Session 3
Administration and Basic Usage – Part II

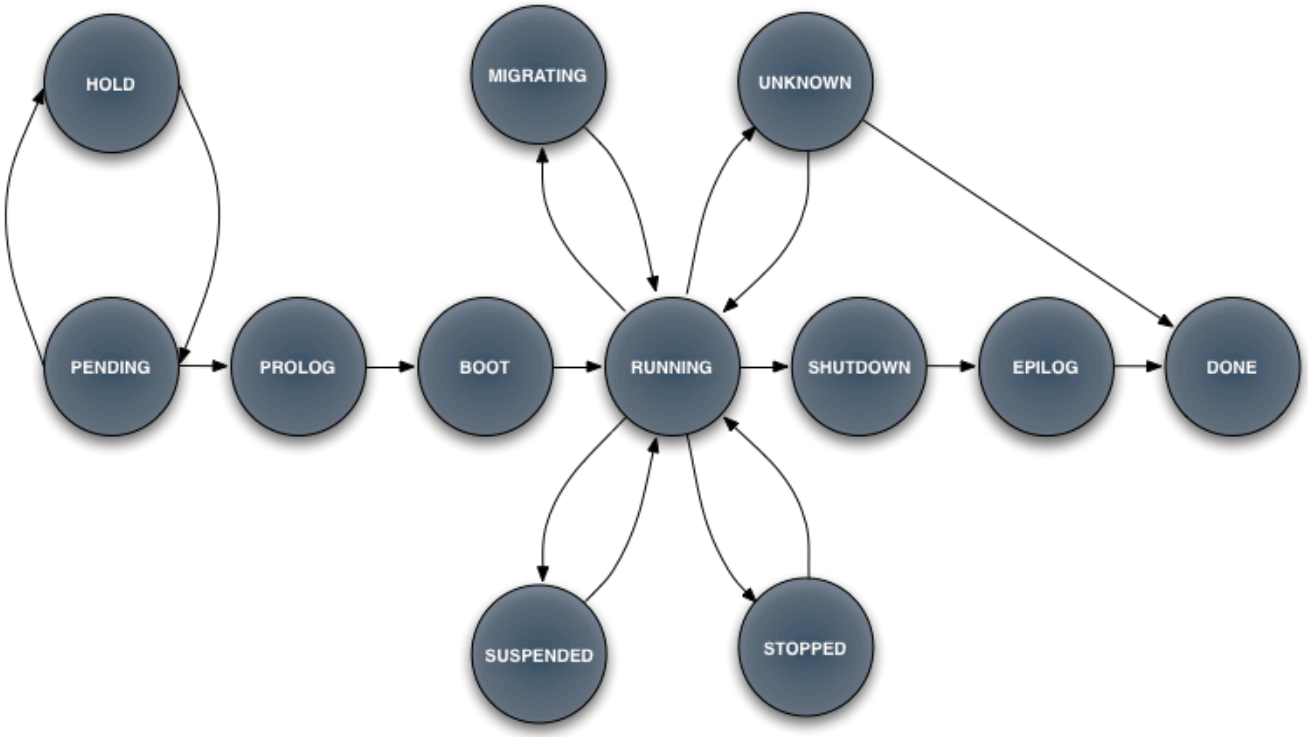
Constantino Vázquez
tinova@fdi.ucm.es

Using the Private Cloud: Virtual Machines

- Preparing a VM to be used with OpenNebula
  - 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`

Using the Private Cloud: Virtual Machines

- Virtual Machine Life-cycle



Using the Private Cloud: Virtual Machines

- A Virtual Machine in OpenNebula
  - A **capacity** in terms memory and CPU
  - A set of **NICs** attached to one or more virtual networks
  - A set of **disk images**, to be "*transferred*" to/from the execution host.
  - A **state file** (optional) or recovery file, with the memory image of a running VM plus some hypervisor specific information.
- Virutal Machines are defined in a VM template
- Each VM has an unique ID in OpenNebula → the VM_ID
- All the files (logs, images, state files...) are stored in `$ONE_LOCATION/var/<VM_ID>`

