DEVELOPMENT OF A TOOL FOR CLOUD DISASTER RECOVERY BASED ON LOG ANALYSIS

A thesis

Submitted in partial fulfillment of the requirement for the Degree of

Master of Computer Science and Engineering

of

Jadavpur University

By

Saptarshi Bhowmik

Registration No.: 129011of 2014-15

Examination Roll No.: M4CSE1621

Under the Guidance of

Prof. Chandan Mazumdar

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2016

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled "Development Of A Tool For Cloud Disaster Recovery Based On Log Analysis" has been carried out by Saptarshi Bhowmik (University Registration No.: 129011 of 2014-15, Examination Roll No.: M4CSE1621) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Computer Science and Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

……………………………………………………

Prof. Chandan Mazumdar (Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-32

Countersigned

…………………………………………………….

Prof. Debesh Kumar Das

Head, Department of Computer Science and Engineering,

Jadavpur University, Kolkata-32.

…………………………………………………….

Prof. Sivaji Bandyopadhyay

Dean, Faculty of Engineering and Technology,

Jadavpur University, Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis entitled "Development Of A Tool For Cloud Disaster Recovery Based On Log Analysis" is a bona-fide record of work carried out by Saptarshi Bhowmik in partial fulfillment of the requirements for the award of the degree of Master of Computer Science and Engineering in the Department of Computer Science and Engineering, Jadavpur University during the period of June 2015 to May 2016. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

…………………………………………………………………………………..

Signature of Examiner 1

Date:

…………………………………………………………………………………..

Signature of Examiner 2

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "Development Of A Tool For Cloud Disaster Recovery Based On Log Analysis" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Science & Engineering.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Saptarshi Bhowmik

Registration No: 129011 of 2014-15

Exam Roll No.: M4CSE1621

Thesis Title: Development Of A Tool For Cloud Disaster Recovery Based On Log Analysis

…..……………………………………………..

Signature with Date

Acknowledgement

With my sincere respect and gratitude, I would like to thank my thesis guide Prof. Chandan Mazumdar for his continuous support for this thesis work, for his patience, motivation and enthusiasm. His guidance helped me a lot throughout the duration of the work. His valuable suggestions inspired me a lot. I feel deeply honored that I got the opportunity to work under his guidance.

I wish to thank Mr Prakash Gowda,Sr. Manager, VMware Software India ltd, Bangalore for providing me all the facilities and support required in this project.

I would also wish to thank Prof. Prof. Debesh Kumar Das, Head of the Department of Computer Science & Engineering, Jadavpur University for providing me all the facilities and for his support to the activities of this project.

I would also wish to thank Prof. Prof. Sivaji Bandyopadhyay, Dean, Faculty of Engineering and Technology, for providing me all the facilities and for his support to the activities of this project.

I would like to thank CDCJU Lab and its' members, for providing me the resources required for the project work and sharing their knowledge and experience with me.

Last, but not the least, I would like to thank my class mates of Master of Computer Science & Engineering batch of 2014-2015, for their co-operation and support. Their wealth of experience has been a source of strength for me throughout the duration of my work.

…………………………………………

Saptarshi Bhowmik

Exam Roll No: M4CSE1621

Class Roll No: 001410502030

Registration No: 129011 of 2014-15

Master of Computer Science and Engineering

Jadavpur University, Kolkata

Table of Contents

List of Figures

List of Table

CHAPTER 1

INTRODUCTION

Cloud computing is a state-of-art computing paradigm, where a large pool of resources are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage.

With the advent of this technology, computational cost, application hosting, content storage and delivery is abated by a huge proportion. Cloud computing has the potential to transform a data center from a capital-intensive set up to a variable priced environment. The main motivation behind cloud computing is “reusability of IT capabilities”. The difference that cloud computing brings compared to traditional concepts of “grid computing”, “distributed computing”, “utility computing”, or “autonomic computing” is to broaden horizons across organizational boundaries.

