

Design Specifications for “Hosting of Virtual-labs using the One-lab-per-VM model”

Suraj Samal

06 November 2013

Contents

1	Introduction	2
2	Document Revision	3
3	Basic Architecture	3
3.1	Actors	4
3.1.1	Lab Developer	4
3.1.2	Lab Administrator	5
3.1.3	Lab User	5
3.2	Entities	5
3.2.1	LabDepository	5
3.2.2	Lab	7
3.2.3	LabManager	7
3.2.4	Host	8
3.2.5	VMMManager	8
3.2.6	VM	9
3.3	Relationships	10
3.3.1	LabDepository - repository - revision	10
3.3.2	Lab - repository - revision	10
3.3.3	LabManager - host - vmmgr - vm - lab	10
3.4	Workflows	17
4	Components and Interfaces	17
4.1	Lab Manager	18
4.2	VM Manager	19
4.3	DeveloperPortal	21
4.4	DeploymentDashboard	22
4.5	LabInfoDatabase	22

5	Network Architecture	22
6	Security	24
7	Performance	24
8	Reliability and Availability Model	24
9	Backup Model	25

List of Figures

1	Overview	4
2	Relationships	10
3	Create a Lab	11
4	Update a Lab	12
5	Test a Lab	13
6	Release a Lab	14
7	Delete a Lab	15
8	Delete a Lab	16
9	LabManager to VMManager Interface	18
10	Other Interfaces	21
11	Network Architecture	23
12	Backup Model	25

1 Introduction

The document discusses the design of the overall architecture of the hosting of virtual-labs using the one-lab-per-vm model. This is as per the requirements specified in “Minutes of the 2013-07-25 Thu Expert Committee meeting evaluating VLEAD’s progress in virtual lab integration” document at **Section-4, Item-3**

VLEAD (Virtual Labs Engineering and Architecture Division) team was setup in June 2012 as a central engineering team for integrating all the virtual-labs (around 180 in number) across all disciplines and institutes onto a common data-center (currently located at IIIT Hyderabad). Currently(as of 2013-11-01) around 100 labs are version-controlled and around 50 hosted out of IIIT data-centre.

2 Document Revision

Current Revision	0.1
Revision Date	2013-11-04

3 Basic Architecture

Figure-1 represents a basic overview of the overall system, its sub-components describing all the actors, entities and they interface with each other

