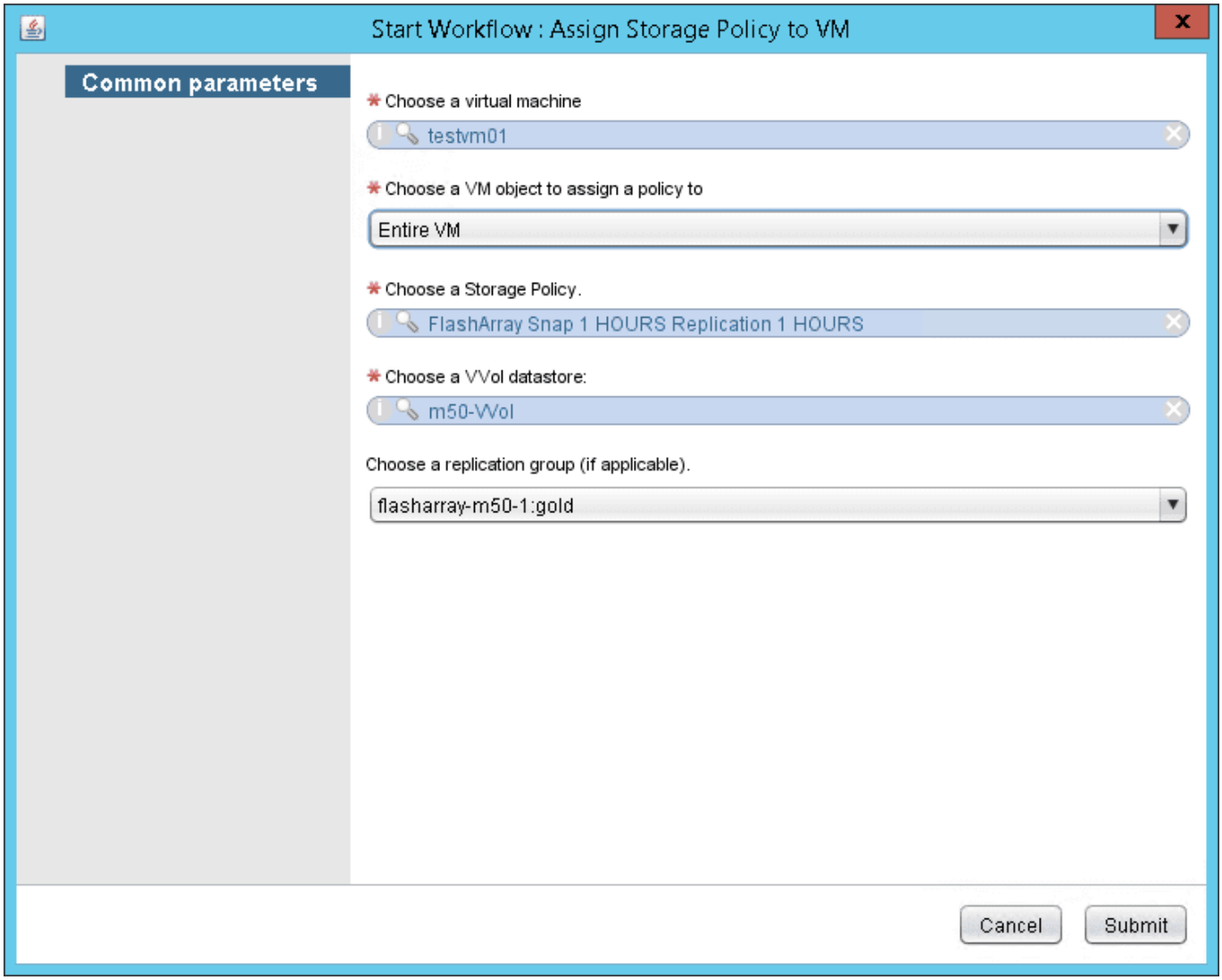
Assigning a VVol VM Storage Policy with vRO



Amidst writing a vMSC guide for our newly-introduced Active-Active replication called ActiveCluster, I have been taking some breaks to finish my vRealize Orchestrator Workflow Package for Virtual Volumes. I posted a starter post recently: Getting Started with vRealize Orchestrator and VVols

I am almost done with v1, but until then another starter post.

