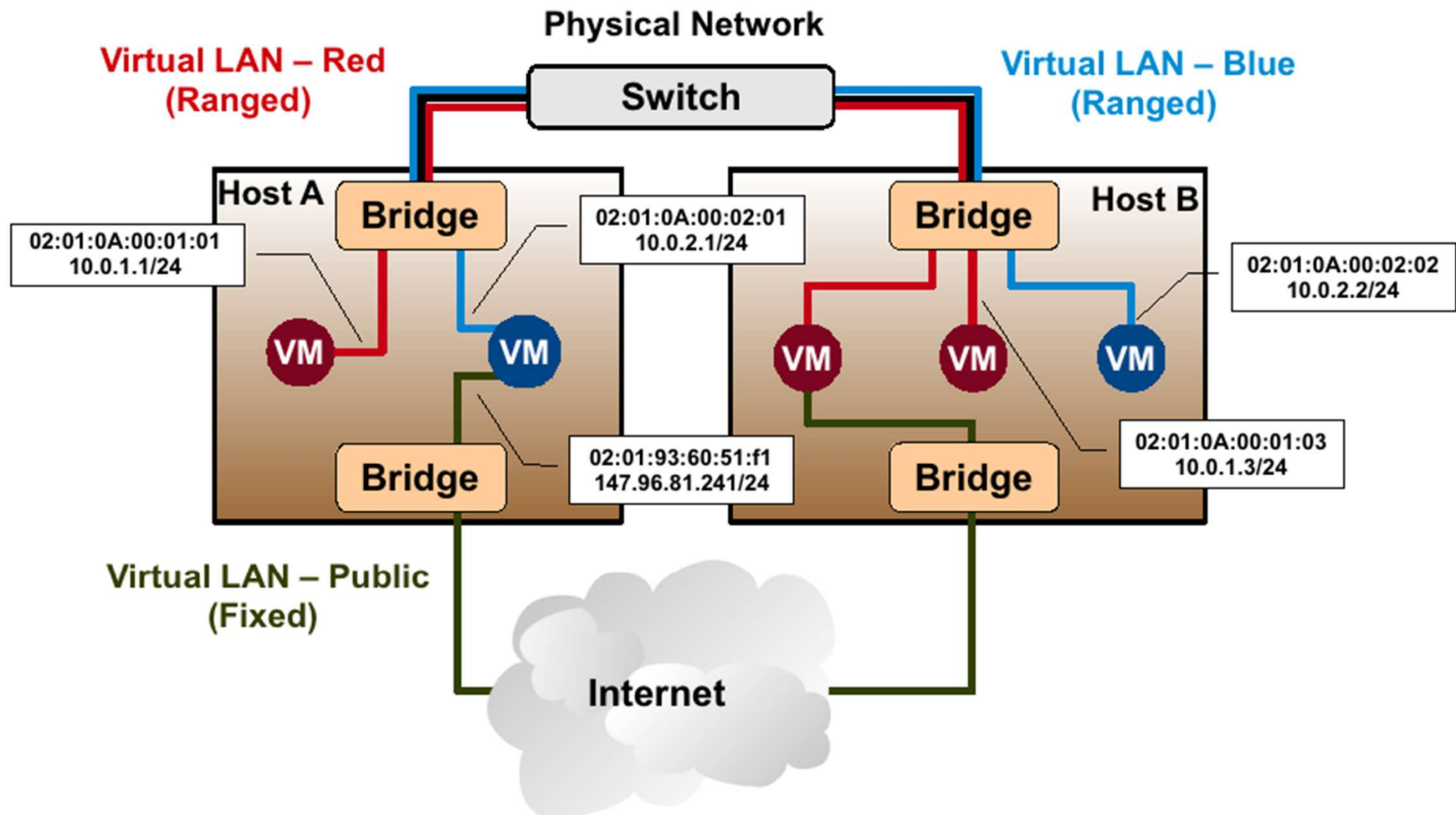


## Networking for private clouds

- OpenNebula management operations use ssh connections
- **Image traffic**, may require the movement of heavy files (VM images, checkpoints). Dedicated storage links may be a good idea
- **VM demands**, consider the typical requirements of your VMs. Several NICs to support the VM traffic may be a good idea
- OpenNebula relies on bridge networking for the VMs



