Project 1
Disaster Recovery System

Submitted

By

Akshay Wattal

akshay.wattal@gmail.com

INTRODUCTION

Disaster Recovery has become an important part of todays computing world. Where, computing has now moved from physical servers to cloud and virtual machines (VM). This report describes the solution architecture of an Availability Manager, which is capable of monitoring the VMs health running at a site. And in case of failures of either the site or the host, is able to recover and provision both.

Goals

The goals of this project are:

- Monitor VMs health and capture VM statistics.
- Have the capability to take periodic snapshots of VMs for provisioning.
- Intelligent recovery and provisioning of vHost and VM incase of failure.
- Instate proper Alarm Management.

Objective

To build an Availability Manager, that monitors the VM for failure and printouts its vital statistics. It should recover and provision the VM using snapshots. The Availability Manager should use implement this solution using the VMware Infrastructure APIs.

Background:

- All the vHost and the virtual machines are managed by centralized application vCenter Server by VMware.
- vCenter manages multiple ESX servers and virtual machines (VMs) using a single vSphere client, which is a bottleneck.
- Therefore, we need a automated disaster recovery system that is capable to monitor the health of virtual machines running and takes appropriate recovery steps to sustain any damage at minimal level.

Requirements:

Functional Requirements:

- The VM is considered alive if it responds to ping in specific configurable time. In case, it doesn't respond its considered dead.
- On VM failure, first the vHost containing it is to be checked, if it responds to ping then the VM is to be provisioned using Snapshot.
- On VM failure, first the vHost containing it is to be checked, if it doesn't respond to ping then try to make it Alive using vHost snapshot.
- On VM failure, first the vHost containing it is to be checked, if it doesn't responds to ping and trying to bring it Alive fails, then the VM is to be provisioned using Snapshot on another vHost that is alive. If, no vHost is alive then a new vHost needs to be added.
- The application should gather all statistics such as CPU, I/O and Network for a VM and display it.
- Snapshot of the VM should be taken at every configurable interval.
- Alarm should be created and triggered on when VM is power off by User. In this scenario the solution should prevent failover from occurring.
- New vHost and VM added should be monitored without re-starting the application.

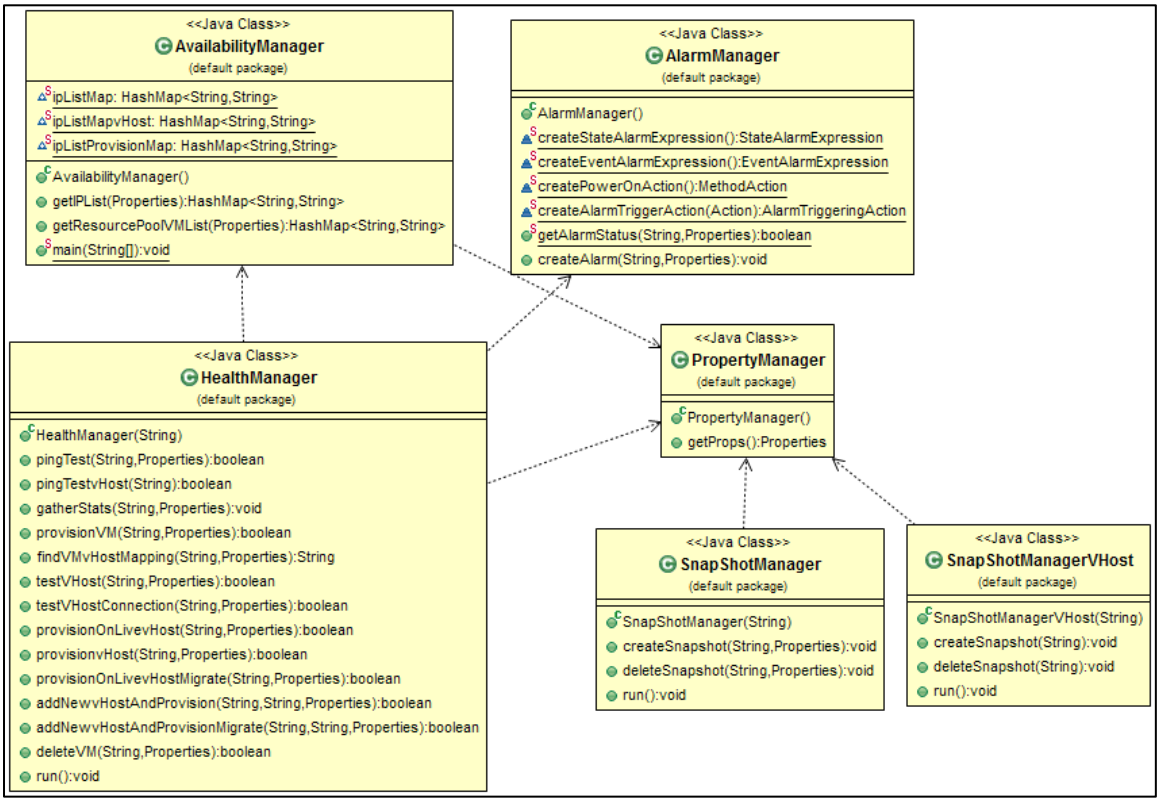
Non-Functional Requirements:

- The solution should behave in a consistent manner for similar use-cases.
- Threading should be handled gracefully and not result in deadlock.
- Exceptions triggered should be handled gracefully.
- The path chosen to recover and provision the VM should be the most optimized one.

Design

Architecture Class Diagram:

Below is the class diagram of the solution architecture, each of the components are explained in detail in the next section.



Components:

1. Availability Manager (AM)

   In current solution, Availability Manager (AM) is responsible for getting the inventory list in terms of vHost and its VMs. Each VM is identified using a key value pair i.e. name and IP mapping, which is stored in distributed hash map. The VMs that do not have an IP assigned are not considered as a part of this solution. Once, the list of VMs present in the inventory is fetched, the AM does the primary task of spanning Health Manager thread for each virtual machine. It also spawn two more threads for virtual machine snapshot and vHost snapshot i.e. Snapshot Manager and Snapshot vHost Manager respectively.

   Total Number of Threads-

   - One per each virtual machine in the inventory for monitoring (Health Manager)
   - One for Snapshot Manager
   - One for Snapshot vHost Manager

The Availability Manager also checks after a configurable interval of time if any new vHost and VMs are added. In scenarios, when new vHost and VM is added, a thread is spawned for the *new VM only*. This functionality is achieved using distributed hash map that keeps track of old and new VMs.

2. Health Manager (HM)
   Health Manager is the core for this project. It performs the following tasks:
   - Creates an alarm for the virtual machine using Alarm Manager. Incase, an alarm is already present, the alarm is not created again.
   - Gather vital status of the virtual machine, such as I/O, CPU, Network and Storage.
   - Performs ping test on the Virtual Machine and vHost.
   - Checks Alarm status of the virtual machine.
   - Performs recovery and provisioning using Cloning with Snapshots and Cold Migration, depending on the failure scenario of vHost and VM.
   - Function to add a new vHost that is already present in the inventory, but not added to the vCenter.