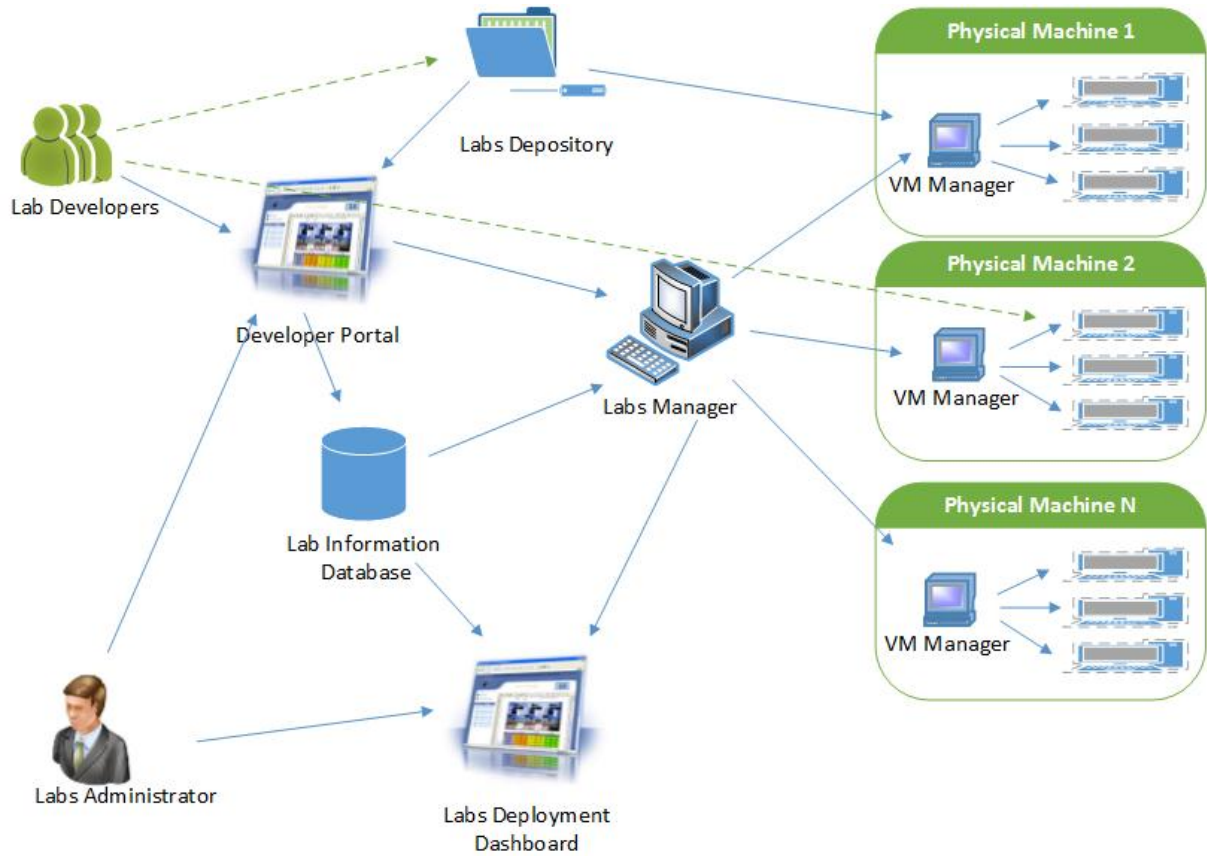


Figure 1: Overview

3.1 Actors

3.1.1 Lab Developer

An person who has agreed to use the services of VLEAD as per the **terms of association** and follows certain standard processes to maintain his/her lab during its development life-cycle. In specific, the roles are as follows:

- Checkin the lab contents (repositories containing sources,dependencies,scripts and other files) into a lab-depository.
- Keep updating the lab-depository with newer revisons of lab contents.
- Instantiate a test lab-instance for testing and debugging issues.
- Instantiate a live lab-instance.
- View live lab-instance statistics.

3.1.2 Lab Administrator

An actor who is responsible for administering all the hosted labs. In specific, the roles are as follows:

- Allocate a unique labid and a depository(collection of repositories) to a lab
- Allocation of resources(physical machines,ip address pools, vmid pools) to the lab-manager and vmmanager

3.1.3 Lab User

These are end-users who use the virtual-labs and run its experiments.

3.2 Entities

3.2.1 LabDepository

All labs are allocated a unique-id and a lab-depository by the labs administrator. A lab-depository represents a collection of various repositories associated with a lab. Each depository has a **metadata** repository which is automatically regenerated when any of the repositories are modified.

lab-depository - An **Object** describing the property of all repositories of a particular lab

labid - Unique identifier of the lab

labinfo - **Object** describing basic properties of a lab

labinst - One of the defined **enumerations** (IITB, IITK, IIITH ,...)

labdisc - One of the defined **enumerations** (chemical, mechanical)

labos - **Object** describing a particular operating-system version

osname - Name of the operating system

osversion - Specific version of the operating system

repos - Collection of repositories

metadata - A structured **object** representation of depository contents describing the number of repos present, actual repos present, their type . This repository is regenerated everytime the lab-developer makes a commit to other repositories.

numrepos - Total number of repositories present

reposit1 - Identification of each repository

reposit2 -

reposit2 - .

reposit2 - .

repositN -

repo1 - A repository **object** which refers to a svn, git or bsr repository

reposit - Identification text that can be used to checkout the repository.
(Eg: cse01, mech09)

reponame - Display text (Eg: Frontend, Backend, UI etc)

reposit - One of the supported **enumerated** types - (git, svn, bsr)

revsnum - Number of revisions of the repository (Eg: 20)

rev - **Object** defining a particular repository revision

revno - Unique revision number generated by the repository tool. (
Eg: 10)

date - Date/Time the revision was checked into the repository. (Eg:
2013-11-10 16:30)

user - Text representing user who checked the revision. (Eg: ramakrishna)

diskspace - Approximate disk-space required. (Eg: 30G)

ram - Approximate memory required. (Eg: 256M)

staticdeps - An **object** describing a list of packages the lab depends on. (Eg: apache2, opencv)

dep1

dep2 . . .

depn

runtimedeps - An **object** describing a list of services to be enabled/started. Services may mean standard packages (eg. apache2) or other custom made scripts (Eg: backup) to be configured during installation of the lab.

dep1

dep2 . . .

depn

size - Number representing the size of the particular repository revision (**Optional**)

repo2 -
repo2 - .
repo2 - .
repo2 - .
repoN -

3.2.2 Lab

An instance of a lab (inactive) which refers to a complete set of properties that can be used to instantiate a particular lab revision. All these properties can be loaded directly from the lab-depository by using its unique labid, unique repoid and a unique revision no.

lab - **Object** describing an lab

labid - Unique id to identify the lab from others

labinfo - **Object** describing basic properties of a lab (described earlier)

repo - **Object** describing a particular repository of a lab (described earlier)

rev - **Object** describing a particular revision of a particular repository of a lab (described earlier)

3.2.3 LabManager

An entity that monitors a set of physical hosts, accepts requests for creation, modification and deletion of lab-instances and sends request to appropriate vm-manager for life-cycle management of labinstances

labmanager - An entity responsible for managing the various vm-managers

labmanagerid - Unique id to describe a labmanager

hosts - **Object** representation of a list of physical-hosts

host1 - **Object** representation of a physical host (described later) . . .

host2 - . . .

host3 -

runtime runtime characteristics of the labmanager

starttime - timestamp the labmanager was instantiated

3.2.4 Host

A physical server that hosts the labs. It constitutes a vm-manager that is responsible for managing all the vms on the host.