# Example network setup in a private cloud



# Virtual Networks

---

- A Virtual Network in OpenNebula
  - Defines a separated MAC/IP address space to be used by VMs
  - Each virtual network is associated with a physical network through a bridge
  - Virtual Networks can be isolated (at layer 2 level) with ebtables and hooks
- Virtual Networks are managed with the `onevnet` utility

# Users

---

A User in OpenNebula

- Is a pair of username:password

Only oneadmin can add/delete users

Users are managed with the oneuser utility

# User Management

---

Native user support since v1.4

- oneadmin: privileged account

Usage, management, administrative rights for:

- Templates, VMs, Images, Virtual Networks

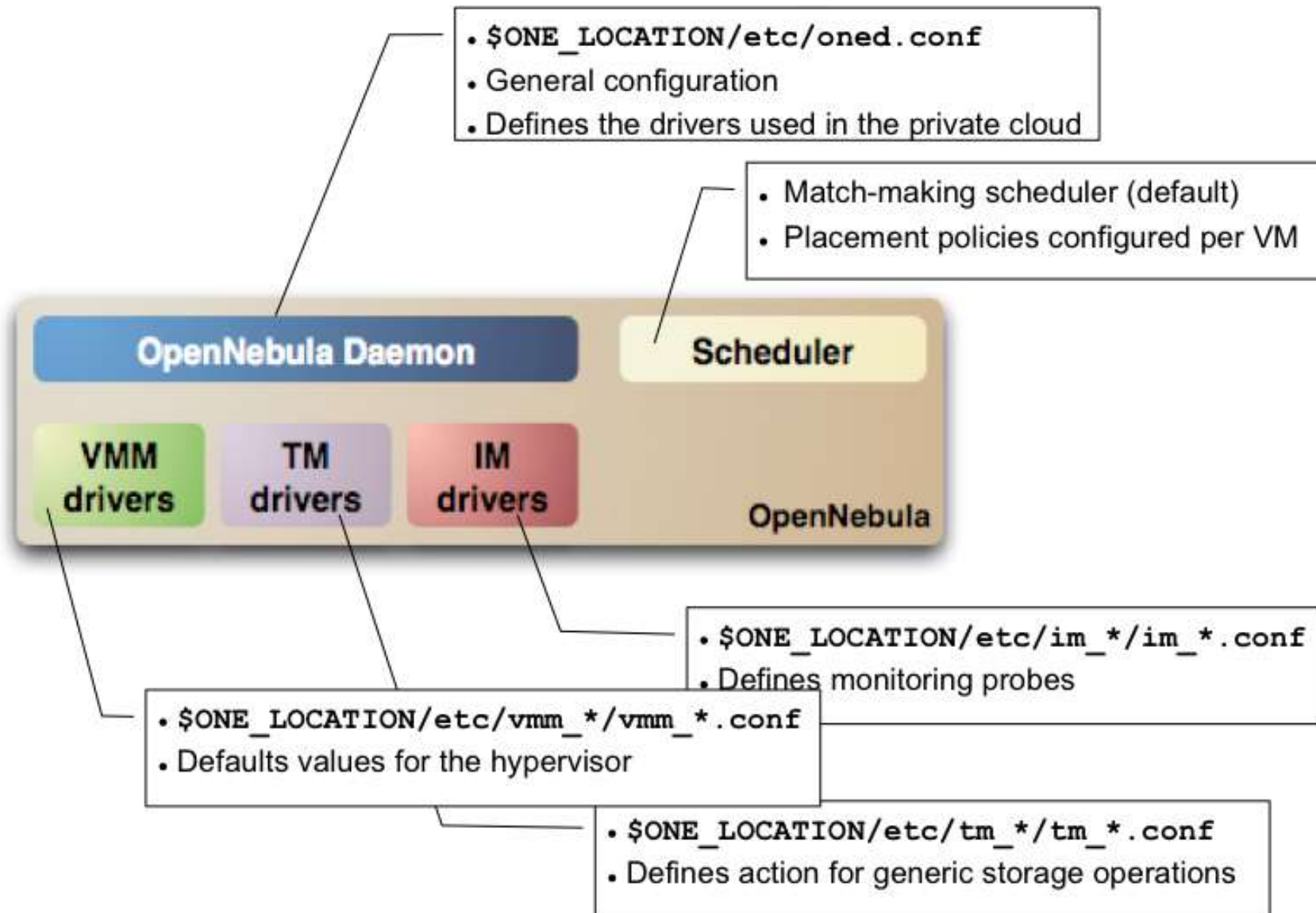
Through ACLs further operations/rights are available:

- Rights for users, groups, datastores and clusters
- Creation operation

SHA1 passwords (+AA module)

- Stored in FS
- Alternatively in environment

# Configuration





# Virtual machines

# Preparing VMs for OpenNebula

---

Virtual Machines are managed with the oneuser utility

You can use any VM prepared for the target hypervisor

**Hint I:** Place the `vmcontext.sh` script in the boot process to make better use of vlans

**Hint II:** Do not pack useless information in the VM images:

- swap. OpenNebula can create swap partitions on-the-fly in the target host
- Scratch or volatile storage. OpenNebula can create plain FS on-the-fly in the target host

**Hint III:** Install once and deploy many; prepare master images

**Hint IV:** Do not put private information (e.g. ssh keys) in the master images, use the `CONTEXT`

**Hint V:** Pass arbitrary data to a master image using `CONTEXT`



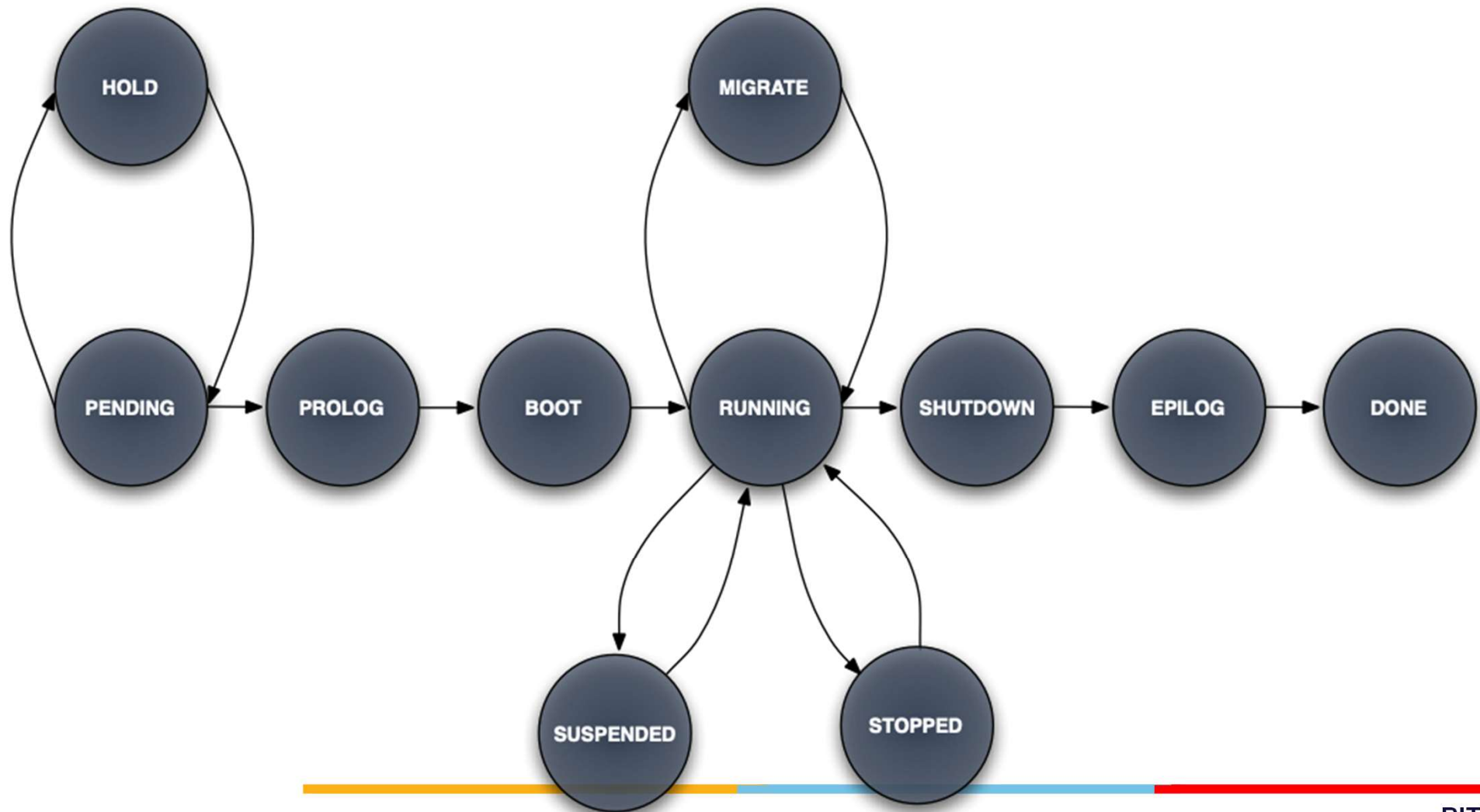
# VM Description

Option	Description
<b>NAME</b>	<ul style="list-style-type: none"><li>• Name that the VM will get for description purposes.</li></ul>
<b>CPU</b>	<ul style="list-style-type: none"><li>• Percentage of CPU divided by 100 required for the Virtual Machine.</li></ul>
<b>OS (KERNEL, INITRD)</b>	<ul style="list-style-type: none"><li>• Path of the kernel and initrd files to boot from.</li></ul>