   The Health Manager thread is specific for each virtual machine and monitors the VM's health after every configurable amount of time.

3. Snapshot Manager (SM)
   Snapshot Manager is responsible for maintain the snapshot of the virtual machines. It performs two main functions:
   - Delete all previous snapshots (this is done to overcome resources constraints.)
   - Create a new snapshot

4. Snapshot Manager vHost (SMV)
   Snapshot Manager is responsible for maintain the snapshot of the vHost machines. It performs two main functions:
   - Delete all previous snapshots (this is done to overcome resources constraints.)
   - Create a new snapshot

5. Alarm Manager (AlM)
   Alarm Manager is responsible for event and alarm management of the virtual machine. It performs the following tasks.
   - Create a new alarm for the VM
   - Check alarm state for the VM

6. Property Manager
   Property Manger is responsible for reading the configurable properties such as vCenter URL, username and password, sleep interval for monitoring the VM and managing the snapshot interval.
   It has been designed such that, even if during the program execution the property is changed, it is able to read the new values.

Solution Workflow Design:

1. VM Failure detection process
   If the VM doesn't respond back to the ping for a configurable period of time, it is considered dead.

2. vHost Failure detected process
   If the vHost is not connected to the vHost it is considered dead irrespective its responding to ping or not.

3. Add/Remove new vHost
   The details of the vHost to be added to the vCenter are configured in the properties file. Property Manager is responsible for reading the values from it. These values are then passed to the function in Health Manger responsible for adding new vHost. Configuration such as HostConnectSpec and ComputeResourceConfigSpec are used.
   For removing the vHost, the details of the vHost is passed to HealthManger's function responsible for deleting.

4. Try to make vHost alive
   Health Manger uses the snapshot of the vHost to recover the vHost first.

5. Clone using Snapshot
   Health Manger performs cloning using snapshot of VM.

6. Migrate using Snapshot
   Health Manger performs migration using snapshot of VM.

Key Workflow:

Below are the three key scenarios and the workflow that is taken:

Scenario 1: vHost is Running, VM is Running
Action: No action is performed by Health Manger.

Scenario 2: vHost is Running, VM is Down (i.e. not responding to ping)
Action:

1. Health Manager first gets the alarm status.
2. If user had shutdown the application, then no action is performed. However, if the machine was shutdown because of failure, it goes to next step.
3. Check If vHost is powered on and responding to ping
    - If Yes-Provision the VM using clone with snapshot (here, snapshot is catered by Snapshot Manager) and delete the stale VM.
    - If No- Go to next step

Scenario 3: vHost is Down/dis-connected, VM is Down (i.e. not responding to ping)
Action:

1. Health Manager first gets the alarm status.
2. If user had shutdown the application, then no action is performed. However, if the machine was shutdown because of failure, it goes to next step.
3. Check If vHost is powered on and responding to ping
    - If Yes-Provision the VM using clone with snapshot and delete the stale VM.
    - If No- Go to next step
4. Try to make the vHost alive using snapshot (here, snapshot is catered by Snapshot Manager vHost)
    - If Successful-Provision the VM using clone with snapshot and delete the stale VM.
    - If Failed- Go to next step

5. Check If another vHost is alive and the current vHost is *connected*
    a. If Yes-Provision the VM using **clone** with snapshot and delete the stale VM.
    b. If No- Add a new vHost and provision the VM using **clone** with snapshot and delete the stale VM.

6. Check If another vHost is alive and the current vHost is *dis-connected*
    a. If Yes-Provision the VM using **migration** with snapshot.
    b. If No- Add a new vHost and provision the VM using **migration** with snapshot.

Implementation

Environment:

The test environment for this solution is as follows:

- The resource pool is central – Team01_vHOSTS contains three vHost. All these three vHosts are in running state.
- In the vCenter – 130.65.132.101 only two of the above mentioned vHost are connected and have VMs running under it. The third vHost is not connected and act as fall back in case a new vHost needs to be added to the vCenter. The name and SSL of this fall back vHost is configured in the properties file.

Tools:

The solution was developed using VMware Infrastructure APIs, Java and Eclipse IDE.

Approaches:

The solution uses the following classes and concepts:

Classes-

1. AvailabilityManager.java – Responsible for performing the Availability Manager tasks explained in Design section.
2. HealthManager.java – Responsible for performing the Health Manager tasks.
3. SnapShotManager.java - Responsible for performing the Snapshot Manager tasks.
4. SnapShotManagerVHost.java - Responsible for performing the Snapshot Manager vHost tasks.
5. AlarmManager.java - Responsible for performing the Alarm Manager tasks.
6. PropertyManager.java - Responsible for performing the Property Manager tasks.