The following definition of cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST):[7]

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.

Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.

Cloud can be supported by a cloud provider that sets up a platform that includes the OS, Apache, a MySQL database, Perl, Python, and PHP with the ability to scale automatically adjusting to changing workloads. Cloud computing is the ability to use applications on the Internet that store and protect data while providing a service

— anything including email, sales force automation and tax preparation.

It can be using a storage cloud to hold application, business, and personal data. And it can be the ability to use a handful of Web services to integrate photos, maps, and GPS information to create a mashup in customer Web browsers. Mainly cloud computing increases the velocity of deployment of application in the world of Information Technology. Cloud computing uses information technology as a service over the network. We define it as services that are encapsulated, have an API, and are available over the network. The infrastructure in a cloud computing is reprogrammable, that is changeable according to the needs of the customer.

Previously the developers needed to write the programs based on the environment provided to him. But with the advent of cloud computing a developer can manipulate the environment in which they are deploying their application or running their programs, with this advantage a developer can easily integrate their programs.

Hadoop, an open-source MapReduce implementation, can be used in a wide range of contexts in which a problem and its data can be refactored so that many parts of it can execute in parallel. When The New York Times wished to convert 11 million articles and images in its archive to PDF format, their internal IT organization said that it would take seven weeks. In the meantime, one developer using 100 Amazon EC2 simple Web service interface instances running hadoop completed the job in 24 hours for less than $300.Developers can create Hadoop platform by layering hadoop on an instance of open source operating system such as OpenSolaris

1.1 SERVICE CLASSIFICATION

The cloud computing provides a large number of services, but three of the main services which are provided by the cloud service provider is

1. Software-as-a-Service,
2. Platform-as-a-Service,
3. Infrastructure-as-a-Service

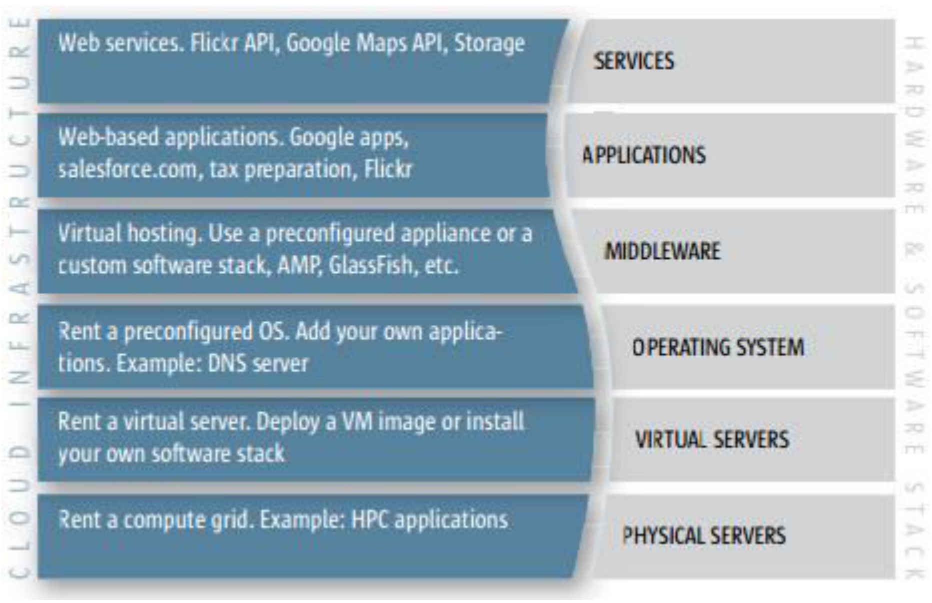

CLOUD INFRASTRUCTURE
Web services. Flickr API, Google Maps API, Storage
SERVICES
Web-based applications. Google apps, salesforce.com, tax preparation, Flickr
APPLICATIONS
Virtual hosting. Use a preconfigured appliance or a custom software stack, AMP, GlassFish, etc.
MIDDLEWARE
Rent a preconfigured OS. Add your own applications. Example: DNS server
OPERATING SYSTEM
Rent a virtual server. Deploy a VM image or install your own software stack
VIRTUAL SERVERS
Rent a compute grid. Example: HPC applications
PHYSICAL SERVERS
HARDWARE & SOFTWARE STACK

Figure 1 Cloud Computing provides anything from hardware resources to software API.