<b>DISK (SOURCE, TARGET, CLONE, TYPE)</b>	<ul style="list-style-type: none"><li>• Description of a disk image to attach to the VM.</li></ul>
<b>NIC (NETWORK)</b>	<ul style="list-style-type: none"><li>• Definition of a virtual network the VM will be attached to.</li></ul>

Multiple disk and network interfaces can be specified just adding more disk/nic statements.

To create swap images you can specify `TYPE=swap`, `SIZE=<size in MB>`.  
By default disk images are cloned, if you do not want that to happen `CLONE=no` can be specified and the VM will attach the original image.

## VM States overview



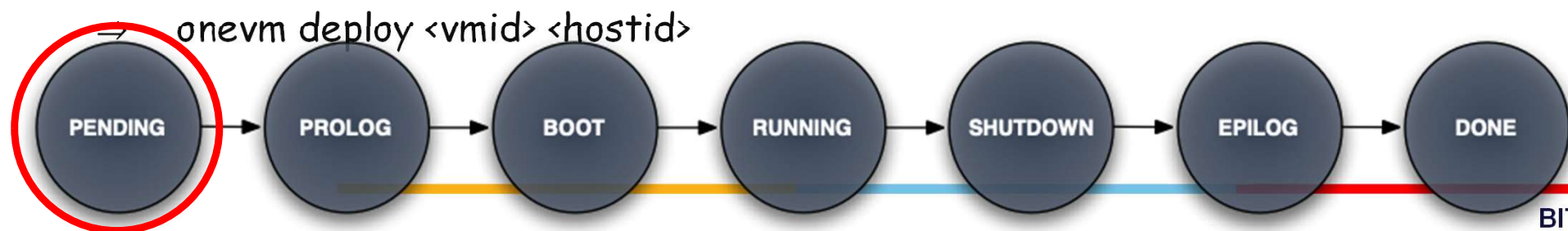
# Pending state

After submitting a VM description to ONE it is added to the database and its state is set to PENDING.

In this state IP and MAC addresses are also chosen if they are not explicitly defined.

The scheduler awakes every 30 seconds and looks for VM descriptions in PENDING state and searches for a physical node that meets its requirements. Then a deploy XML-RPC message is sent to oned to make it run in the selected node.

Deployment can be also made manually using the Command Line Interface:

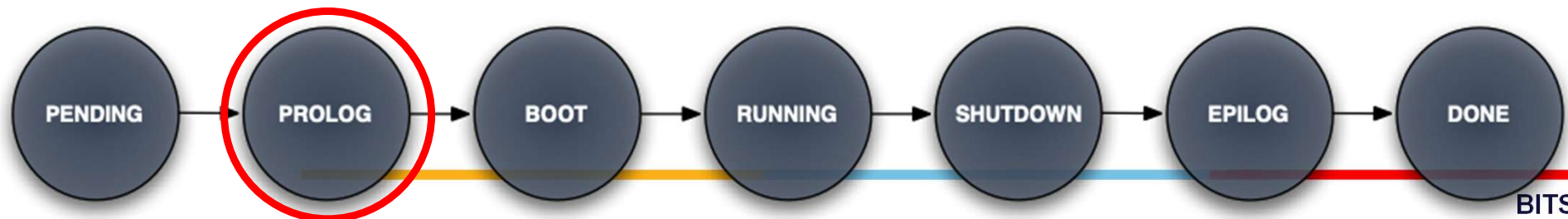


# Prolog state

In PROLOG state the Transfer Driver prepares the images to be used by the VM.