First, a demo:

Let me say first, I am a big fan of vRO–once you get the hang of it, it is a really powerful product and very flexible. I think the major advantage you have with it over say PowerCLI, is that it is much easier to build intelligent, GUI-based, wizard driven workflows. So if you want to hand off the workflows for others to run, they can do some clicking and selecting, not running a script and entering values. Furthermore, it can be leveraged in really cool ways in vRA:

vRealize Automation

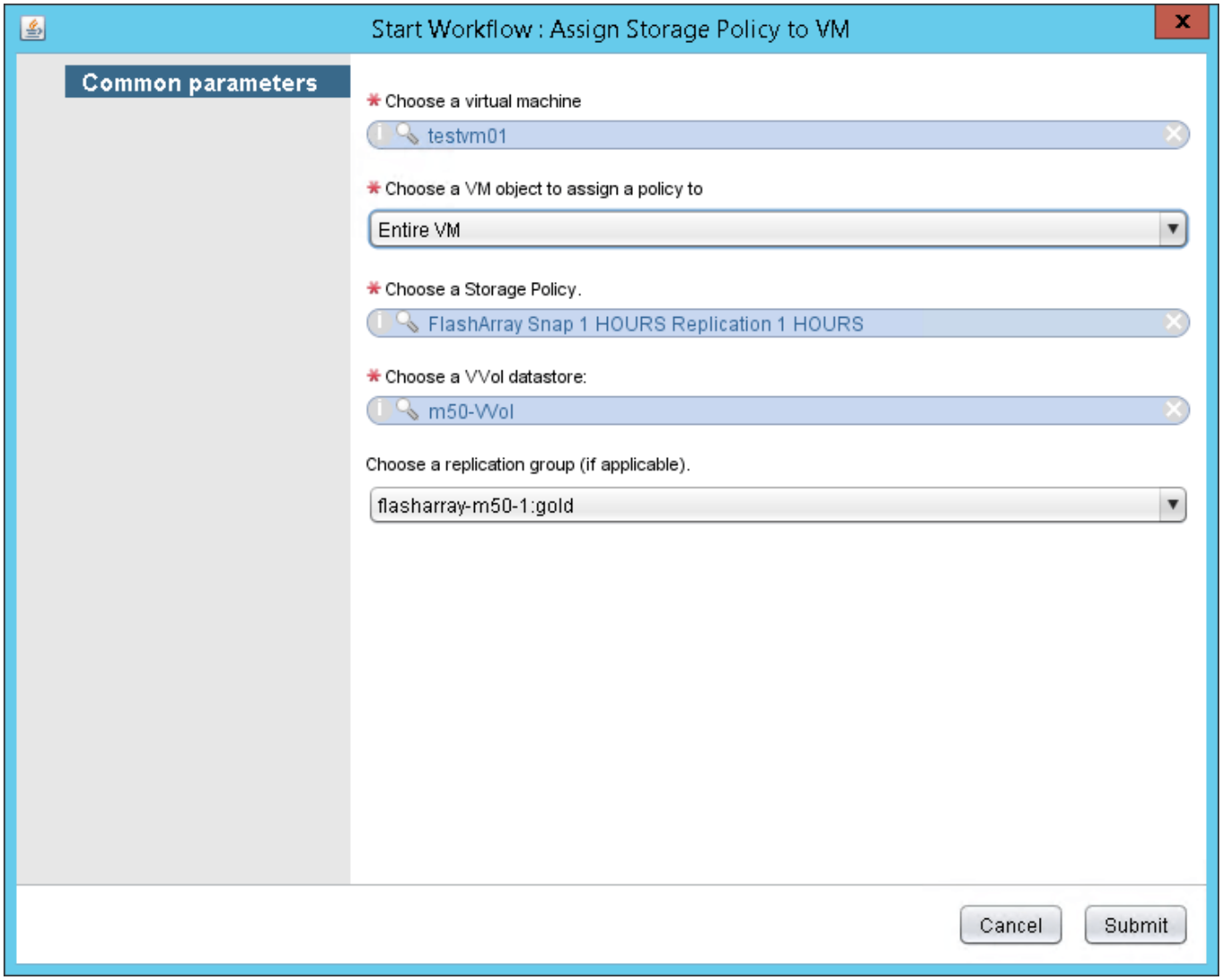
Or directly inside of the vSphere Web Client:

Building your own Web Client Plugin with vRO

Or in many other ways (it is REST controlled). Even better, it is included with vCenter, so no extra licensing. With all of the plugins offered and workflow packages, you likely don't really even have to do much custom work either.

Okay, so back to the post...

One of the major benefits of VVols is the ability to apply policies. I want this VM to be replicated. I want this virtual disk snapshotted once a week. Etc. You can assign VM policies in the vSphere Client, but it is pretty easy to do in vRO as well.



My workflow takes in a few things:

- A VM
- A selection (do you want to assign the policy to the whole VM, just the configuration VVol, or just a specific virtual disk (data VVol).
- A storage policy

- A VVol datastore–my inputs only returns VVol datastores that are compatible with the selected policy. User chooses one.
- If the policy includes replication/snapshot policies you enter in a replication group (aka a consistency group). The drop down shows all groups that are compatible with that policy from the FlashArray hosting that VVol datastore

The first part of the workflow Storage vMotions the object to the VVol datastore if it is not already there. So if it is on VMFS, it will convert it to VVols on that target datastore.

I won't get into detail on that part.

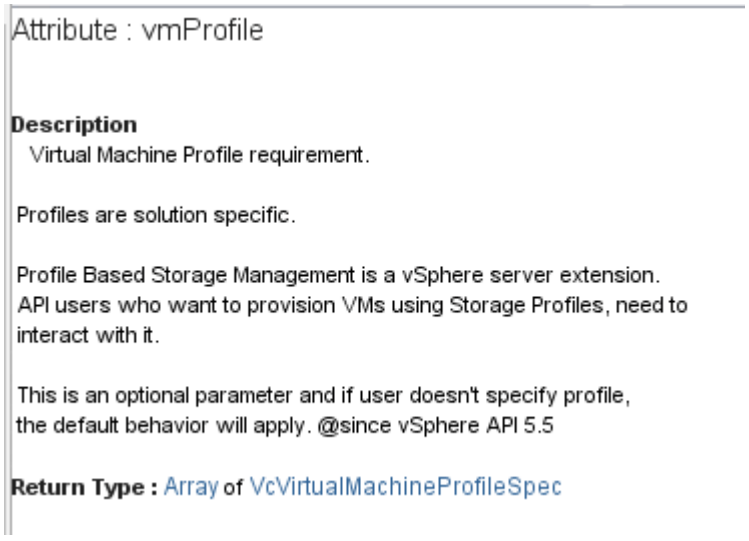
The next part assigns the policy. There are two ways of associating policies. You can use the PBM service directly and use the pbmAssociate() method to associate the policy with the object and then call pbmApplyAssociated() to actually have VASA apply the configuration.

This is unnecessarily complicated.

I recommend using the trusty reconfigVM_task. This is the method you use to add virtual disks, add a NIC, basically anything that changes the VM.
[crayon-6515bf7705718041089203/]
This is executed against a VM object. It takes in an object of type "VcVirtualMachineConfigSpec"

So you can instantiate one:
[crayon-6515bf7705724841399481/]
Then the next thing is you do want to change the policy of the config VVol (the home directory), the whole VM or a single virtual disk (data VVol)?

To change the home directory, you need to specify it in the vmProfile attribute.


Attribute : vmProfile

Description
Virtual Machine Profile requirement.