In the above figure we can see that on the top of the stack there are API's such as Google maps API, Flickr API etc., so we can see that a cloud provides an API as part of the services. Just below it there is the application which is provided as a part of the service. It can provide application servers such a Glassfish and web servers such a Tomcat as a service, these are referred to as a middleware in the above figure. It can also provide operating system, or the core hardware resources such a memory, processing power as a service.

Cloud Computing abolished the need for having a dedicated resources, it helps in end to end outsourcing from hardware to software, you name them, it gets it.

The three core service models are described below:-

1.1.1 SOFTWARE-AS-A-SERVICE:

With the advent of internet and smartphones the number of users of a application and enterprise software increased. To make the software affable, cloud plays a role that is more than that of an ancillary. It gives a complete application as a service. A single instance of a software runs on a cloud and it serves multiple end users and customer organizations.

An example of SaaS is salesforce.com, though many other examples have come to market, including the Google Apps offering of basic business services including email and word processing.

1.1.2 PLATFORM-AS-A-SERVICE:-

Platform as a service encapsulates a layer of software and provides it as a service that can be used to build higher-level services. There are at least two perspectives on PaaS depending on the perspective of the producer or consumer of the services:

1. Someone producing PaaS might produce a platform by integrating an OS, middleware, application software, and even a development environment that is then provided to a customer as a service.
   For example, someone developing a PaaS offering might base it on a set of SuNxVM hypervisor virtual machines that include a NetBean integrated development environment, a Sun GlassFish Web stack and support for additional programming languages such as Perl or Ruby.

2. Someone using PaaS would see an encapsulated service that is presented to them through an API. The customer interacts with the platform through the API, and the platform does what is necessary to manage and scale itself to provide a given level of service. Virtual appliances can be classified as instances of PaaS. A content switch appliance, for example, would have all of its component software hidden from the customer, and only an API or GUI for configuring and deploying the service provided to them. PaaS offerings can provide for every phase of software development and testing, or they can be specialized around a particular area such as content management. Commercial examples of PaaS include the Google Apps Engine

1.1.3 INFRASTRUCTURE-AS-A-SERVICE:-

Infrastructure as a service delivers basic storage and compute capabilities as standardized services over the network. Servers, storage systems, switches, routers, and other systems are pooled and made available to handle workloads that range from application components to high-performance computing applications. A third party provider hosts the infrastructure and makes them available to their customers. The providers are responsible for maintenance, backup and other cloud operations such as migration, synchronization and etc. The resources provided by an IaaS are highly scalable. System level security related issues are also taken care of by the providers.
Some of the prominent IaaS providers are Amazon and VMware through multiple number of their products such as Amazon EC2, which is provided by Amazon and VMware ESXi which is provided by VMware.
IAAS is suited for places where there is an in the experiments or change unexpectedly. It does dynamic scaling, desktop virtualization and policy based servicing. Customers pay according to their respective usage depending on whether the tariff is charged on a daily basis, monthly basis or annual basis. This reduces the capital expense of customers, because now rather than buying the entire hardware, customer can use the hardware provided and compatibly scale it up if required without any issue

However the problem here is, if the IaaS providers are down the customers may be affected even though techniques such as migration help out a lot. Also, since the IaaS service providers have access to the infrastructure, the customers may have limited access to the hardware resources. In majority of the cases the customers are not given administrative rights.