Transfer actions:

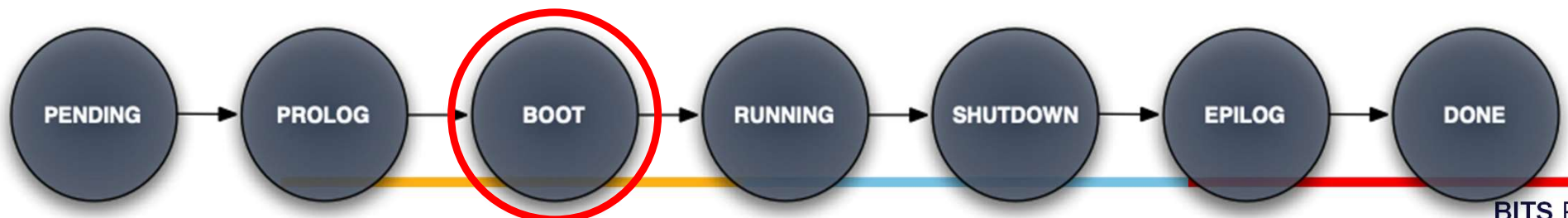
- **CLONE**: Makes a copy of a disk image file to be used by the VM. If Clone option for that file is set to false and the Transfer Driver is configured for NFS then a symbolic link is created.
- **MKSWAP**: Creates a swap disk image on the fly to be used by the VM if it is specified in the VM description.



# Boot state

In this state a deployment file specific for the virtualization technology configured for the physical host is generated using the information provided in the VM description file. Then Virtual Machine Driver sends deploy command to the virtual host to start the VM.

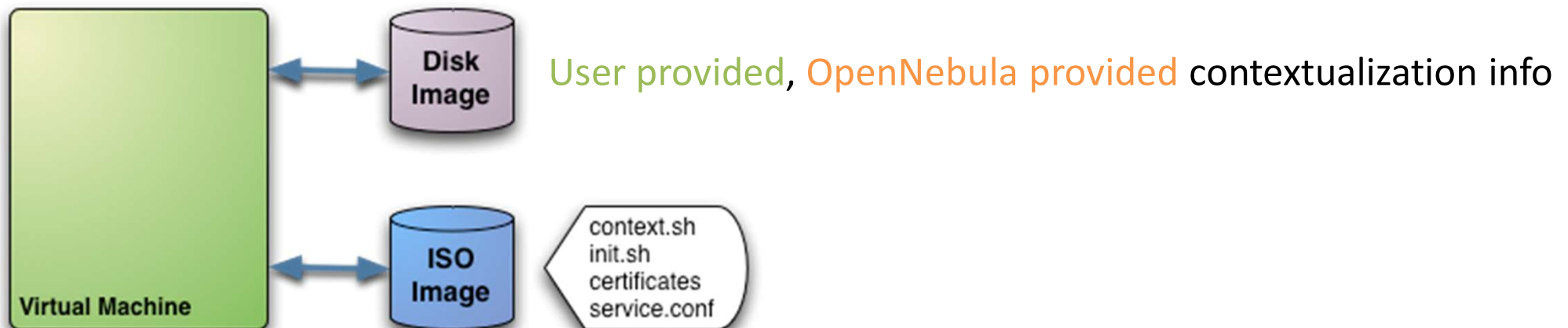
The VM will be in this state until deployment finishes or fails.



# Contextualization

The ISO image has the contextualization for that VM:

- **context.sh**: contains configuration variables
- **init.sh**: script called by VM at start to configure specific services
- **certificates**: directory that contains certificates for some service
- **service.conf**: service configuration



# Running and Shutdown states

While the VM is in **RUNNING** state it will be periodically polled to get its consumption and state.

In **SHUTDOWN** state Virtual Machine Driver will send the shutdown command to the underlying virtual infrastructure.