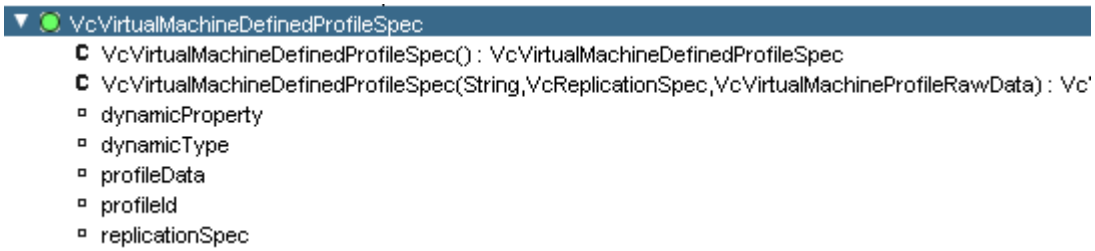
Profiles are solution specific.

Profile Based Storage Management is a vSphere server extension.
API users who want to provision VMs using Storage Profiles, need to interact with it.

This is an optional parameter and if user doesn't specify profile, the default behavior will apply. @since vSphere API 5.5

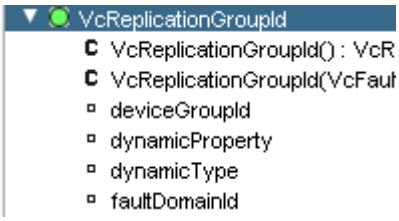
Return Type : Array of VcVirtualMachineProfileSpec

This asks to be populated with an array of objects of the type "VcVirtualMachineDefinedProfileSpec". Yes, technically above it says VcVirtualMachineProfileSpec, but the defined version extends that type and it the one you want to use for VM storage policies with VVols.



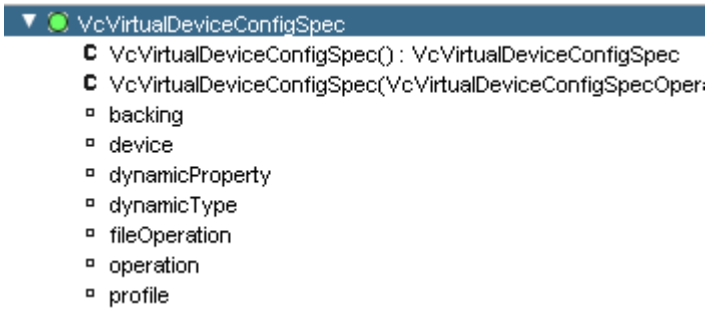
In this, you need to populate one or two attributes. You need profileID for sure. You can get this from the VM storage policy object (type VcPbmProfile) in the attribute myPolicy.profileId.uniqueId

If you policy also needs a replication group you can get all of the compatible replication groups by using the PBM manager as explained in the previous blog post and the method:
[crayon-6515bf7705727223305771/]
Pass in your datastore info and the storage policy and it will return any compatible replication groups. Choose one and pass it in the replicationSpec like so:
[crayon-6515bf7705728828953591/]
The replication group ID should be that object type. Which will be part of the pbmCheckCompatibility response.



Then put it all together:
[crayon-6515bf770572a764038297/]
The vmProfile requires this be in an array, so I will create an empty array and put the single entry in it:
[crayon-6515bf770572c811911489/]
Then finally I add it to the vmProfile attribute:
[crayon-6515bf770572d887373172/]
So now I can pass that into my reconfigVM_task but if I want to change my virtual disks too, I can add that in. This would go into myVcVirtualMachineConfigSpec.deviceChange.

In this, you set the deviceChange attribute equal to an array of var VcVirtualDeviceConfigSpec.



For each virtual disk you want to change, create another one of these and push it into the array that will be the deviceChange attribute.

In here, pass in the device. Which is the instance of virtualDisk. Then the profile (which is the same

as changeSpec above) and make the operation equal to edit.
[crayon-6515bf770572f996400600/]
Do that for each device and then pass it into the deviceChange attribute.
[crayon-6515bf770573c061227509/]
Now that you have configured the config and data VVols (you don’t have to do both, or all of the devices, just choose what you want) you can run the reconfigVM_task.
[crayon-6515bf770573e905219158/]
And you are done! That associates the profile and then applies the configurations too.

I hope to have my workflow package out soon, so stay tuned! I am almost done.