Infrastructure as a Service sits at the base of all the services provided. An example goes like this, suppose there is a software company which wants to make a software and deploy it, It can cost effectively do that by creating the software and deploying it on the infrastructure provided by the Infrastructure as a Service provider, then testing it, and after all these operations are completed, it can remove the software from the infrastructure and freeing the infrastructure.

The below is an image describing all the service models and specifically showing which part is provided to the customers and which part is not.

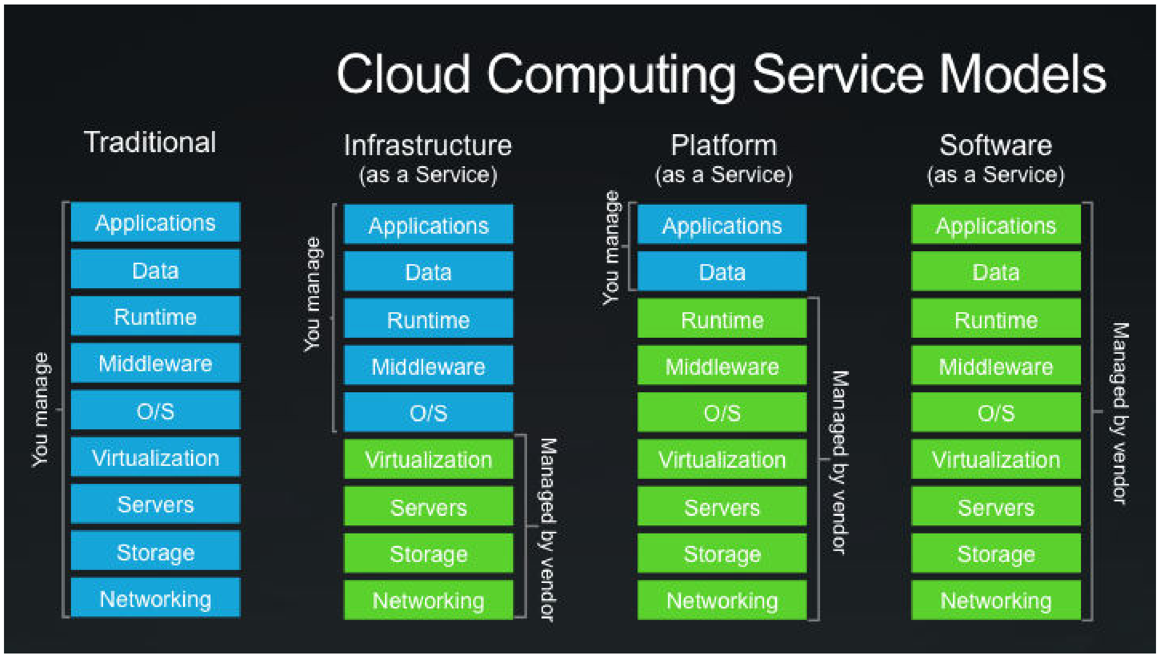


Figure 2 showing the various resources managed by cloud in the three service models

[http://blog.toddysm.com/2013/01/cloud-computing-service-models.html]

Table 1 Cloud service models comparison

Cloud Service Model	Level of Control	Functionality available to consumer	Consumer Activities	Provider Activities
Software-as-a-Service	Usage related access	Front end user Interface	uses and configures cloud service	implements, manages, and maintains cloud service monitors usage by cloud consumers
Platform-as-a-Service	Limited administrative	Moderate level of administrative control over IT resources relevant to cloud consumer's usage of platform	develops, tests, deploys, and manages cloud services and cloud-based solutions	pre-configures platform and provisions underlying infrastructure, middleware, and other needed IT resources, as necessary monitors usage by cloud consumers
Infrastructure-as-a-Service	Full administrative	Full access to virtualized infrastructure-related IT resources and, possibly, to underlying physical IT resources	sets up and configures bare infrastructure, installs, manages, and monitors any needed software	provisions and manages the physical processing, storage, networking, and hosting required monitors usage by cloud consumers

1.2 CLOUD DEPLOYMENT MODELS

There are four types of cloud depending on the type of deployment.