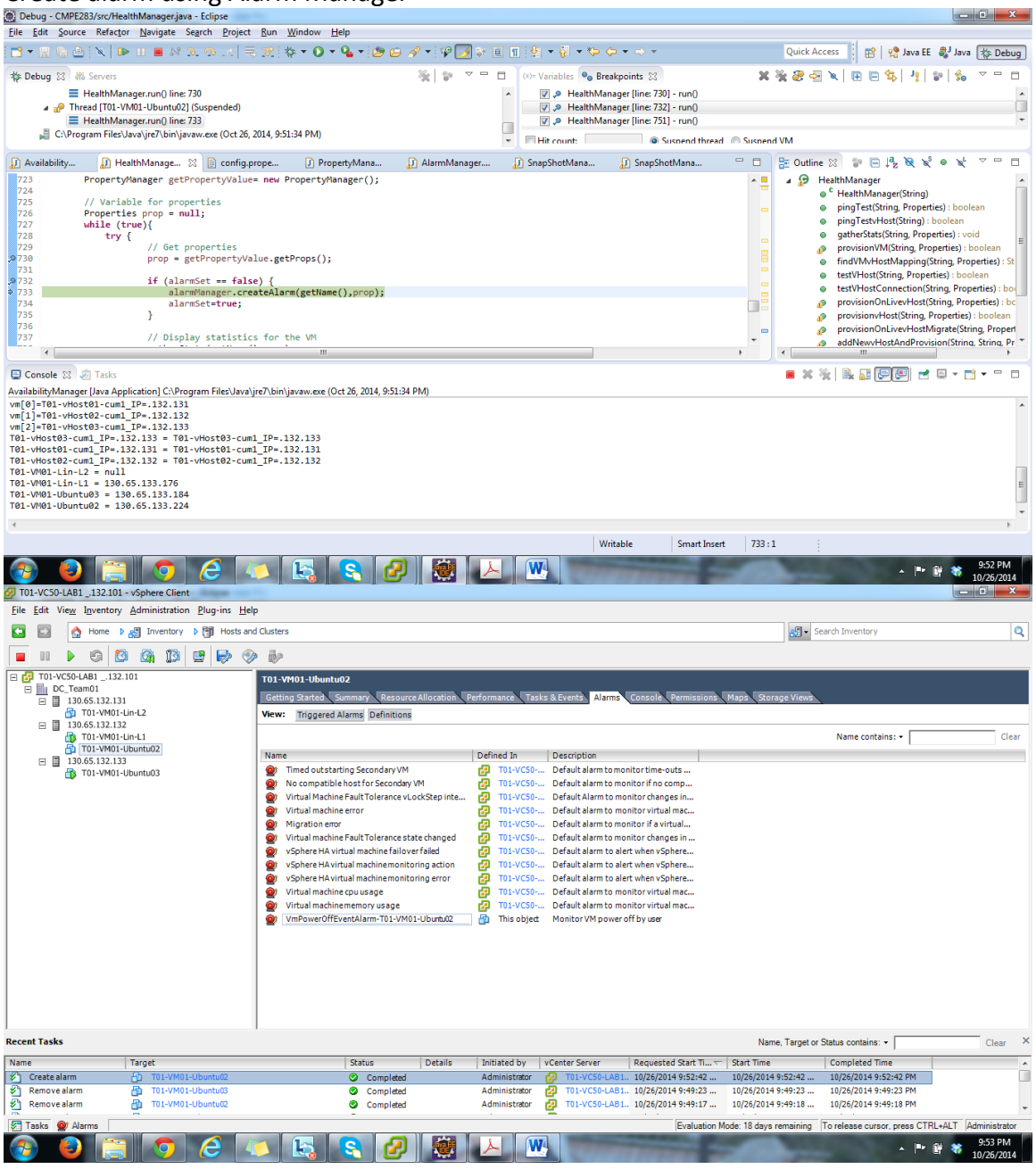
Concepts-

1. Threading for HealthManager, SnapShotManager and SnapShotManagerVHost.
2. Static distributed hash map for storing VM's name & IP, vHost's name and list of VMs to be monitored.

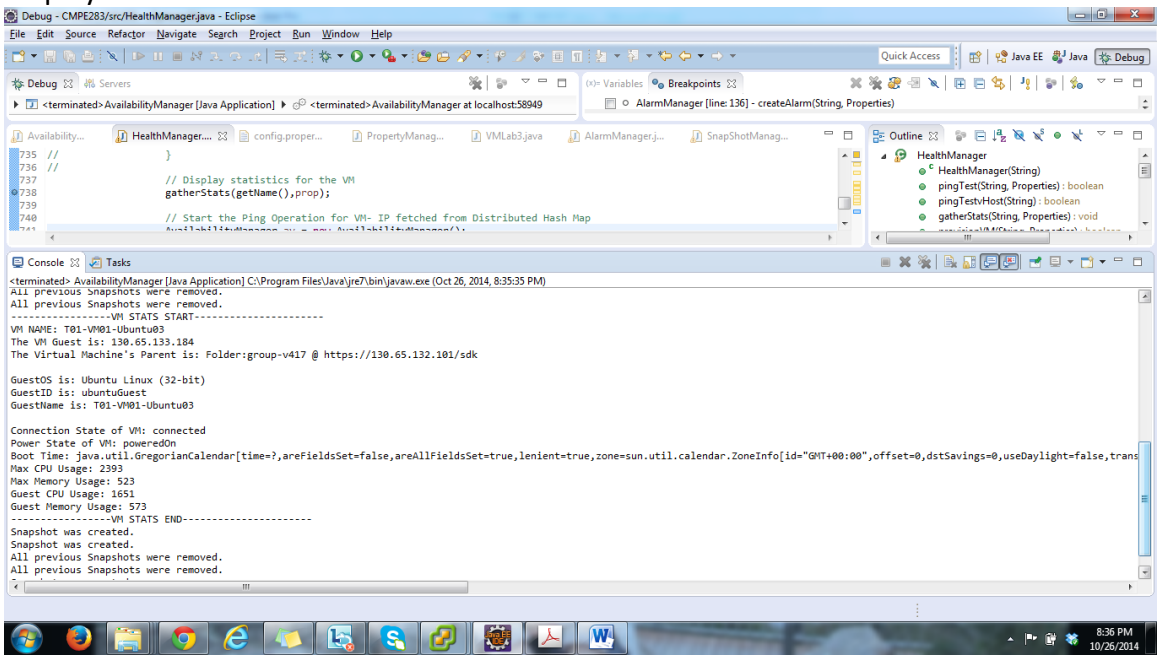
Screenshots:

Below are the screenshots of different scenarios:

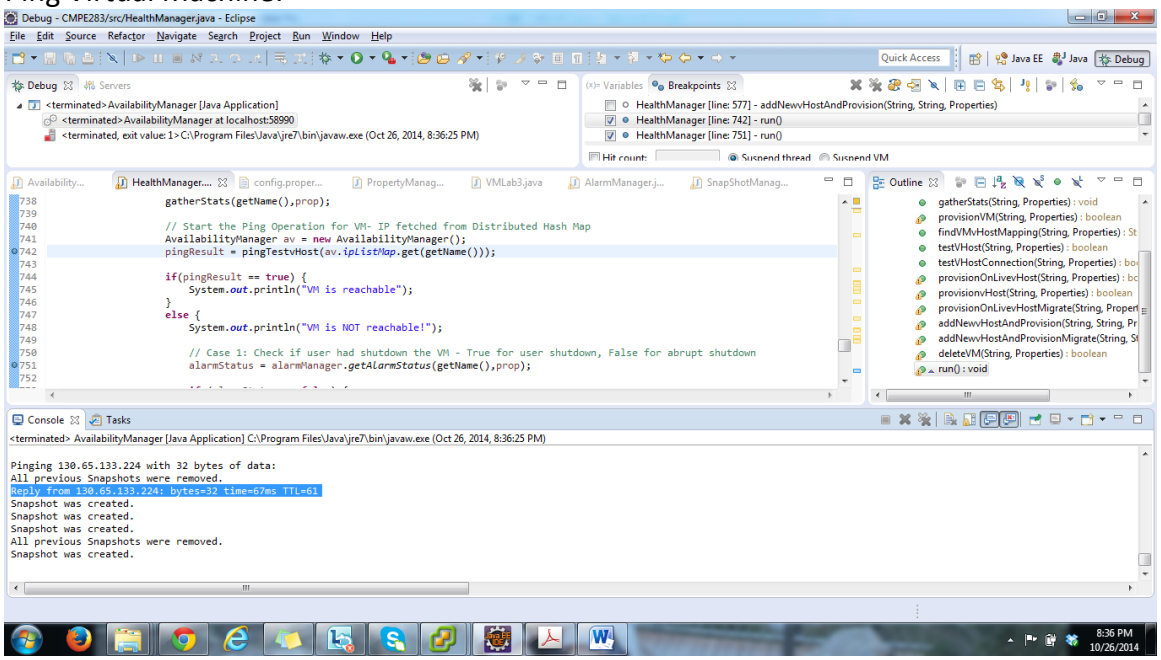
Create alarm using Alarm Manager



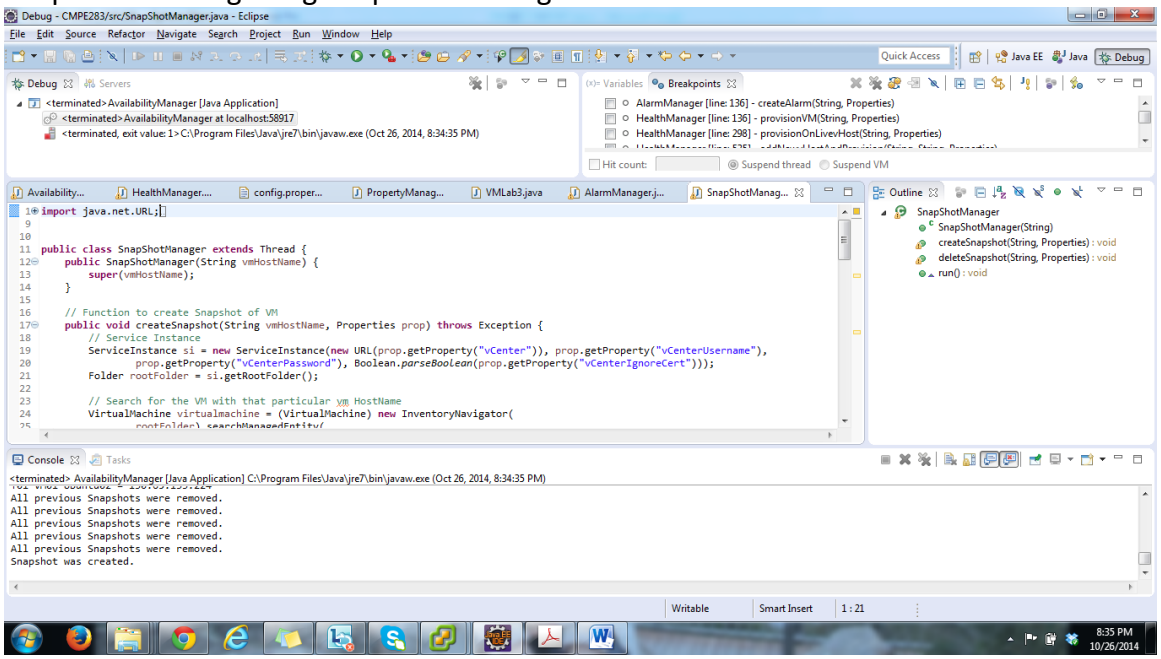
Display Statistics:

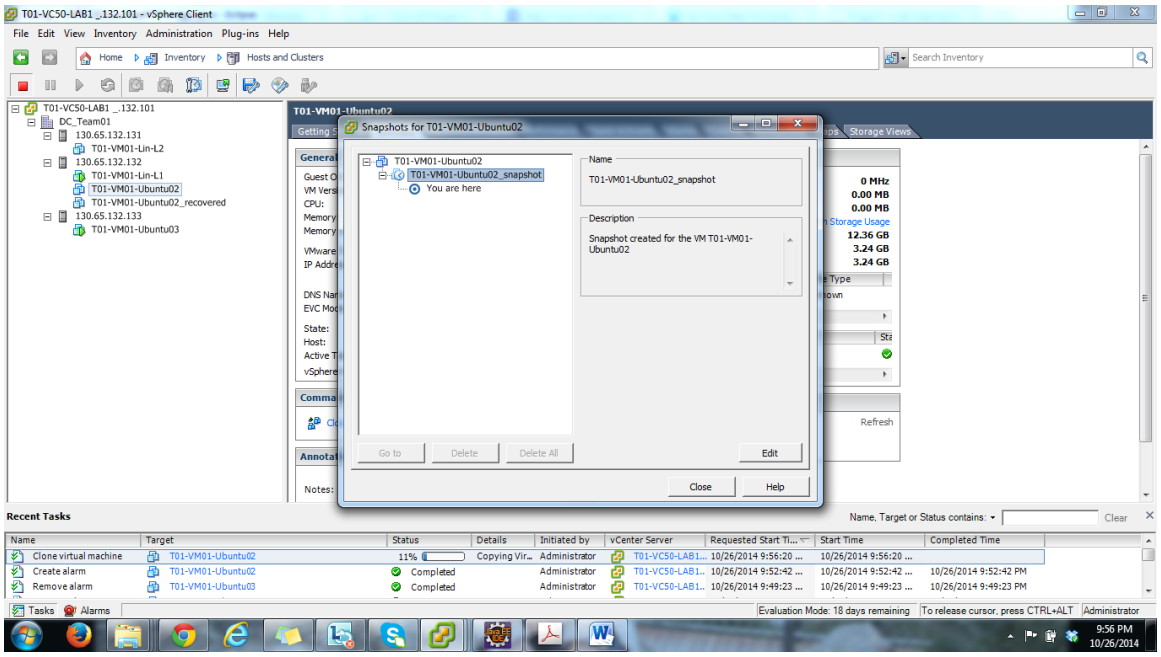


Ping Virtual Machine:

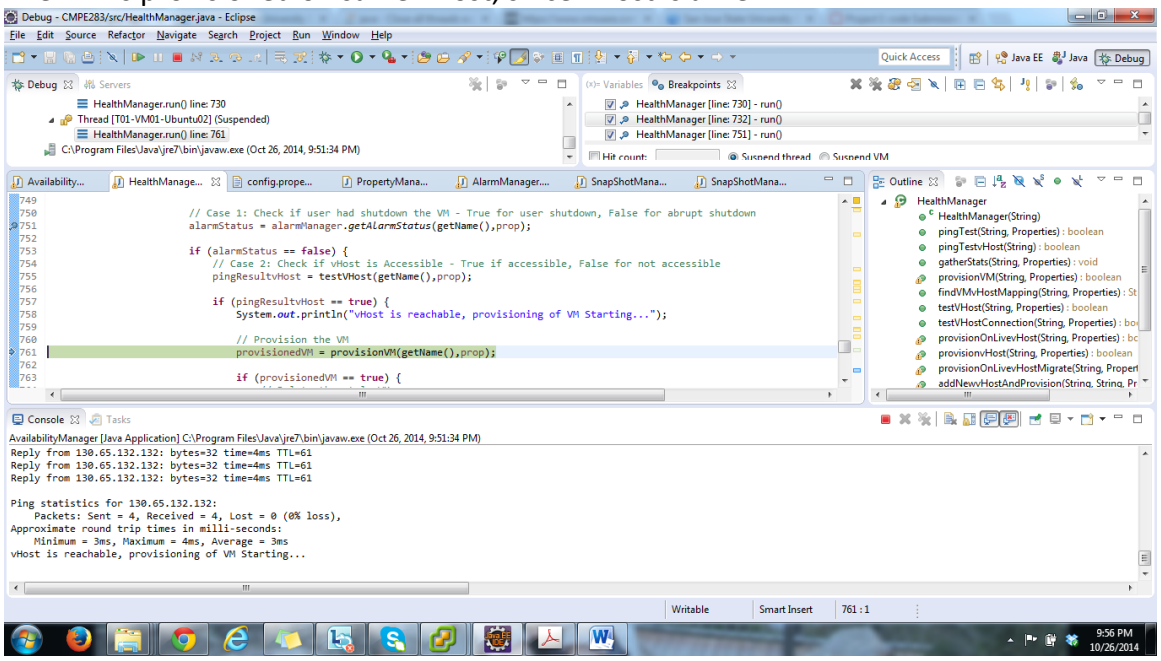


Snapshot Creating using Snap Shot Manager:

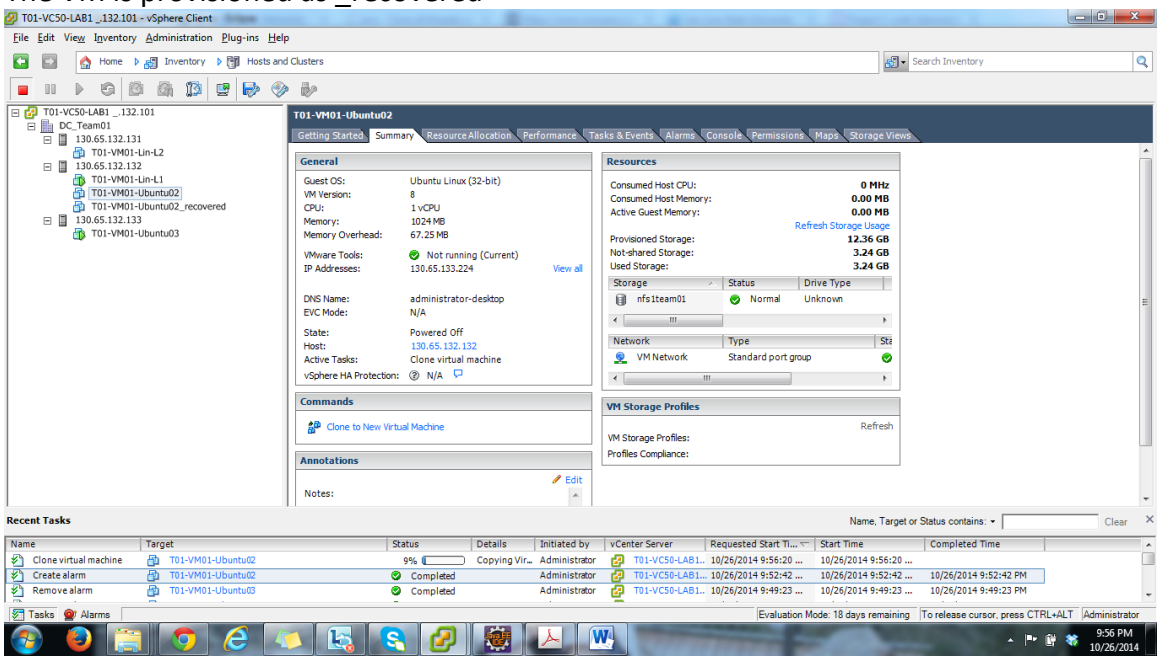




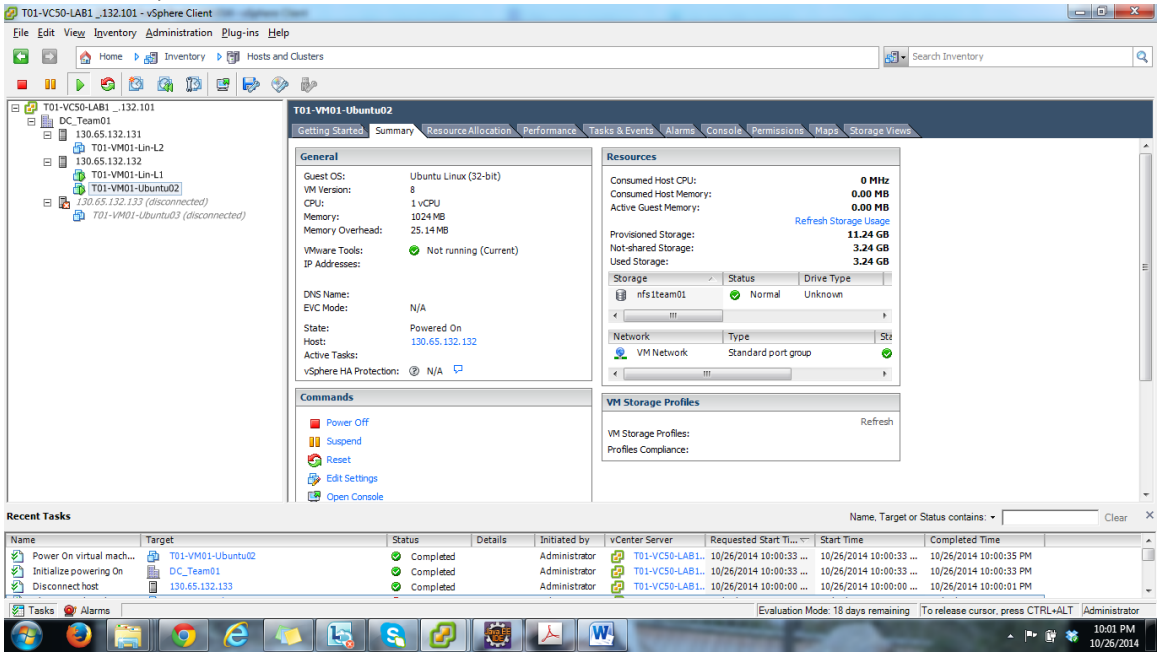
VM is down, vHost is Running:
The VM is provisioned on same vHost, since vHost is alive