Host - Entity representing a physical host

hostname - Common name of the host

vmmgr - **Object** representation of the vm-manager (described later) managing the host

hostid - Unique-id representation of the host

hostip - IPaddress of the physical host

resource - **Object** representation of resources of the physical host

diskspace - (Eg. 2000GB)

mem - (Eg. 64GB)

cpu - (Eg. 2)

runtime - Runtime properties of the host

status - one of running, stopped, shutoff

starttime - timestamp the host was started

useddiskspace - (Eg. 100GB)

usedmem - (Eg. 20GB)

usedcpu - (Eg. 1)

3.2.5 VMManager

An entity that is responsible for managing virtual machines(vms) on a particular host

vmmgr - Entity describing an instance of a vm-manager residing on a physical machine

vmmgrid - Unique id to represent the vm-manager

vms - List of vm objects

vm1 - **Object** representation of a vm (described later)

vm2 -

vmN -

resources - **Object** representation of resources

vmids - List of available vmids

vmid1 -

vmid2 - ..

vmidn -

- ips** - List of available ips
 - ip1** -
 - ip2** - . . .
 - ipn** -
- runtime** - Runtime properties
 - status** - up, down, stopped
 - starttime** - start timestamp

3.2.6 VM

A VM is a running instance of a lab.

vm - An active instance of a lab that runs on a specified host

- guid** - Global Universal id of the vm generated to identify the VM universally (never ever repeated in time). It can be used to refer to a particular VM that exists or ever existed before.

- vmid** - Unique identification of a vm among its current running VMs. This is allocated from a defined pool of ids when the vm is created and re-sent to the pool when the vm gets destructed.

- vmname** - Common name to identify the VM instance.

- vmos** - Operating system **object** of the running vm.

- osname** - Name of the operating system

- osversion** - Particular version of the operating system

- lab** - A particular instance of a lab associated with a vm

- runtime** - **Object** describing run-time properties of the vm

- state** - running, stopped, suspended, archived

- createddate** - Creation time-stamp of the VM

- modifieddate** - Modification time-stamp of the VM

- lastbackedup** - Timestamp when the vm was last backedup

- stats** - **Object** describing stats of a vm

- userstats** - User-level statistics of the vm

- userinfo** -

- perfstats** - Performance statistics

- cpuinfo** -

- meminfo** -

- netinfo** -

3.3 Relationships

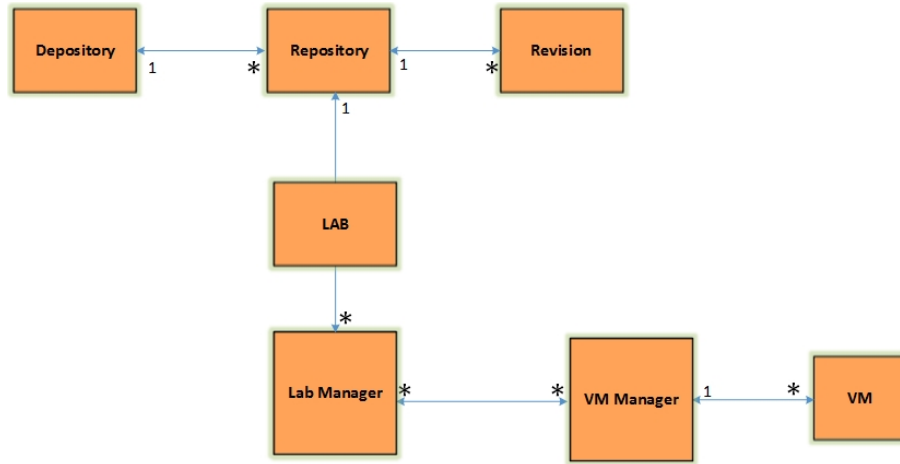


Figure 2: Relationships

3.3.1 LabDepository - repository - revision

[Lab-Depository] 1 ——— * [repo] 1 ——— * [rev]

3.3.2 Lab - repository - revision

[Lab] 1 ——— 1 [repo] 1 ——— 1 [rev]

3.3.3 LabManager - host - vmmgr - vm - lab

[Labmanager] * ——— * [host] 1 ——— 1 [vmmgr] 1 ——— * [vm] 1 ——— 1 [lab]