The following are the four types:-

1. PRIVATE CLOUD
2. PRIVATE CLOUD
3. HYBRID CLOUD
4. COMMUNITY CLOUD

We will be describing each of them shortly but before that an image shows how each of them are related

1.2.1 PUBLIC CLOUD

A PUBLIC CLOUD is a cloud model which is deployed in such a way that it is accessible publicly, but the environment is owned by a third-party cloud provider. Usually the IT resources provisioned on public clouds via the previously described cloud delivery models.

The IT resources are generally offered to consumers at a price.

The price paid by the consumers is far less than the cost they might have incurred if they build their own setup

The cloud provider is responsible for the creation and on-going maintenance of the public cloud and its IT resources.

1.2.2 PRIVATE CLOUD

Owned by a single organization, Private clouds allow an organization to use cloud computing technology for accessing the IT resources by different parts, locations, or departments of the organization. When a private cloud exists as a controlled environment, the problems described in the Risks and Challenges section do not tend to apply.

The use of a private cloud can change how organizational and trust boundaries are defined and applied. The actual administration of a private cloud environment may be carried out by internal or outsourced staff.

A private cloud is hosted in the data center of a company and provides its services only to users inside that company or its partners. A private cloud provides more security than public clouds, and cost saving in case it utilizes otherwise unused capacities in an already existing data center.

The only big advantage that private cloud has over public cloud is that of data security and privacy.
The major drawback of private cloud is its higher cost. When comparisons are made with public cloud; the cost of purchasing equipment, software and staffing often results in higher costs to an organization having their own private cloud.

1.2.3 HYBRID CLOUD

Hybrid clouds are more complex than the other deployment models, since they involve a composition of two or more clouds (private, community, or public). Each member remains a unique entity, but is bound to others through standardized or proprietary technology that enables application and data portability among them.

A hybrid cloud is a composition of at least one private cloud and at least one public cloud. A hybrid cloud is typically offered in one of two ways: a vendor has a private cloud and forms a partnership with a public cloud provider, or a public cloud provider forms a partnership with a vendor that provides private cloud platforms Hybrid cloud infrastructure is a composition of two or more clouds that are unique entities, but at the same time are bound together by standardized or proprietary technology that enables data and application portability. In hybrid cloud, an organization provides and manages some resources in house and some out-house.

1.2.4 COMMUNITY CLOUD

A community cloud falls between public and private clouds with respect to the target set of consumers. It is somewhat similar to a private cloud, but the infrastructure and computational resources are exclusive to two or more organizations that have common privacy, security, and regulatory considerations, rather than a single organization. The community cloud aspires to combine distributed resource provision from grid computing, distributed control from digital ecosystems and sustainability from green computing, with the use cases of cloud computing, while making greater use of self-management advances from autonomic computing.
It replaces vendor clouds by shaping the underutilized resources of user machines to form a community cloud, with nodes potentially fulfilling all roles, consumer, producer, and most importantly coordinator.

1.3 DISASTER RECOVERY IN CLOUD

Disaster Recovery lets administrators of small sites to protect their virtual workloads from a wide class of disasters by replicating those workloads into the cloud. This cloud is provided by an external organization. The entire infrastructure of the cloud including storage, computational resources and network. The external organization will charge a fee for provisioning of the resources. Some of the popular Cloud IaaS providers are now providing Disaster recovery options. It is termed as Recovery-as-a-Service.

One such IaaS giant VMware does it through their product VCloud Air Disaster Recovery. VCloud Air Disaster Recovery uses the host-based replication feature to copy the protected source virtual machines into the infrastructure of the cloud provider. The virtual machines are replicated to the Data centers provided by the cloud organization. The virtual machine files such as .vmx .vmfs and etc are transported through a virtual private network to the cloud organization and over there these files are used to replicate the virtual machine. If a disaster occurs, Disaster Recovery servers can convert the replicated data into vApps and virtual machines in the cloud.

