

Goal of the Project:

The goal of the project is to implement Import and Export Instances(Virtual Machines) functionality for KVM Hypervisor. It allows cloudstack operators to manage or unmanage VMs from their cloud environment. Currently, this functionality is only available for VMWare and I would be working on extending it to KVM Hypervisor. Manage Instance are the ones which are managed by apache cloudstack and persist in cloudstack databases. Unmanage Instances are the ones which are not managed by apache cloudstack and do not persist in cloudstack Database.By managing through cloudstack, the users get all the benefits that cloudstack as an orchestrator provides.

Contributions made during GSoC :

Extension of the Implementation of three API's:

- ListUnmanagesInstances
- unManageVirtualMachine
- importUnmanagedInstances

listUnmanagedInstances API: This API is used to list all the unmanage instances in apache cloudstack. The purpose of this api is to fetch and list out all the virtual machines that may be running on a hypervisor but are not managed by apache cloudstack. The api also lists the meta data for the virtual machines. Previously it was implemented for VMWare and now as part of this work we extended it for KVM Hypervisor as well. In terms of implementation KVM and VMware are significantly different hypervisors with KVM being open source. Consequently the implementation to support VMWare and KVM are maintained in two separate plugins within apache cloudstack project.

For extension of this api for KVM ,this api executes the required qemu/virsh commands and presents the users with VM details. It will list all the unmanage instances in the apache cloudstack (meaning these instances are not managed by cloudstack and do not persist in cloudstack database). To extend listUnmanagedInstances api for KVM or similar, I created a wrapper class LibvirtPrepareUnmanageVMInstanceCommandWrapper which wraps the core commands (PrepareUnmanageVMInstanceCommand) which is send from management server to the host/agent. This wrapper class is executed on the host which returns an object of answer class(PrepareUnmanageVMInstanceAnswer) which has the list of all the unmanaged virtual machines which is sent back to the management server and then management server sends it back to the user. This class is used to check if the virtual machine which needs to be unmanaged exists or not.

unmanageVirtualMachine API: It is used to unmanage a managed instance.

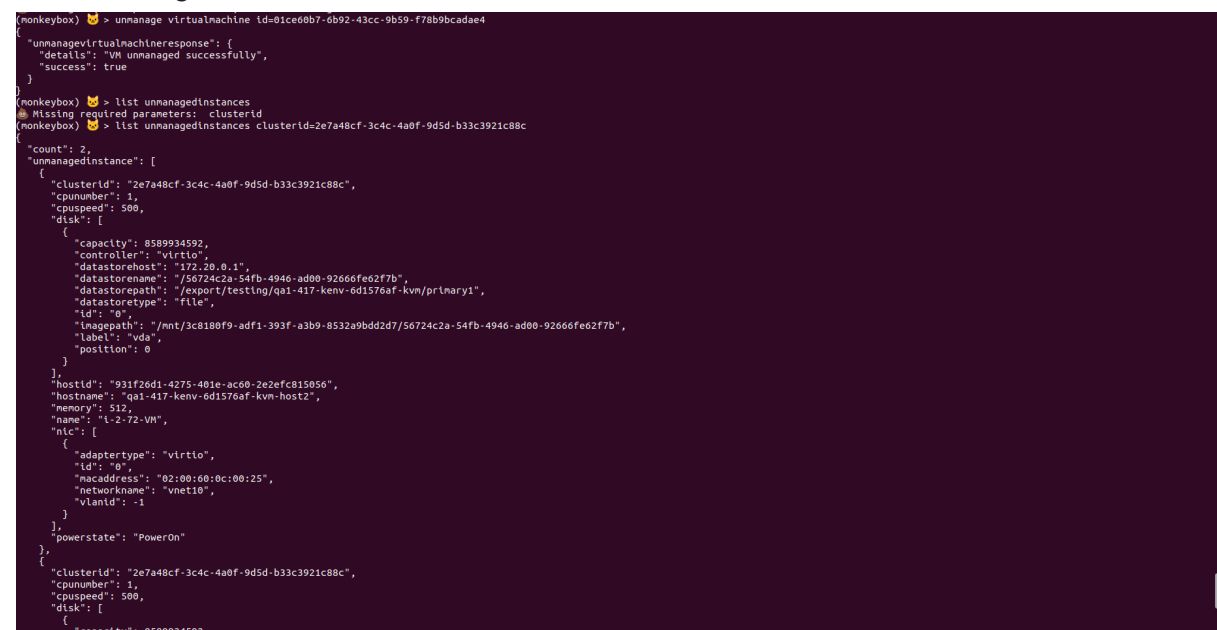
This API cleans up the cloudstack resources dedicated to managing the instance. It takes virtual machine id as parameter and then unmanage it by removing it from the cloudstack database. To implement this api, I tried to populate all the attributes of the UnmanagedInstanceTO class. For this, I created a wrapper class LibvirtGetUnmanagedInstancesCommandWrapper and inside this an unmanaged Instance function which returns an instance of UnmanagedInstanceTO class. Inside this function, I used parseDomainXML() inside LibvirtDomainXMLParser class which will parse the domain xml and will return the attributes of the virtual machine. I wrote the function extractCpuTuneDef() and extractCpuModeDef() which returns the attribute from managed VM xml and parse it and return value such as Shares,Quota,Period which is use to set the VM attributes such as CpuCoresperSocket,cpuSpeed etc.

