> We want to extend sample-controller to operate VM resource in Private Cloud through k8s.
>
> Precondition and Requirement are following, please consider/implement solution to satisfy requirement. you don't need to create a real cloud API server to provide vm service. If you want, you can just create a mock server to simulate API behavior.
>
> Requirement
>
> Custom Resource schema should be like following
>
> ```
> apiVersion: samplecontroller.k8s.io/v1alpha1
> kind: VM
> metadata:
>     name: sample
> spec:
>     vmname: vmsample # any name
> status:
>     vmId: XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXXX #uuid
>     cpuUtilization: 50 #0-100
> ```
>
> Controller should create VM by calling API which is provided by Private Cloud If new VM kind resource has been detected

corresponding code:

```
func (c *Controller) onVMAdd(obj interface{}) {
    var key string
    var err error
    if key, err = cache.MetaNamespaceKeyFunc(obj); err != nil {
        utilruntime.HandleError(err)
        return
    }

    klog.Infof("VM '%s' was added, enqueue it for submission.", key)
    c.workqueue.Add(key)
}

... ...

    switch vm.Status.OpStatus {
    case samplev1alpha1.NewState:
        err = c.createVmAndUpdateCrStatus(vm)
    case samplev1alpha1.FailedState, samplev1alpha1.UnknownState:
        klog.Infof("VM in %s, try to create again", vm.Status.OpStatus)
        err = c.createVmAndUpdateCrStatus(vm)
```

> Some vmname is prohibited to use for political reason. So Need to check vmname is available or not by calling VM Name Availability Check API before create custom resource.

corresponding code

```
func (c *Controller) createVmAndUpdateCrStatus(vm *samplev1alpha1.VM) error {
    //vm name check
    if !c.vmManager.CheckNameIsOk(vm.Spec.VMName) {
        vmStatus := &samplev1alpha1.VMStatus{
            OpStatus: samplev1alpha1.FailedState,
            Msg: "checkName failed",
            CPUUtilization: "",
            VMId: "",
        }
        _ = c.updateVmStatus(vm, vmStatus)
        return fmt.Errorf("checkName failed: " + vm.Spec.VMName)
    }

    vmResponse := c.vmManager.CreateVm(vm)
    if vmResponse.success {
        vmStatus := &samplev1alpha1.VMStatus{
... ...
```

> vmId: This filed will not be changed after VM got created, You can find vmId in the response of VM creation API. cpuUtilization: This field should be updated periodically, You can get utilization information by calling VM Status Get API.

corresponding code

```
    vmResponse := c.vmManager.CreateVm(vm)
    if vmResponse.success {
        vmStatus := &samplev1alpha1.VMStatus{
            OpStatus: samplev1alpha1.RunningState,
            VMId: vmResponse.vmEntity.id,
            CPUUtilization: vmResponse.vmEntity.cpuUtilization,
            Msg: "CreateVm success",
        }
        err := c.updateVmStatus(vm, vmStatus)
        return err
    } else {
        vmStatus := &samplev1alpha1.VMStatus{
            OpStatus: samplev1alpha1.FailedState,
            Msg: vmResponse.msg,
            VMId: "",
            CPUUtilization: "",
        }
        _ = c.updateVmStatus(vm, vmStatus)
        return fmt.Errorf("error msg: %s", vmResponse.msg)
    }

... ...

// periodically update the VM cpu utilization and update the Status field
func (c *Controller) updateVmCpuUtilization() {
    var ret []*samplev1alpha1.VM
    var err error

    ret, err = c.vmsLister.List(labels.Everything())
    if err != nil {
        return
    }

    for _, val := range ret {
        if val.Status.VMId == "" {
            continue
        }

        vmResponse := c.vmManager.GetVm(val.Status.VMId)
        if !vmResponse.success {
            continue
        }

        vmStatus := &samplev1alpha1.VMStatus{
            VMId:           val.Status.VMId,
            CPUUtilization: vmResponse.vmEntity.cpuUtilization,
            OpStatus:       val.Status.OpStatus,
            Msg:            val.Status.Msg,
        }
        _ = c.updateVmStatus(val, vmStatus)
    }
}
```

> When user delete VM kind resource on k8s, Controller should delete VM by calling VM Delete API and make sure VM get deleted before VM kind resource get deleted

use Finalizer in CRD, corresponding code

```
func (c *Controller) onVMAdd(obj interface{}) {
    vm := obj.(*samplev1alpha1.VM)

    spaceKey, err := cache.MetaNamespaceKeyFunc(vm)
    if err != nil {
        utilruntime.HandleError(err)
        return
    }

    //add finalizer to do pre delete actions
    deepCopy := vm.DeepCopy()
    deepCopy.Finalizers = []string{VMFinalizerName}
    _, updateErr := c.sampleclientset.SamplecontrollerV1alpha1().VMs(deepCopy.Namesp
    if updateErr != nil {
        utilruntime.HandleError(updateErr)
        return
    }
... ...
func (c *Controller) onVMUpdate(oldObj, newObj interface{}) {
    oldVm := oldObj.(*samplev1alpha1.VM)
    newVM := newObj.(*samplev1alpha1.VM)

    spaceKey, err := cache.MetaNamespaceKeyFunc(newVM)
    if err != nil {
        utilruntime.HandleError(err)
        return
    }

    //pre delete actions
    if !newVM.ObjectMeta.DeletionTimestamp.IsZero() {
        vmRes := c.vmManager.DeleteVm(newVM.Status.VMId)
        if vmRes.success {
            deepCopy := newVM.DeepCopy()
            deepCopy.Finalizers = []string{}
            _, _ = c.sampleclientset.SamplecontrollerV1alpha1().VMs(newVM.Namespace)
        }
        return
    }
```