There are two sites one is source site and other is the target site.

- **SOURCE SITE:-**The source side provides business-critical data center services. The source site can be any site where cloud manager such as in case of VMware Center Server supports a critical business need. The source site may or may not be vulnerable to external damages or internal damages that is it may be or may not be prone to disasters.
- **TARGET SITE:-**The target site is an alternative facility to which you can migrate these services. The target site can be in another location, or in the same facility to establish redundancy. The target site is

usually located in a facility that is unlikely to be affected by environmental, infrastructure, or other disturbances that might affect the source site.

For efficiently performing the process of disaster recovery there are some necessary conditions which need to be fulfilled.
The following are the requirements for environments at each site:

- Each site must have at least one data center.
- The target site must have hardware, network, and storage resources that can support the same virtual machines and workloads as the source site.
- The sites must be connected by a reliable IP network.
- The target site must have access to networks (public and private) comparable to those on the source site, although not necessarily the same range of network addresses.

1.4CLOUD OPERATIONS

The virtual machines in the cloud are involved in a lot of operations. These operations are mainly concerned with the functioning of the virtual machines and involve only virtual machine. The cloud operations are

1. CLONING
2. CLUSTERING
3. CONFIGURE REPLICATION
4. PLANNED FAILOVER
5. TEST FAILOVER
6. SYNC
7. REVERSE REPLICATION
8. TEST CLEANUP

1.4.1 CLONING

Cloning is a process by which a replica or a copy of an existing virtual machine is created. The virtual machine which is cloned is called the parent virtual machine and the virtual machine which is there after cloning is done called the child virtual machine. The clone is a separate virtual machine. Clones are useful when you must deploy many identical virtual machines to a group.
There are two types of clones:-

a. **Full Clones:** A full clone is an independent virtual machine, with no need to access the parent. Full clones do not require an ongoing connection to the parent virtual machine. Because a full clone does not share virtual disks with the parent virtual machine, full clones generally perform better than linked clones. However, full clones take longer to create than linked clones. Creating a full clone can take several minutes if the files involved are large.
b. **Linked Clones:** A linked clone is made from a snapshot of the parent. All files available on the parent at the moment of the snapshot continue to remain available to the linked clone. Ongoing changes to the virtual disk of the parent do not affect the linked clone, and changes to the disk of the linked clone do not affect the parent. A linked clone must have access to the parent. Without access to the parent, a linked clone is disabled. Linked clones are created swiftly, so you can easily create a unique virtual machine for each task you have. You can also easily share a virtual machine with other users by storing the virtual machine on your local network, where other users can quickly make a linked clone. This facilitates collaboration: for example, a support team can reproduce a bug in a virtual machine, and an engineer can quickly make a linked clone of that virtual machine to work on the bug.

1.4.2 CLUSTERING

Clustering, a new concept in virtual infrastructure management, gives the power of multiple hosts with the simplicity of managing a single entity. New cluster support in cloud infrastructure reduces the management complexity by combining standalone hosts into a single cluster with pooled resources and inherently higher availability. Clusters helps to aggregate the hardware resources of individual Server hosts but manage the resources as if they resided on a single host. Henceforth a virtual machine, can be given resources from anywhere in the cluster, rather than only confining it to a specific ESX Server host. Two important phenomenon which provide help with the management of clusters are High Availability, and Dynamic Resource Scheduling. High Availability allows virtual machines running on specific hosts to be restarted automatically using other host resources in the cluster in the case of host machine failures. Dynamic Resource Scheduling provides automatic initial virtual machine placement and makes automatic resource relocation and optimization decisions as hosts are added or removed from the cluster or the load on individual virtual machines goes up or down. Dynamic Resource Scheduling also makes cluster wide resource pools possible.