importUnmanagedInstances API: This API imports an unmanaged Virtual machine into cloudstack from the above list. The management server can populate the details relevant for importing the virtual machine and managing it with the virtual machine's XML configuration. To implement it, I populated all the required parameters such as templateid, serviceofferingid, nicnetworklist, nicipaddresslist from unmanageinstanceTO class and the properties which can't be fetched from TO object would be given by the user during import/managing an unmanaged instance.

I also made changes in the UI for this API to be able to work/triggered from UI along with CLI.

Below is a screenshot for the API from CLI.

Here we can see the response of unmanageVirtualMachine and listUnmanagedInstances



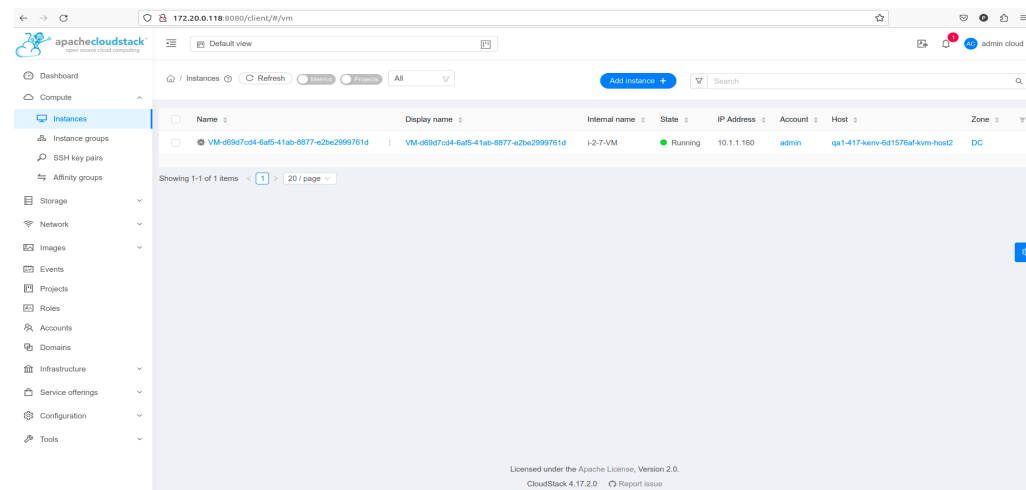
```
(monkeybox) ➤ unmanage virtualmachine id=01ce60b7-6b92-43cc-9b59-f78b9bcadae4
{"unmanagevirtualmachineresponse": {
  "details": "VM unmanaged successfully",
  "success": true
}}
(monkeybox) ➤ list unmanagedinstances
Missing required parameters: clusterid
(monkeybox) ➤ list unmanagedinstances clusterid=2e7a48cf-3c4c-4a0f-9d5d-b33c3921c88c
{"count": 2,
 "unmanagedinstance": [
  {
    "clusterid": "2e7a48cf-3c4c-4a0f-9d5d-b33c3921c88c",
    "cpunumber": 1,
    "cpuspeed": 500,
    "disk": [
      {
        "capacity": 8589934592,
        "controller": "virtio",
        "datastorehost": "172.20.0.1",
        "datastorename": "/56724c2a-54fb-4946-ad00-92666fe62f7b",
        "datastorepath": "/export/testing/q01-417-kenv-6d1576af-kvm/primary1",
        "datastoretype": "file",
        "id": "0",
        "imagepath": "/mnt/3c8180f9-adf1-393f-a3b9-8532a9bdd2d7/56724c2a-54fb-4946-ad00-92666fe62f7b",
        "label": "vda",
        "position": 0
      }
    ],
    "hostid": "931f26d1-4275-401e-ac00-2e2efc815056",
    "hostname": "q01-417-kenv-6d1576af-kvm-host2",
    "memory": 512,
    "name": "1-2-72-VH",
    "nic": [
      {
        "adaportype": "virtio",
        "id": "0",
        "macaddress": "02:00:00:0c:00:25",
        "networkname": "vnet10",
        "vlanid": -1
      }
    ],
    "powerstate": "PowerOn"
  },
  {
    "clusterid": "2e7a48cf-3c4c-4a0f-9d5d-b33c3921c88c",
    "cpunumber": 1,
    "cpuspeed": 500,
    "disk": [
      {
        "capacity": 8589934592,
```

```
(konkopyou) ➤ Import-UnmanagedInstance -ClusterName "cncf-k8s-3x4c-4d01-p9ds-63sc-92ic8dc-templateid74700005-2f11-11ee-9373-52540ba08b61c-name-l2-72-vn" -ServiceTier "mgmt-dmz8ac7e1e3e-429f-bfcs-3e0b119d-bda-nicnetworklist[0].nic="0" nicnetworklist[0].network=a09381c0-8f2c-4eac-8553-9fa5fd7e89d6 niclpadresslist[0].nic="0" niclpadresslist[0].ipAddress=auto
```