- Create a Lab

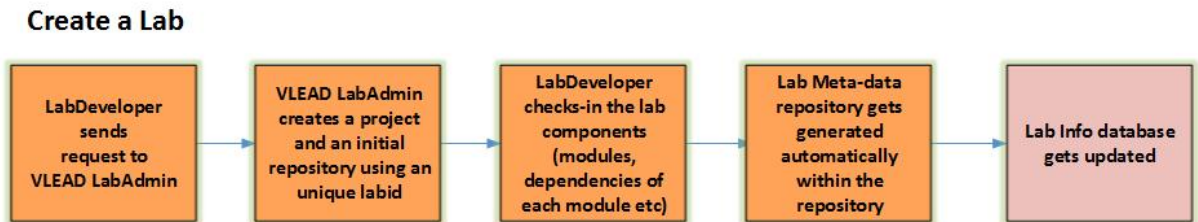


Figure 3: Create a Lab

- Update a Lab

Update a Lab

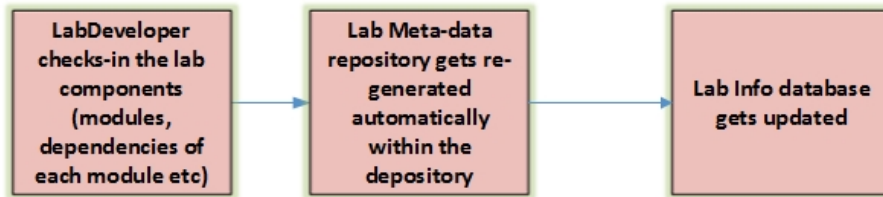


Figure 4: Update a Lab

- Test a Lab

Test a Lab

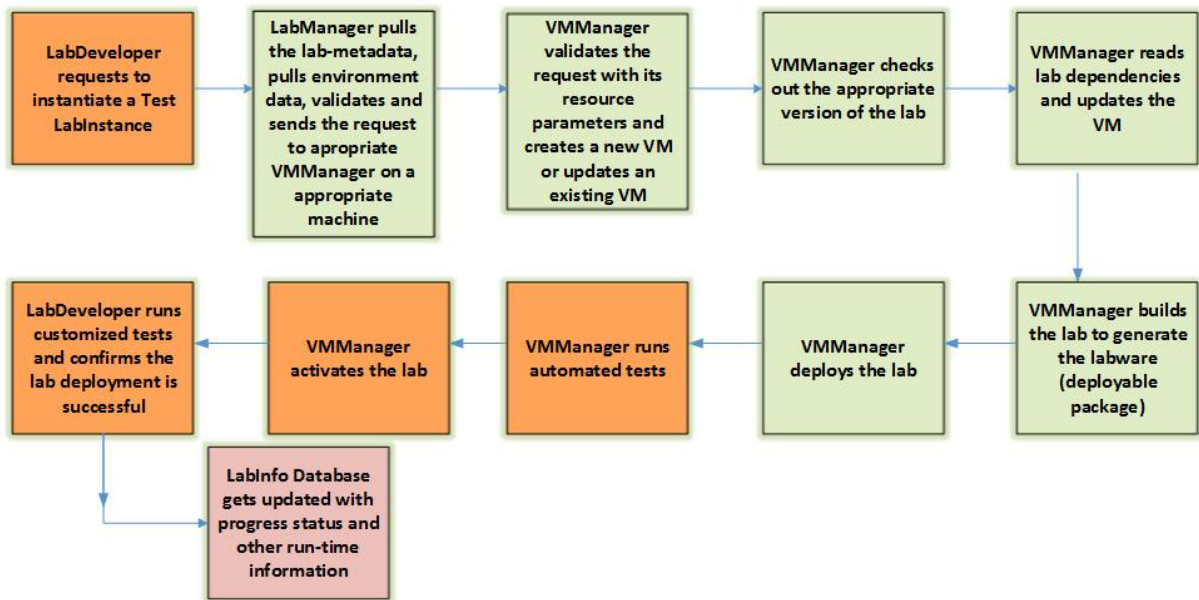


Figure 5: Test a Lab

- Release a Lab

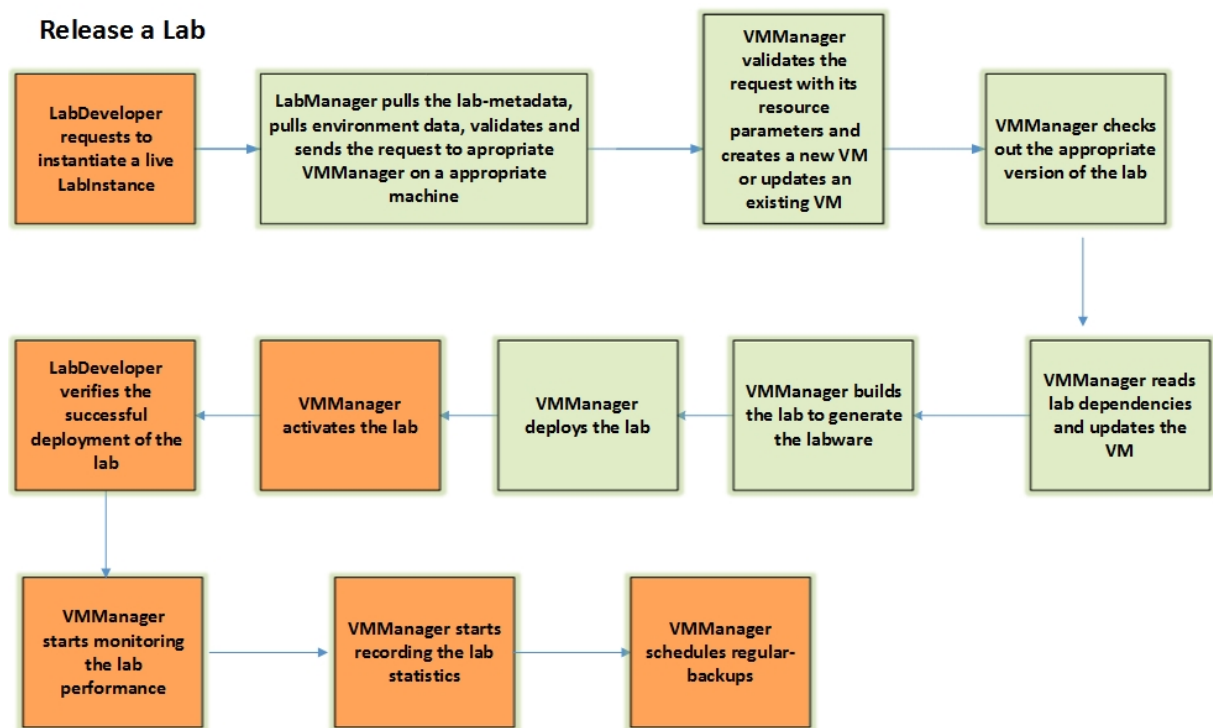


Figure 6: Release a Lab

- Delete a Lab

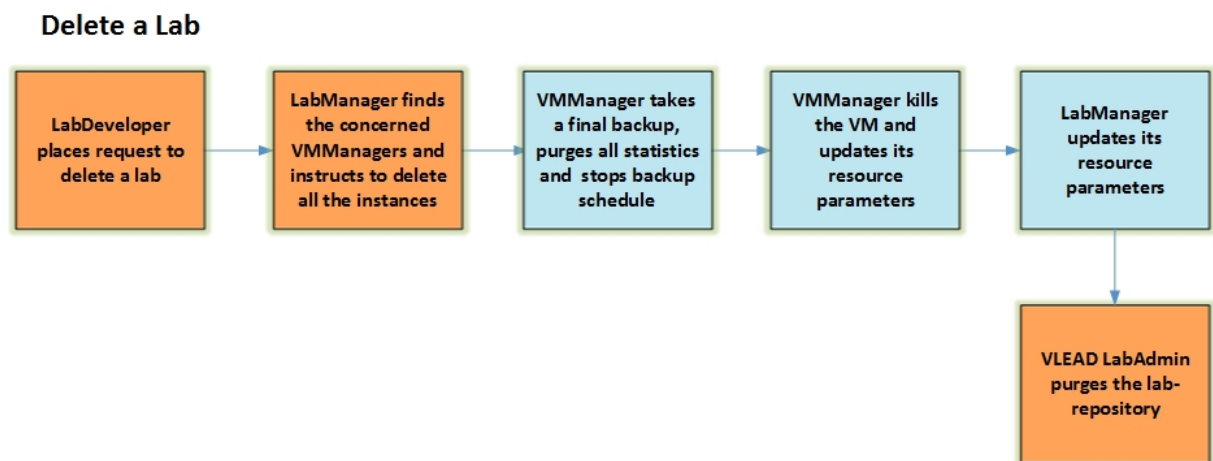


Figure 7: Delete a Lab