1.4.3 CONFIGURE REPLICATION

The configure replication is the very first phase of the cloud operations which are involved in Disaster Recovery. It is not only the first operation but also one of the compulsory operations, because for every other operation configure replication is needed. Configure Replication can protect individual virtual machines and their virtual disks by replicating them to another location. In configure replication, a recovery point objective (RPO) is set to determine the period of time between replications. For example, an RPO of 1 hour seeks to ensure that a virtual machine loses no more than 1 hour of data during the recovery. For smaller RPOs, less data is lost in a recovery, but more network bandwidth is consumed keeping the replica. Configure Replication guarantees crash consistency amongst all the disks that belong to a virtual machine.

1.4.4 PLANNED FAILOVER

Planned failover is a planned migration for replications to cloud. Planned migrations allow you to move your workloads from local site to cloud organization. When a planned migration operation runs, the replication source virtual machine is powered off. The placeholder virtual machine that is created in the cloud during replication is configured to run as a fully functional virtual machine. When the recovered virtual machine is powered on in the target cloud site, the replication task on the source is no longer active.

1.4.5 TEST FAILOVER

Test failover allows to verify that source data is replicated correctly on the target site. While initiating a replication task to cloud, Disaster Recovery to Cloud creates a placeholder virtual machine on the target virtual data center. If the replication uses a seed, that seed is the placeholder virtual machine. The placeholder virtual machine is not visible on the network and is not accessible until you recover it or run a test recovery.

1.4.6 SYNC

Sync is an abbreviated version of synchronization. Sync operation runs from time to time automatically and keep the two copies of the virtual machines, one on the source side and other on the target side. It synchronizes the virtual machines on two both the organization cloud and local cloud. It also keeps the time synchronized between the source site and target site with respect to the NTP servers.

1.4.7 REVERSE REPLICATION

Reverse Replication is in some way opposite of Configure Replication. Though not exactly the same and cannot be replaced for configure replication. Configure replication creates virtual machine replicas on the organization cloud. What happens after a disaster is critical and messy. The local cloud virtual machines no longer exist. The virtual machines in the local cloud is either destroyed, erased or damaged such that they cannot be used, or might be some essential information in the virtual machines are damaged. In such a situation we can assume that the local copies of the virtual machines can’t be used. But the copies on the cloud organization is not damaged, so in that case the images from the cloud organization is taken back and replicated to the local datacenter. This process of replicating the virtual machines from organization datacenter to local datacenter is called Reverse Replication.

1.4.8 TEST CLEANUP

After the replication works are completed it is required to clean the datacenters off the replicated virtual machines. That is when two copies of a virtual machine is not required, usually the virtual machine on the cloud organization is cleaned. It is usually because the organization cloud provides the cloud infrastructure at a price and to cut the price the usually the organization cloud is cleaned up. This process of cleaning up the organizational cloud infrastructure from the already replicated virtual machines in termed as Test Cleanup. Test Cleanup is an important operation in the entire cloud computing. The process not only removes the virtual machines but also frees up the variables, static or non-static variables, reset all the pointers, de-allocates the memory and stored. It takes care of the end to end cleanup process.

1.5 ORGANIZATION OF THE REMAINING THESIS

In the remaining portion of the thesis we will be discussing about what we will be doing, what are the previous works which are done in this domain or in related domain, we will focus on the implementation details, problems faced and what more works that can be done. In the next chapter we will be dealing with the related works which are previously done. The works which we will be doing here, has a huge dependency on the log files of the cloud. So we will be exploring through some of the related works which were previously done with the log files. After that we will be discussing about the project and the implementation details. The project mainly focuses on the development of a tool which helps to easily analyze the log files. We will be discussing finer details of about it in those chapters. Also we will focus on some of the advantages and disadvantages about the same. After this we will be having a conclusion in which we will sum up the entire thing, purpose and future works which can be done.