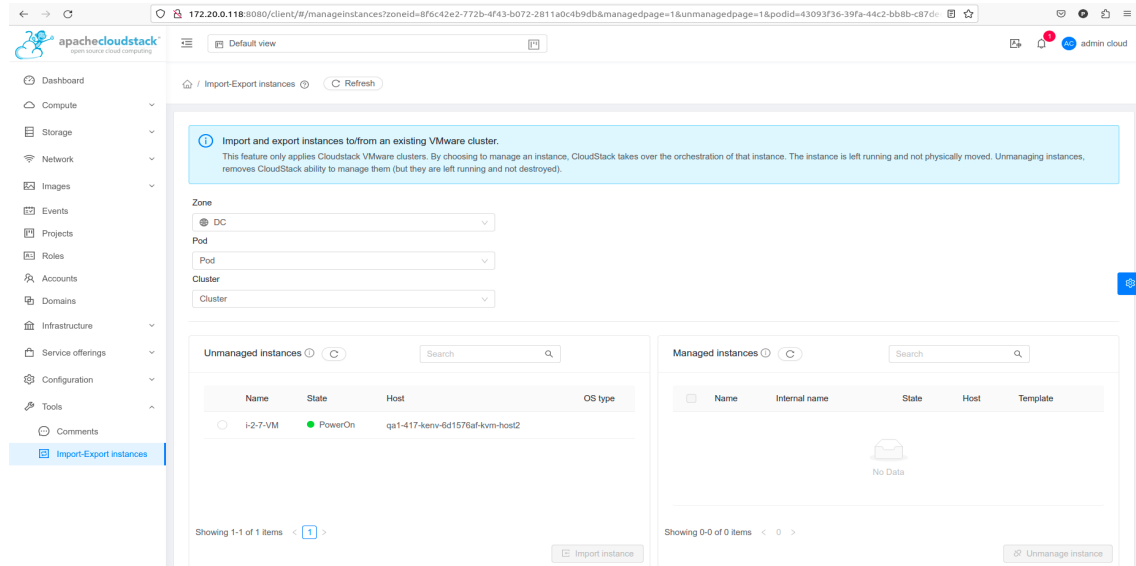
```
{  
  "virtualMachine": {  
    "account": "admin",  
    "affinitygroup": [],  
    "cpunumber": 1,  
    "cpspeed": 500,  
    "created": "2023-08-28T15:46:19+0000",  
    "details": {  
      "deployvm": "true",  
      "kvm.import.dynamicscaling": "false",  
      "kvm.vnc.password": "htg0ML6aaaaaaaaaaaaaa",  
      "nicAdapter": "virtio",  
      "rootDiskController": "virtio"  
    },  
    "displayName": "l-2-72-VN",  
    "displayvm": true,  
    "domain": "ROOT",  
    "donatId": "476db4de-2f11-11ee-9373-52540ba08b61c",  
    "guestosId": "476d3e16-2f11-11ee-9373-52540ba08b61c",  
    "haenable": false,  
    "hasannotations": false,  
    "hostId": "931f26d1-4275-401e-ac0b-2e2efc815856",  
    "hostname": "qal-i27-kenv-6d1576af-kvm-host2",  
    "hypervisor": "KVM",  
    "Id": "6abe1025-4a30-41c5-99c6-3483f629f6b2",  
    "instancename": "l-2-72-VN",  
    "Isdynamicallyscalable": false,  
    "memory": 512,  
    "name": "l-2-72-VN",  
    "nic": [  
      {  
        "deviceId": "0",  
        "extradhcpoption": [],  
        "gateway": "10.1.1.1",  
        "Id": "f02fed6e-ea23-49b5-9482-13b7e1d27a5b",  
        "Ipaddress": "10.1.1.12",  
        "Isdefault": true,  
        "macaddress": "02:00:00:0c:00:25",  
        "netmask": "255.255.255.0",  
        "networkId": "a09381c0-8f2c-4eac-8553-9fa5fd7e89d6",  
        "networkname": "test",  
        "secondaryIP": [],  
        "traffictype": "Guest",  
        "type": "isolated"  
      }  
    ],  
    "osdisplayname": "CentOS 5.5 (64-bit)",  
    "ostypeId": "476d3e16-2f11-11ee-9373-52540ba08b61c",  
    "passwordenabled": false,  
    "poolType": "NetworkFilesystem",  
    "provisioningMethod": "PXE"
```

I've finished the development part and currently my changes are being reviewed by my mentor and other contributors. Based on the feedbacks/inputs I would be refracting the code and then my changes will be merged.