- Fetch Lab-Statistics

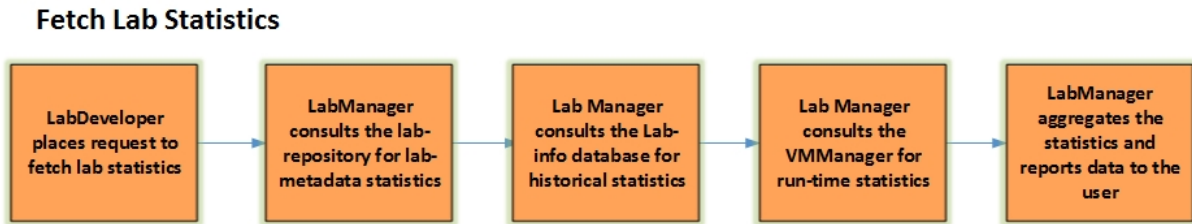


Figure 8: Delete a Lab

- Update Lab Information
 - Host Parameters
 - Network Parameters
 - VM Manager Information
 - VM Parameters
 - Backup Schedule
 - Lab Status (Deactivate a Lab)
 - Purge VM Logs
- Take a Lab snapshot (backup)
- Restore a Lab from its snapshot (backup)
- Monitor VM Statistics
- View a Lab
- Log Lab History
- AutoPurge Lab History