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#-----------------------------------
# Name of the VM
#-----------------------------------
NAME = "vm-example" # Optional, Default: one-$VMID

#-----------------------------------
#               Capacity
#-----------------------------------
CPU    = "amount_of_requested_CPU"
MEMORY = "amount_of_requested_MEM"
VCPU   = "number of virtual cpus"

#-----------------------------------
#         OS and boot options
#-----------------------------------
OS = [
  kernel     = "path_to_os_kernel",     # para-virtualization
  initrd     = "path_to_initrd_image",  # para-virtualization
  kernel_cmd = "kernel_command_line",
  root       = "device to be mounted as root"
  bootloader = "path to the boot loader exec"
  boot       = "device to boot from" ]
```


Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#-----------------------------------
#      Features of the hypervisor
#-----------------------------------

FEATURES = [
  pae  = "yes|no",   # Optional, KVM
  acpi = "yes|no" ]  # Optional, KVM

#-----------------------------------
#                VM Disks
#-----------------------------------

DISK = [
  type     = "floppy|disk|cdrom|swap|fs|block",
  source   = "path_to_disk_image_file|physical_dev",
  format   = "type for fs disks",
  size     = "size_in_GB",
  target   = "device_to_map_disk",
  bus      = "ide|scsi|virtio|xen",
  readonly = "yes|no",
  clone    = "yes|no",
  save     = "yes|no" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#           Network Interfaces
#---------------------------------------

NIC = [
  network = "name_of_the_virtual_network",
  ip      = "ip_address",
  bridge  = "name_of_bridge_to_bind_if",
  target  = "device_name_to_map_if",
  mac     = "HW_address",
  script  = "path_to_script_to_bring_up_if",
  Model   = "NIC model"]

#---------------------------------------
#  I/O Interfaces
#---------------------------------------

INPUT = [
  type = "mouse|tablet",
  bus  = "usb|ps2|xen" ]
```


Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#  I/O Interfaces
#---------------------------------------

GRAPHICS = [
  type   = "vnc|sdl",
  listen = "IP-to-listen-on",
  port   = "port_for_VNC_server",
  passwd = "password_for_VNC_server" ]

#---------------------------------------
#  Raw Hypervisor attributes
#---------------------------------------

RAW = [
  type = "xen|kvm",
  data = "raw_domain_configutarion"]
```

Not all the parameters are supported for each hypervisor. Complete reference and examples for all sections in
http://www.opennebula.org/doku.php?id=documentation:rel1.4:template

Using the Private Cloud: Virtual Machines

- Hands on... define a ttylinux VM

```
NAME   = ttylinux
CPU    = 0.1
MEMORY = 64

DISK   = [
  source   = "/srv/cloud/images/ttylinux/ttylinux.img",
  target   = "hda",
  readonly = "no" ]

NIC      = [ NETWORK = "One-TD" ]

FEATURES = [ acpi="no" ]

#This may be useful to debug your VMs (can use also console)

GRAPHICS = [
  type = "vnc",
  listen  = "loclahost",
  port = "5902",
  keymap="es"]
```


Using the Private Cloud: Virtual Machines

- Hands on
  - Copy the one ttylinux image form the front-end

```
$ scp gw:ttylinux-xen.tar.gz .
$ tar xvzf ttylinux-xen.tar.gz
```

- Virtual Machines are managed with the onevm utility
  - Operations: create, deploy shutdown, livemigrate, stop, cancel, resume, suspend, delete, restart
  - Information: list, show, top, history

```
$ onevm create ttylinux.one

$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
  1  oneadmin ttylinux pend   0       0                  00 00:00:28

$ onevm top
```

- Hands on...
  - Create a basic VMs
  - Create a couple of network enabled VMs
    - Check virtual network usage (onevnet)
  - Try control operations with the VMs
    - stop, shutdown, resume...
    - migrate – check xm list
  - Modify the template
    - Add one more NIC for the One-Td-Invisible network
    - Add another `DISK` for VM data (`type="fs", format="ext2"`)