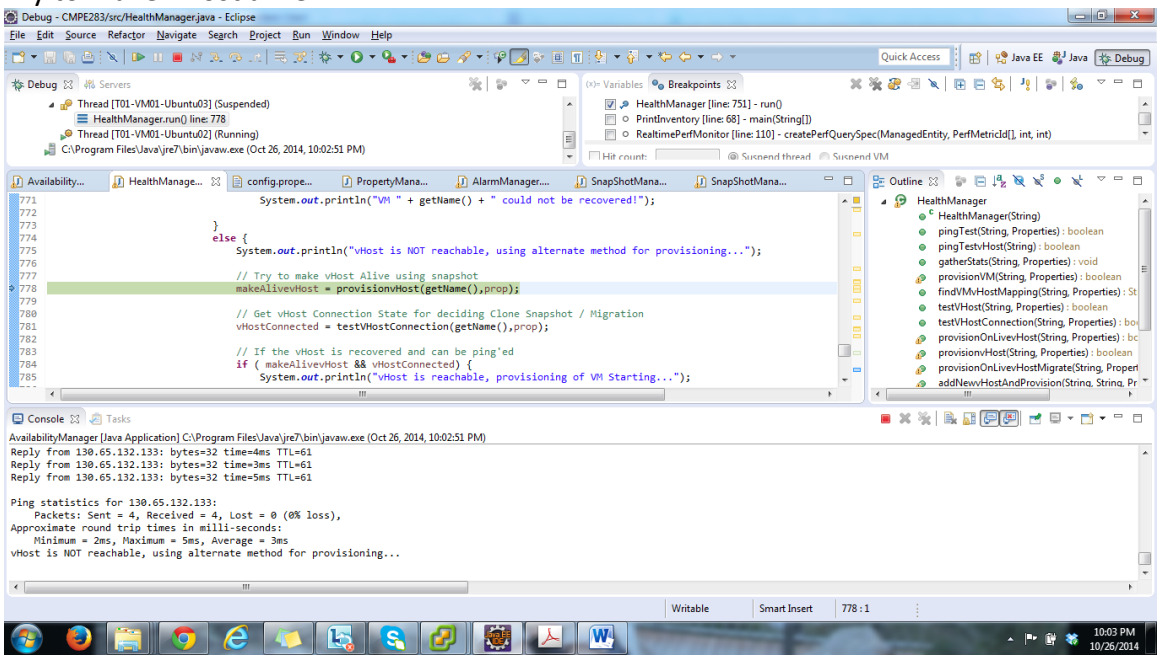
The VM is provisioned as _recovered



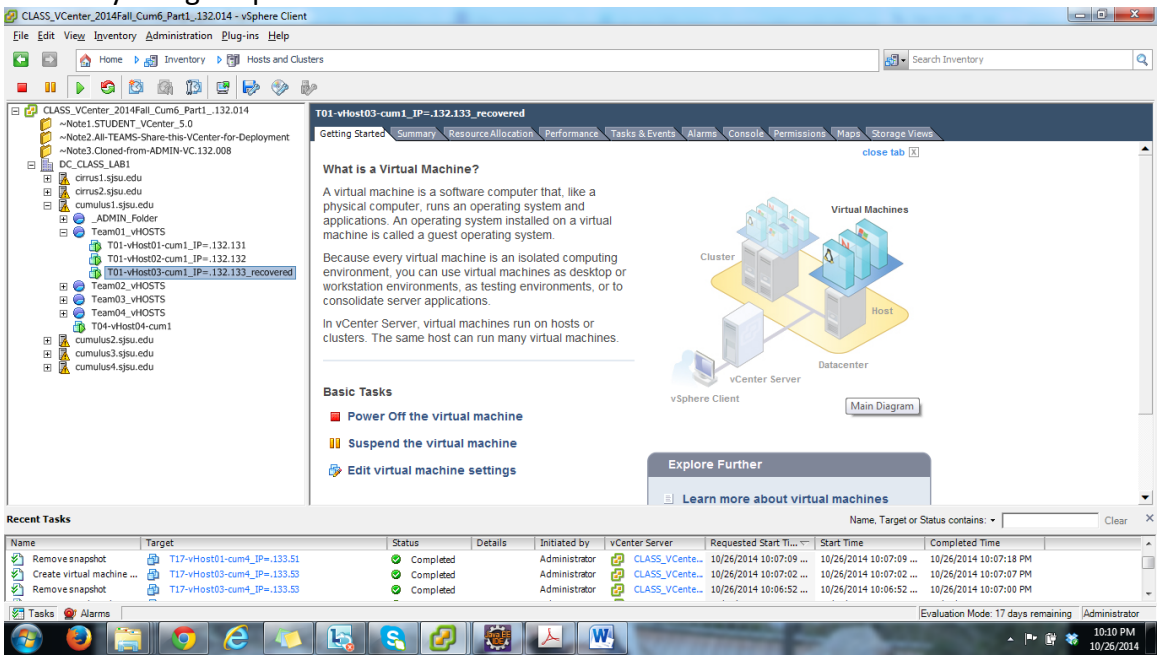
VM is down, vHost is down:



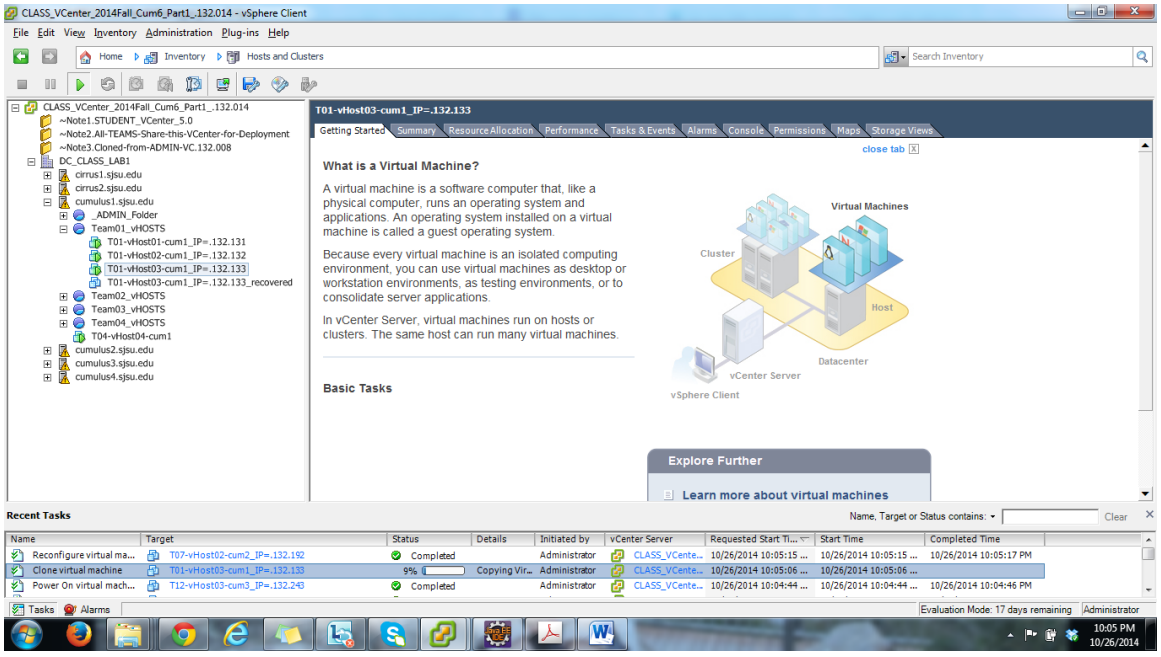
Try to make vHost alive



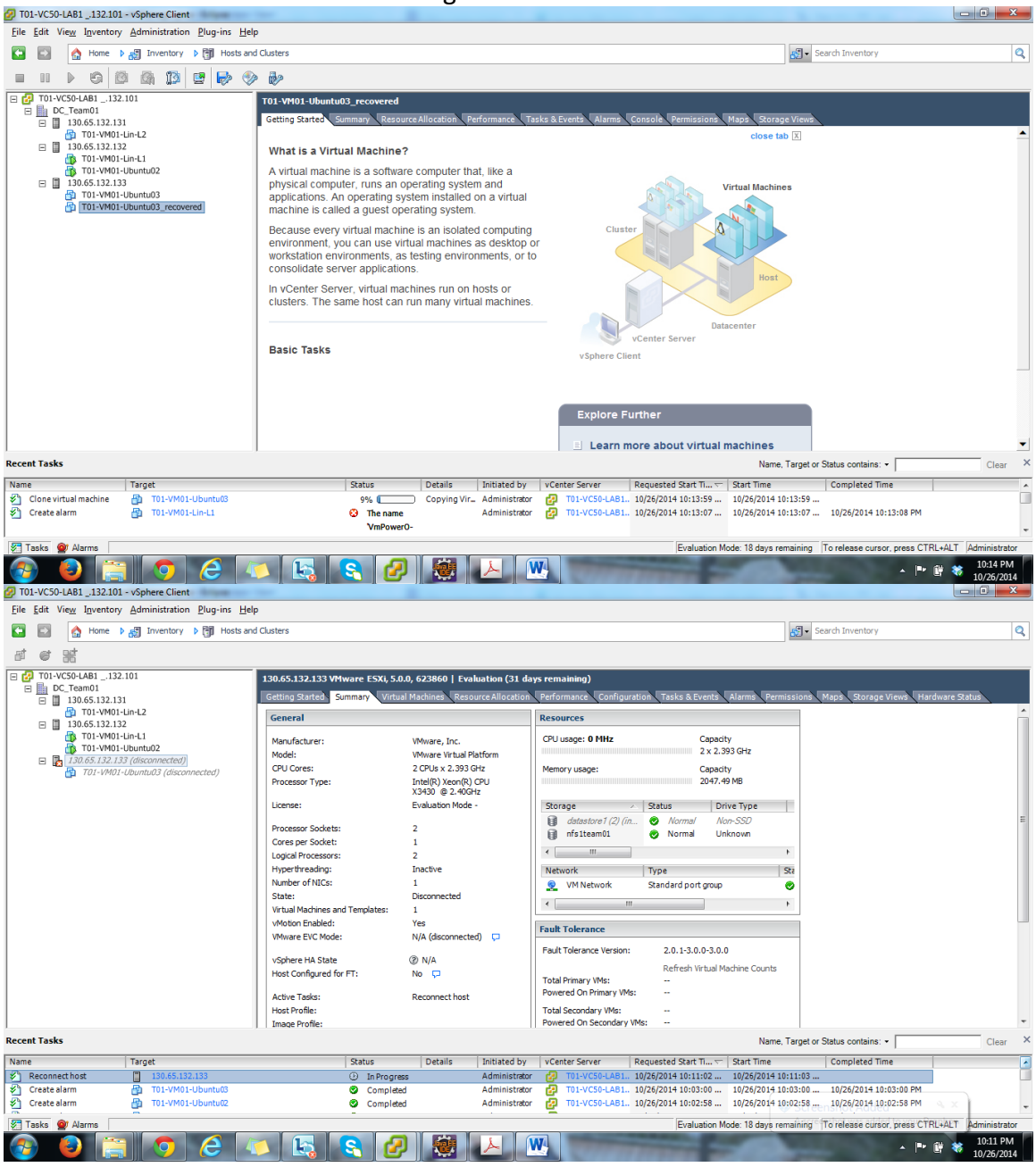
Recovery using snapshot of vHost

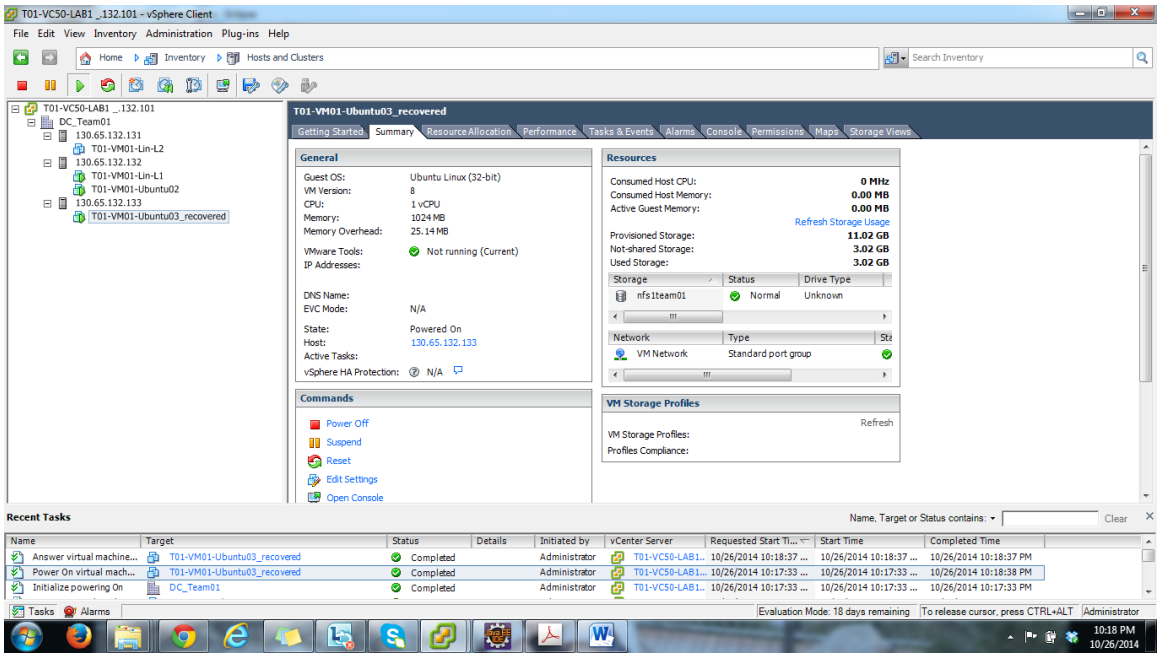


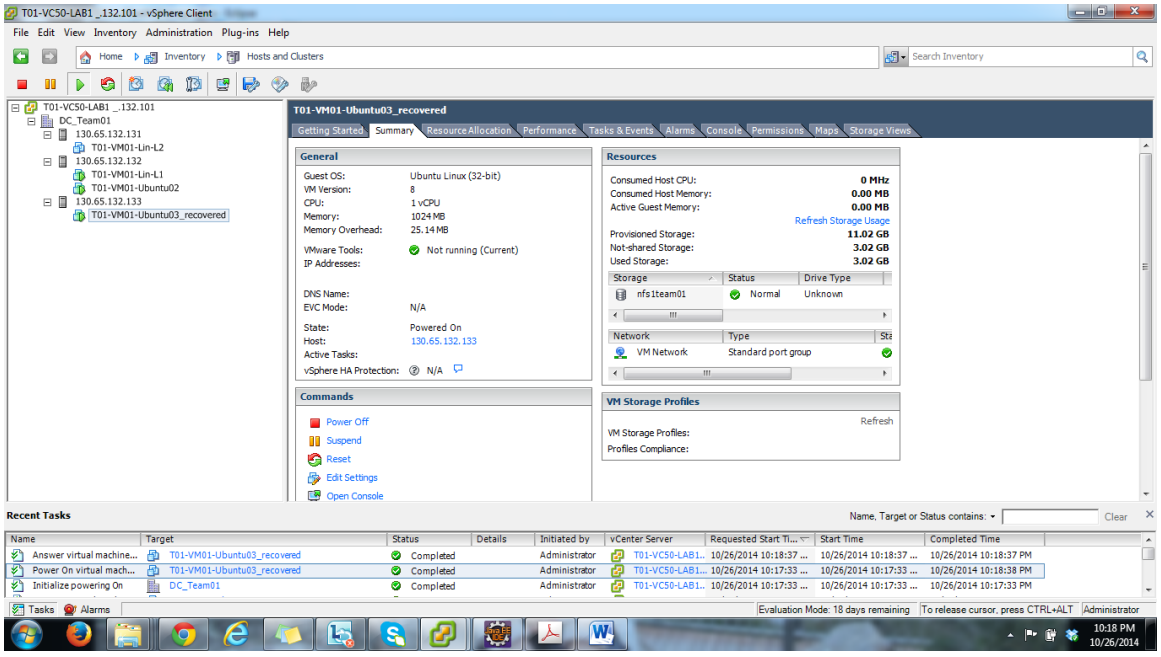
vHost recovered



VM recovered next after re-connecting




T01-VC50-LAB1 _.132.101 - vSphere Client
File Edit View Inventory Administration Plug-ins Help
Home Inventory Hosts and Clusters
Search Inventory
T01-VC50-LAB1 _.132.101
DC_Team01
130.65.132.131
T01-VM01-Lin-L2
130.65.132.132
T01-VM01-Lin-L1
T01-VM01-Ubuntu02
130.65.132.133
T01-VM01-Ubuntu03_recovered
T01-VM01-Ubuntu03_recovered
Getting Started Summary Resource Allocation Performance Tasks & Events Alarms Console Permissions Maps Storage Views
General
Guest OS: Ubuntu Linux (32-bit)
VM Version: 8
CPU: 1 vCPU
Memory: 1024 MB
Memory Overhead: 25.14 MB
VMware Tools: Not running (Current)
IP Addresses:
DNS Name:
EVC Mode: N/A
State: Powered On
Host: 130.65.132.133
Active Tasks:
vSphere HA Protection: N/A
Commands
Power Off
Suspend
Reset
Edit Settings
Open Console
Resources
Consumed Host CPU: 0 MHz
Consumed Host Memory: 0.00 MB
Active Guest Memory: 0.00 MB
Refresh Storage Usage
Provisioned Storage: 11.02 GB
Not-shared Storage: 3.02 GB
Used Storage: 3.02 GB
Storage Status Drive Type
nfs1team01 Normal Unknown
Network Type
VM Network Standard port group
VM Storage Profiles
Refresh
VM Storage Profiles:
Profiles Compliance:
Recent Tasks
Name, Target or Status contains:
Clear
Name Target Status Details Initiated by vCenter Server Requested Start Ti... Start Time Completed Time
Answer virtual machine... T01-VM01-Ubuntu03_recovered Completed Administrator T01-VC50-LAB1... 10/26/2014 10:18:37 ... 10/26/2014 10:18:37 ... 10/26/2014 10:18:37 PM
Power On virtual mach... T01-VM01-Ubuntu03_recovered Completed Administrator T01-VC50-LAB1... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 ... 10/26/2014 10:18:38 PM
Initialize powering On DC_Team01 Completed Administrator T01-VC50-LAB1... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 PM
Tasks Alarms
Evaluation Mode: 18 days remaining
To release cursor, press CTRL+ALT
Administrator
10:18 PM
10/26/2014


T01-VC50-LAB1 _.132.101 - vSphere Client
File Edit View Inventory Administration Plug-ins Help
Home Inventory Hosts and Clusters
Search Inventory
T01-VC50-LAB1 _.132.101
DC_Team01
130.65.132.131
T01-VM01-Lin-L2
130.65.132.132
T01-VM01-Lin-L1
T01-VM01-Ubuntu02
130.65.132.133
T01-VM01-Ubuntu03_recovered
T01-VM01-Ubuntu03_recovered
Getting Started Summary Resource Allocation Performance Tasks & Events Alarms Console Permissions Maps Storage Views
General
Guest OS: Ubuntu Linux (32-bit)
VM Version: 8
CPU: 1 vCPU
Memory: 1024 MB
Memory Overhead: 25.14 MB
VMware Tools: Not running (Current)
IP Addresses:
DNS Name:
EVC Mode: N/A
State: Powered On
Host: 130.65.132.133
Active Tasks:
vSphere HA Protection: N/A
Commands
Power Off
Suspend
Reset
Edit Settings
Open Console
Resources