3.4 Workflows

4 Components and Interfaces

- Following are the components that need to be designed for the proposed architecture:

4.1 Lab Manager

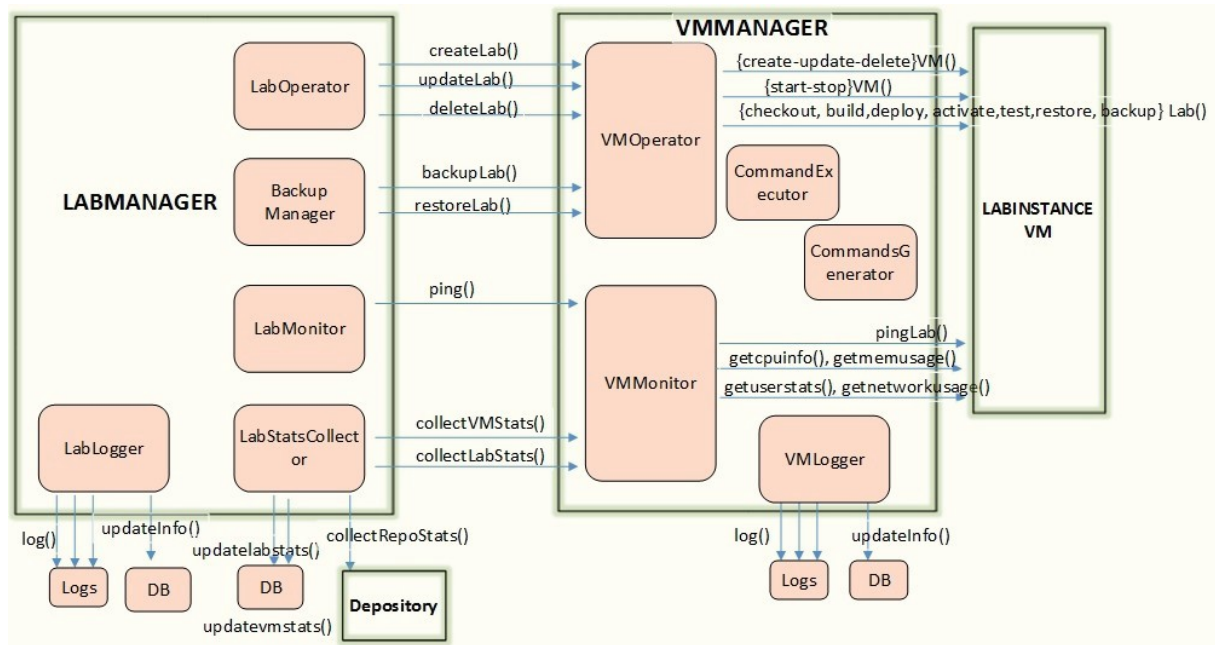


Figure 9: LabManager to VMManager Interface

LabOperator Manages basic operations for the life-cycle management of lab

- createLab(vmmanager, lab)
- updateLab(vmmanager, lab)
- deleteLab(vmmanager, lab)
- updatereources() - Adds or removes resources information (Eg. vmmanager, hosts)

LabMonitor Regularly monitors the status of labs and vms

- ping(vmmanager, lab)

LabLogger Logs status and history information to the lab-info database

- loginfo()
- logwarn()
- logerror()
- purgelogs()

LabStatsCollector • collectvmstats(vmmanager)

- collectlabstats(vmmanager, lab)
- collectrepostats(lab)
- updatevmstatstoDB()
- updatelabstatstoDB()

BackupManager • backup(vmmanager, lab)

- restore(vmmanager, lab)
- schedule(lab)

4.2 VM Manager

VMOperator Manages basic operations for life-cycle of a vm and a lab

- createvm(lab)
- updatevm(vmid)
- deletevm(vmid)
- stopvm(vmid)
- startvm(vmid)
- updateresources(host)
- checkoutlab(vmid, lab)
- buildlab(vmid, lab)
- deploylab(vmid, lab)
- activatelab(vmid, lab)
- testlab(vmid, lab)
- restorelab(lab, snapshot)
- backuplab (lab, snapshot)
- updateinfoDB()

VMMonitor • pinglab(lab)

- getcpuinfo(vmid)
- getmemusage(vmid)
- getnetworkusage(vmid)
- getuserstats(vmid)
- getcpuinfo(host)
- getmemusage(host)
- getnetworkusage(host)