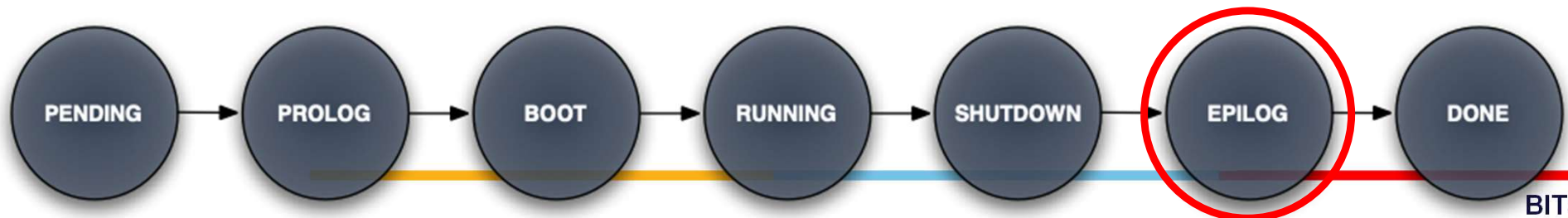





# Epilog state

In EPILOG state the Transfer Manager Driver is called again to perform this actions:

- Copy back the images that have **SAVE**=yes option.
- Delete images that were cloned or generated by **MKSWAP**.



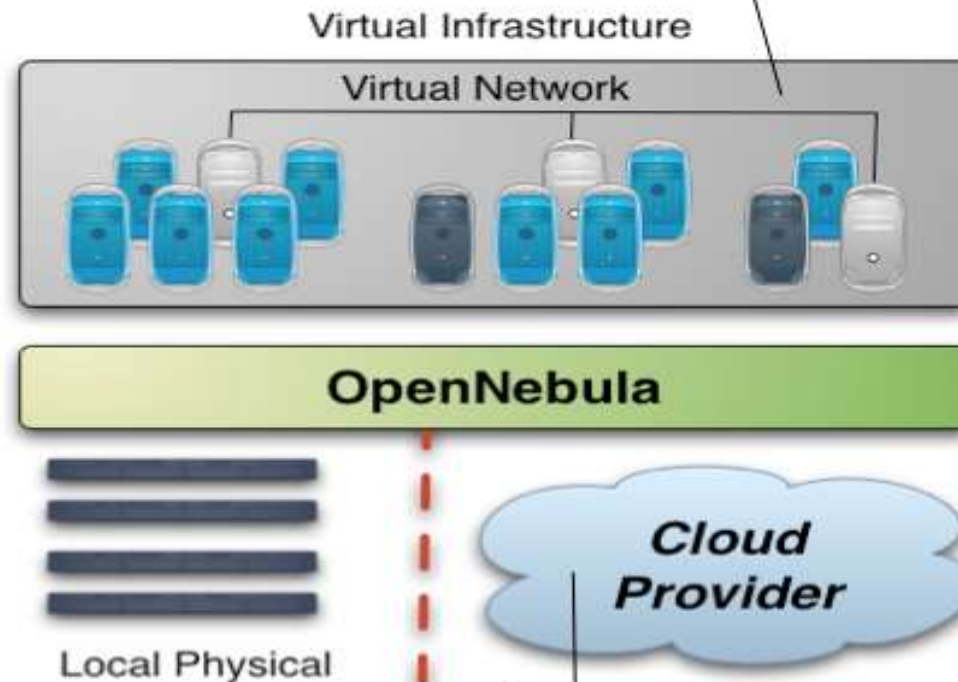




# Hybrid cloud

# Overview

- VMs can be local or remote
- VM connectivity has to be configured, usually VPNs



- External Clouds are like any other host
- Placement constraints

# Making an Amazon EC2 hybrid

---

Amazon EC2 cloud is managed by OpenNebula as any other cluster node

- You can use several accounts by adding a driver for each account (use the arguments attribute, -k and -c options). Then create a host that uses the driver
- You can use multiple EC2 zones, add a driver for each zone (use the arguments attribute, -u option), and a host that uses that driver
- You can limit the use of EC2 instances by modifying the IM file

# Using an EC2 hybrid cloud

---

Virtual Machines can be instantiated locally or in EC2  
The VM template must provide a description for both instantiation methods.

The EC2 counterpart of your VM (AMI\_ID) must be available for the driver account

The EC2 VM template attribute should describe not only the VM's properties but the contact details of the external cloud provider

# Hybrid cloud Use Case

## On-demand Scaling of Computing Clusters

- On-demand Scaling of Web Servers
- Elastic execution of the NGinx web server
- The capacity of the elastic web application can be dynamically increased or decreased by adding or removing NGinx instances