Consumed Host CPU: 0 MHz
Consumed Host Memory: 0.00 MB
Active Guest Memory: 0.00 MB
Refresh Storage Usage
Provisioned Storage: 11.02 GB
Not-shared Storage: 3.02 GB
Used Storage: 3.02 GB
Storage Status Drive Type
nfs1team01 Normal Unknown
Network Type
VM Network Standard port group
VM Storage Profiles
Refresh
VM Storage Profiles:
Profiles Compliance:
Recent Tasks
Name, Target or Status contains:
Clear
Name Target Status Details Initiated by vCenter Server Requested Start Ti... Start Time Completed Time
Answer virtual machine... T01-VM01-Ubuntu03_recovered Completed Administrator T01-VC50-LAB1... 10/26/2014 10:18:37 ... 10/26/2014 10:18:37 ... 10/26/2014 10:18:37 PM
Power On virtual mach... T01-VM01-Ubuntu03_recovered Completed Administrator T01-VC50-LAB1... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 ... 10/26/2014 10:18:38 PM
Initialize powering On DC_Team01 Completed Administrator T01-VC50-LAB1... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 ... 10/26/2014 10:17:33 PM
Tasks Alarms
Evaluation Mode: 18 days remaining
To release cursor, press CTRL+ALT
Administrator
10:18 PM
10/26/2014

Popular Questions:

1. Briefly explain the design of your Availability Manager with the help of a class diagram. Also explain the number of threads you've used for the Availability Manager.
Answer: Explained in Design Section

2. How does your availability manager handle the scenario where-in the vHost itself is found not to be alive?
Answer: Explained in Design ->Key Workflow Section

3. In case of failure, what is a good approach during Disaster Management of Virtual Machines?

- Check the Host first, then the Virtual Machine
- Check the Virtual Machine, then the Host

Answer: The main aim of for disaster recovery is to make sure the virtual machine is running. In case the VM goes down because of failure we need to provision and recover it. In my opinion, checking the VM first and then the Host is a good approach because, consider the scenario in which vHost is running and its VMs are also running – so if we go by "Check the Host first, then the Virtual Machine" we are unnecessary pinging both Host and VM even though vHost & its VM is up. However, by using the "Check the Virtual Machine, then the Host" approach we only perform the ping operation on vHost only if VM is down, thus optimizing out calls.
Also "Check the Virtual Machine, then the Host" is more scalable since, we ping vHost only when it's required.

Conclusion:

Built a Disaster Recovery solution, that monitors VM for failure and recovers and provisions it using snapshots.

Concepts Learned:

- VMware Infrastructure APIs
- Cloning using Snapshot
- Cold Migration using Snapshot
- Alarm Management
- Snapshot Management
- Threading in Java
- Distributed Static HashMap

References:

[1] Java API: http://vijava.sourceforge.net//
[2] Tutorial at http://vijava.sourceforge.net/doc/getstarted/tutorial.htm
[3] Project Description document