VMLogger • loginfo()

- logwarn()
- logerror()
- purgelogs()

CommandsGenerator A component that generates the configuration commands based on operation specified by the VMOperator

- generateconfig(configid)

CommandExecutor A component that runs the configuration commands generated earlier by the CommandsGenerator

- applyconfig(configid)

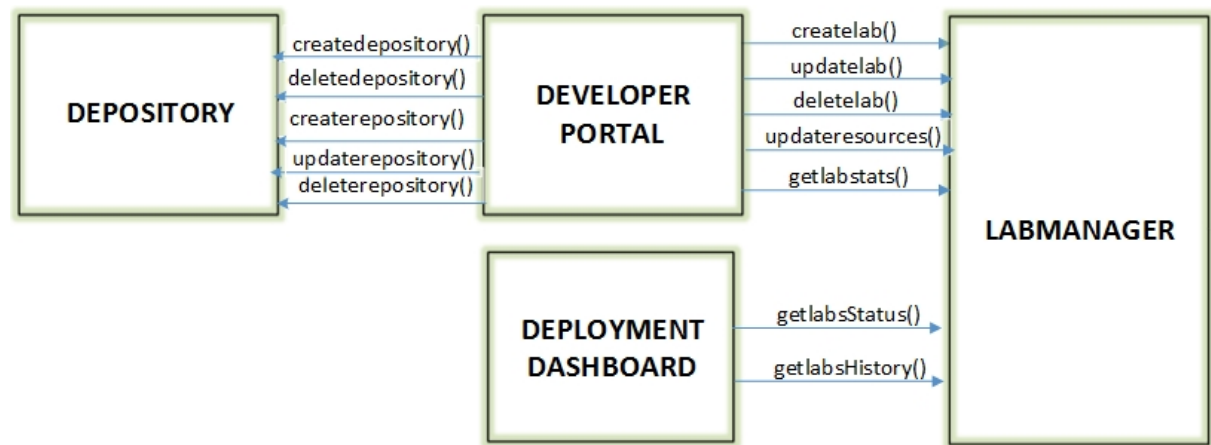


Figure 10: Other Interfaces

4.3 DeveloperPortal

- createdepository(lab)
- createrepository(labdepository, lab)
- updaterepository(labdepository, lab)
- deleterepostitory(labdepository, l
- deletedepository(lab)
- sendrequest(labmanager, lab, operation) - Operation could be one of create/update/test/release a lab or getlabstats

- `updateresources(labmanager)` - Information about physical-hosts, network parameters etc

4.4 DeploymentDashboard

- `getlabsStatus(labmanager)`
- `getlabsHistory(labInfoDb)`

4.5 LabInfoDatabase

- `VMHistory`
- `LabHistory`
- `VMMManagerHistory`
- `LabManagerHistory`

5 Network Architecture

Presented below is a network architecture diagram of the proposed solution:

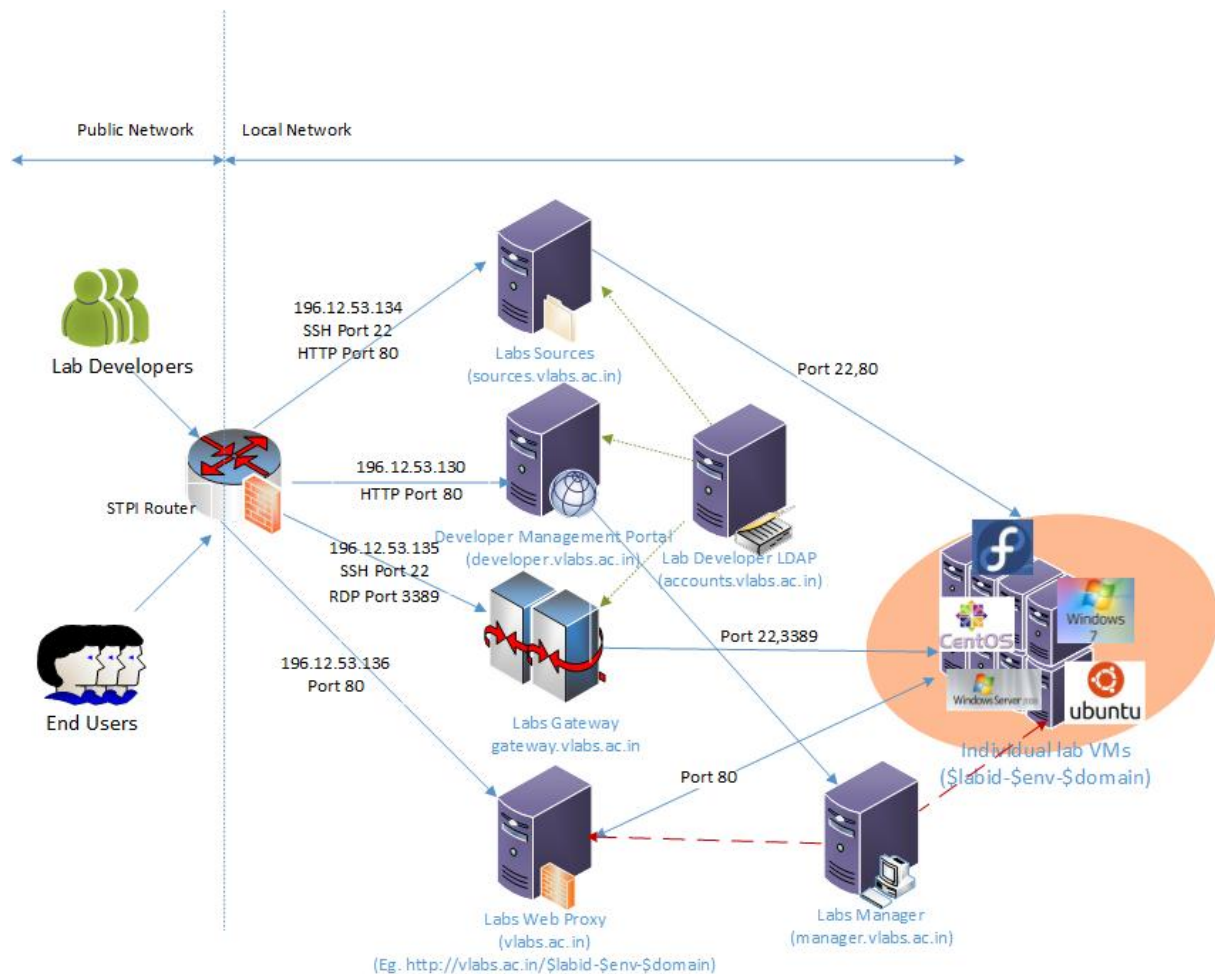


Figure 11: Network Architecture

End-users They access the labs through a public-url (eg: <http://vlabs.ac.in/labid-test>). Internally, the requests are proxied to the actual labvms through a web-proxy over port 80

Lab-developers

- Access to the management-portal is through a public url which gets mapped to the portal over port 80.
- Access to the lab-vms is by a using a forwarding-gateway(eg: gateway.virtual-labs.ac.in) that tunnels SSH(Port 22) requests to their lab-vms for linux VMs and Remote-Desktop(Port 3389) requests for windows VMs.
- Access to the lab-depository is by using a direct SSH(Port 22) tunnel to the version-control server.

6 Security

- Firewall rules are configured at the router-interface for translating public requests to private requests. These rules allow only specific requests on specific ports (as described above) and block all other requests.
- Labs are accessed by users through a web-proxy that logically isolates the actual lab-instances from public world. In any case, the security of only the web-proxy host is compromised. The web-proxy can be configured for additional security and monitored for user statistics.
- Labs are accessed by lab-developers using a gateway that isolates the actual lab-vms from the public world. So, in any case the security of only the gateway is compromised.
- All SSH access requests are authenticated by the lab-developer LDAP which would be administered by the VLEAD Admin team.

7 Performance

- The lab-developer is responsible for performance of the lab and hence suggest VLEAD admin the appropriate required memory and cpu parameters for creating a VM for the lab.
- Overall network-performance of all labs would be shared across a 4Mbps:4Mbps(upward:downward) network link.

8 Reliability and Availability Model

- Redundant network-links would to be configured to make sure labs are available 24*7

9 Backup Model

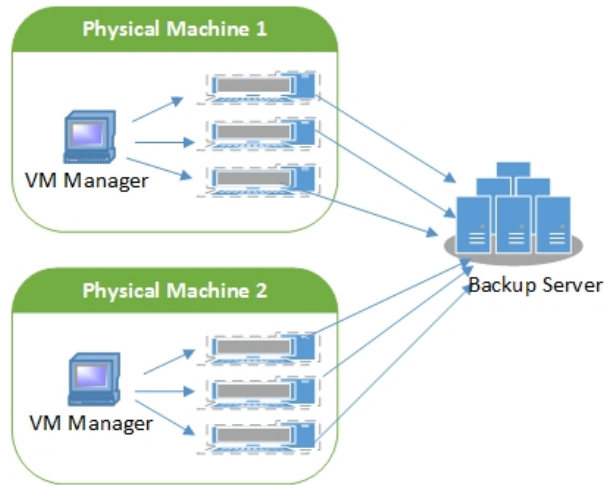


Figure 12: Backup Model

- All labs would be backedup as per the following schedule. For full backups, the whole image of the VM would be backedup. For incremental backups, the lab-developer has to specify the appropriate files to be backed-up.

Type of Backup	Schedule
Auto Full(VM level)	Weekly
Auto Incremental(Filesystem level)	Daily
Full/Incremental	OnRequest