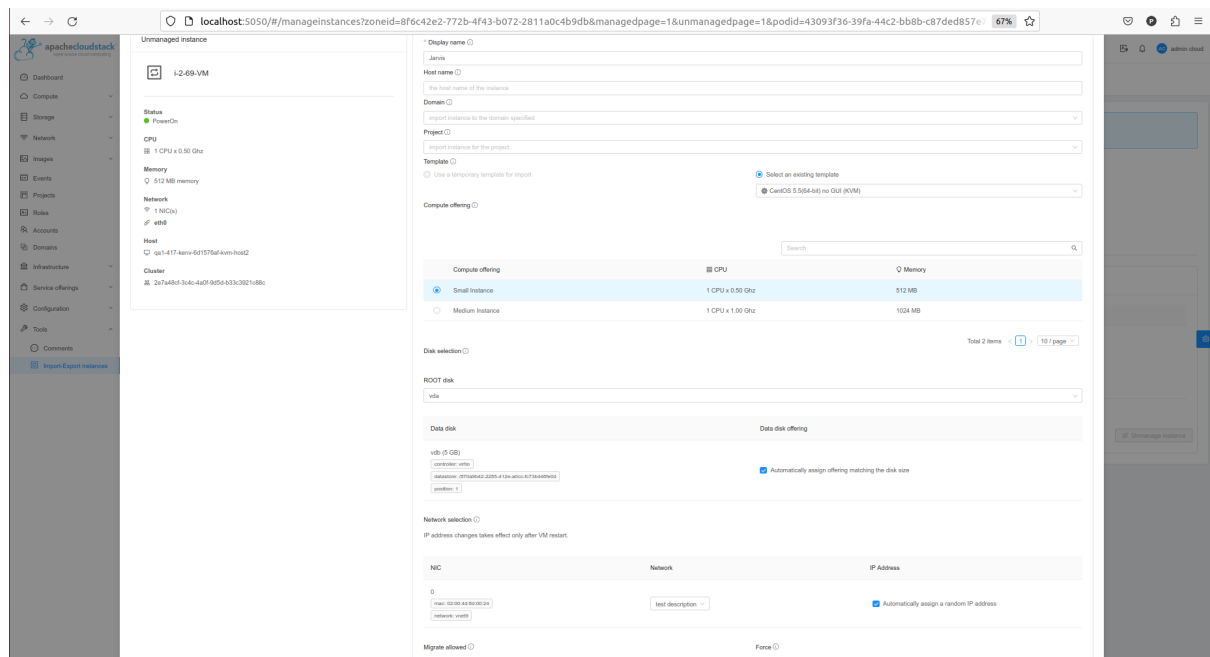
Below is the Image of the Instance that I created to test functionality of `unmanageVirtualMachine` API



As we can see the above VMs is unmanaged now and we can see all the unmanaged instances using list unmanage instances api



These are some fields which needs to be populated on click of import instance button to import the unmanaged instance



As you can see in the below picture the unmanage instance is managed now

Import and export instances to/from an existing VMware cluster.
This feature only applies Cloudstack VMware clusters. By choosing to manage an instance, CloudStack takes over the orchestration of that instance. The instance is left running and not physically moved. Unmanaging instances, removes CloudStack ability to manage them (but they are left running and not destroyed).

Zone: DC
Pod: Pod
Cluster: Cluster

Name	State	Host	OS type
No Data			

Showing 0-0 of 0 items

Import instance

Name	Internal name	State	Host	Template
<input type="checkbox"/> i-2-69-VM	i-2-69-VM	Running	qa1-417-kenv-6d1576af-kvm-host2	CentOS 5.5(64-bit) no GUI (KVM)

Showing 1-1 of 1 items

Unmanage instance

We can see the details of the managed instances in the instances section of compute tab

Instances / Jarvis

Status: Running

ID: 376e11f6-2e18-49c8-b455-03dcbf1e141

OS type: CentOS 5.5 (64-bit)

CPU: 0.5 Ghz (30.95% Used)

Memory: 512 MB memory (100.00% Used)

Disk size (in GB): 13.00 GB Storage

Network: eth0 10.1.1.222 (test) Default

IP address: 10.1.1.222

Template: CentOS 5.5(64-bit) no GUI (KVM)

OS type: CentOS 5.5 (64-bit)

Compute offering: Small Instance

Dynamically scalable: false

HA enabled: false

Hypervisor: KVM

Account: admin cloud

Future Goals :

- Currently there can only be one instance managed or unmanaged at a time. In future, implementing functionality for managing or unmanaging multiple VMs simultaneously.
- Adding more documentation for the feature I implemented

Conclusion :

I would like to thank my mentor Nicolas and Alex for their support and help throughout the project. There are also other contributors in the community, such as Daan Hoogland, Rohit Yadav and others, who have also helped me a lot. This summer vacation, I have witnessed the whole process/architecture of Infrastructure as a Service(IaaS) and how an open source works. Special thanks to the Apache SoftwareFoundation and the GSoC community for such a great opportunity to enter in the open Source realm.

Work Link Section:

Github Link of my work/commits in the Project:

<https://github.com/apache/cloudstack/pull/7712>

Github Link of the Project:

<https://github.com/apache/cloudstack/issues/7127>

GSoC Project Link :

<https://summerofcode.withgoogle.com/programs/2023/projects/f0gpheQM>