# Allocators

Allocators are part of the CloudStack Adapters. There are following categories of allocators currently supported:

* HostAllocators - Allows you to create custom rules to determine which physical host to allocate the guest virtual machines on.
* StoragePoolAllocators - Allows you to create custom rules to determine which storage pool to allocate the guest virtual machines on.

# Implementing a custom HostAllocator

* **HostAllocator Interface:**

HostAllocators are written by extending com.cloud.agent.manager.allocator.HostAllocator interface. The interface defines following two methods.

/\*\*

\* Checks if the VM can be upgraded to the specified ServiceOffering

\* @param UserVm vm

\* @param ServiceOffering offering

\* @return boolean true if the VM can be upgraded

\* \*/

**public** boolean isVirtualMachineUpgradable(final UserVm vm, final ServiceOffering offering);

/\*\*

\* Determines which physical hosts are suitable to allocate the guest virtual machines on

\*

\* **@param** VirtualMachineProfile vmProfile

\* **@param** DeploymentPlan plan

\* **@param** Type type

\* **@param** ExcludeList avoid

\* **@param** int returnUpTo

\* **@return** List<Host> List of hosts that are suitable for VM allocation

\*\*/

**public** List<Host> allocateTo( VirtualMachineProfile<?**extends** VirtualMachine> vmProfile, DeploymentPlan plan, Type type, ExcludeList avoid, **int** returnUpTo);

A custom HostAllocator can be written by implementing the ‘**allocateTo’** method

Input Parameters for the method ‘HostAllocator :: allocateTo’:

1. com.cloud.vm.VirtualMachineProfile vmProfile

VirtualMachineProfile describes one virtual machine. This allows the adapters like Allocators to process the information in the virtual machine and make determinations on what the virtual machine profile should look like before it is actually started on the hypervisor.

HostAllocators can make use of the following information present in the VirtualMachineProfile:

* The ServiceOffering that specifies configuration like requested CPU speed, RAM etc necessary for the guest VM.
* The VirtualMachineTemplate, the template to be used to start the VM.

1. com.cloud.deploy.DeploymentPlan plan

DeploymentPlan should specify:

* dataCenterId: The data center the VM should deploy in
* podId: The pod the Vm should deploy in; null if no preference
* clusterId: The cluster the VM should deploy in; null if no preference
* poolId: The storage pool the VM should be created in; null if no preference

1. com.cloud.host.Host.Type type

Type of the Host needed for this guest VM. Currently com.cloud.host.Host.Type interface defines the following Host types:

* Storage
* Routing
* SecondaryStorage
* ConsoleProxy
* ExternalFirewall
* ExternalLoadBalancer

1. com.cloud.deploy.DeploymentPlanner.ExcludeList avoid

The ExcludeList specifies what datacenters, pods, clusters, hosts, storagePools should not be considered for allocating this guest VM. HostAllocators should avoid the hosts that are mentioned in ExcludeList.hostIds

* Set<Long> dcIds;
* Set<Long> podIds;
* Set<Long> clusterIds;
* Set<Long> hostIds;
* Set<Long> poolIds;

1. int returnUpTo

This specifies return up to that many available hosts for this guest VM

To get all possible hosts, set this value to -1

* **Reference HostAllocator implementation**

Refer com.cloud.agent.manager.allocator.impl.FirstFitAllocator that implements the HostAllocator interface. This allocator checks available hosts in the specified datacenter, Pod, Cluster and considering the given ServiceOffering requirements.

If returnUpTo = 1, this allocator would return the first Host that fits the requirements of the guest VM.

* **Loading a custom HostAllocator**
* Write a custom HostAllocator class, implementing the interface described above.
* Package the code into a JAR file and make the JAR available in the classpath of the Management Server/tomcat.
* Modify the components.xml and components-premium.xml files found in /client/ tomcatconf as follows.

Search for ‘HostAllocator’ in these files.

<adapters key="com.cloud.agent.manager.allocator.HostAllocator">

<adapter name="FirstFit" class="com.cloud.agent.manager.allocator.impl.FirstFitAllocator"/>

</adapters>

Replace the FirstFitAllocator with your class name. Optionally, you can change the name of the adapter as well.

* Restart the Management Server.

# Implementing a custom StoragePoolAllocator

StoragePoolAllocators are written by extending com.cloud.storage.allocator. StoragePoolAllocator interface.

* **StoragePoolAllocator Interface:**

A custom StoragePoolAllocator can be written by implementing the ‘**allocateTo**’ method

/\*\*

\* Determines which storage pools are suitable for the guest virtual machine

\* @param DiskProfile dskCh

\* @param VirtualMachineProfile vmProfile

\* @param DeploymentPlan plan

\* @param ExcludeList avoid

\* @param int returnUpTo

\* @return List<StoragePool> List of storage pools that are suitable for the VM

\*\*/

**public** List<StoragePool> allocateToPool(DiskProfile dskCh, VirtualMachineProfile<? extends VirtualMachine> vm, DeploymentPlan plan, ExcludeList avoid, int returnUpTo);

This interface also contains some other methods to support some legacy code. However your custom allocator can extend the existing com.cloud.storage.allocator. AbstractStoragePoolAllocator. This class provides default implementation for all the other interface methods.

Input Parameters for the method ‘StoragePoolAllocator :: allocateTo’:

1. com.cloud.vm.DiskProfile dskCh

DiskCharacteristics describes a disk and what functionality is required from it. It specifies the storage pool tags if any to be used while searching for a storage pool.

1. com.cloud.vm.VirtualMachineProfile vmProfile

VirtualMachineProfile describes one virtual machine. This allows the adapters like Allocators to process the information in the virtual machine and make determinations on what the virtual machine profile should look like before it is actually started on the hypervisor.

StoragePoolAllocators can make use of the following information present in the VirtualMachineProfile:

* The VirtualMachine instance that specifies properties of the guest VM.
* The VirtualMachineTemplate, the template to be used to start the VM.

1. com.cloud.deploy.DeploymentPlan plan

DeploymentPlan should specify:

* dataCenterId: The data center the VM should deploy in
* podId: The pod the VM should deploy in; null if no preference
* clusterId: The cluster the VM should deploy in; null if no preference
* poolId: The storage pool the VM should be created in; null if no preference

1. com.cloud.deploy.DeploymentPlanner.ExcludeList avoid

The ExcludeList specifies what datacenters, pods, clusters, hosts, storagePools should not be considered for allocating this guest VM. StoragePoolAllocators should avoid the pools that are mentioned in ExcludeList.poolIds

* Set<Long> dcIds;
* Set<Long> podIds;
* Set<Long> clusterIds;
* Set<Long> hostIds;
* Set<Long> poolIds;

1. int returnUpTo

This specifies return up to that many available pools for this guest VM

To get all possible pools, set this value to -1

* **Reference StoragePoolAllocator implementation**

Refer com.cloud.storage.allocator.FirstFitStoragePoolAllocator that implements the StoragePoolAllocator interface. This allocator checks available pools in the specified datacenter, Pod, Cluster and considering the given DiskProfile characteristics.

If returnUpTo = 1, this allocator would return the first Storage Pool that fits the requirements of the guest VM.

* **Loading a custom StoragePoolAllocator**
* Write a custom StoragePoolAllocator class, implementing the interface described above.
* Package the code into a JAR file and make the JAR available in the classpath of the Management Server/tomcat.
* Modify the components.xml and components-premium.xml files found in /client/ tomcatconf as follows.

Search for ‘StoragePoolAllocator’ in these files.

<adapters key="com.cloud.storage.allocator.StoragePoolAllocator">

<adapter name="Storage" class="com.cloud.storage.allocator.FirstFitStoragePoolAllocator"/>

</adapters>

Replace the FirstFitStoragePoolAllocator with your class name. Optionally, you can change the name of the adapter as well.

* Restart the Management Server.