CHAPTER 2

RELATED WORKS

2.1 LOG ANALYSIS

We did survey on some of the cloud forensics model that have been implemented in the Cloud environment.
In the paper Cloud Application Logging for Forensics by Rafael Marty their paper discusses a logging framework and guidelines that provide a proactive approach to logging to ensure that the data needed for forensic investigations is generated and collected. The standardized framework eliminates the need for logging stakeholders to reinvent their own standards. The part of the SaaS infrastructure is setup up on top of Amazon AWS cloud, this infrastructure is a three tier infrastructure. The applications that were used include Django, JavaScript, Java Backend, My SQL and Apache. It also mentions usage of an operating system for system status and operation failure related works. It then starts with description of the products and explains various ways by which the product integrates and the pitfalls of each of the product.

It first talks about Django, Django developers are expected to use the standard python logging libraries. No additional support for logging is built into Django. Because of minimal logging support implementation of a separate logging solution for Django and instrumentation of Django, or more precisely, the Django authentication methods to write log entries was necessary. It wrote small logging library that can be included in any code. Once included, it exports logging calls for each severity level, such as debug (), error (), info (), warn (), and feature () the first five calls all work similar; they require a set of key-value pairs.
For example
Error ({'object'customer','action':'delete', 'reason': 'does not exists','id':'22'}) The corresponding log entry looks as follows: 2010 Jan 28 13:03:47 127.0.0.1 severity=ERROR, user=pixlcloud_zrlram,object=customer, action=delete, status=failure, reason=does not exist,id=22, request_id=dxlsEwqgOxAAABrrhZgAAAAB.[3]

The extra key-values in the log record are automatically added to the log entries by our logging library without burdening the developer to explicitly include them. The unique id is extracted from the HTTP request object. This unique ID for each user request is used on each application tier to allow correction of messages. The user can also be extracted through the request object. The severity is included automatically based on the logging command. These different levels of logging calls (i.e., severities) are used to filter messages, and also to log debug messages only in development. In production, there is a configuration setting that turns debug logging.

It logs the following category fields: an object, an action, a status, and if possible a reason. This allows to do very powerful queries on the logs, like looking for specific objects or finding all failed calls. In addition to the regular logging calls, it implements separate call to log feature usage. This goes back to the use-cases where the interested in how much each of the features in the product is used.

Secondly, it talks about the grim logging techniques in JavaScript, because the logs end up in the client side that is why it is very challenging to collect it and correlate with other logs. Because of this problem, they have built a little logging library that can be encompassed in the HTML code. It called the library results in an Ajax call to an end point on the server that will log the message which is handed as a payload to the call. In order to not spawn too many HTTP requests, the library can be used in batch mode to bundle multiple log records into a single call.

Thirdly, it speaks about Apache, apache logging system is based on Apache's Defaults. It mentions of manipulating the server to get timestamp, originating server, session ID, URL accessed, HTTP return code etc.

```
76.191.189.15 - - [29/Jan/2010:11:15:54 -0800] "GET / HTTP/1.1" 200 3874 "http://pixlcloud.service.ch/"
"Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; en-us) AppleWebKit/531.21.8 (KHTML, like Gecko)
Version/4.0.4 Safari/531.21.10" duvpqQqgOxAAABruAPYAAAAE[2]
```

They are configuring the log format to hold more information. These information are necessary for the analysis of log files. So this early configuration in the log files later actually helps in the long run.

Fourthly, the MySQL setup is using Amazon's Relational Database Service. The problem with this approach is that it does not get any MySQL logs. It talks about configuring MySQL to send logs to a separate database table and then exporting the information from there. We haven't done so yet. One of the challenges with setting up MySQL logging will be to include the common session ID in the log messages to enable correlation of the database logs with application logs and Web requests.

Then it says that since the infrastructure is heavily relied on handling request and processing backend data. Collected is used to monitor each computer or machine and to monitor individual metric. Each machine is monitored for data, but the collection of the data is centralized with the help of alerts and scripts codes. Some other log files in the operating systems are also collected for monitoring. It plans to identify misconfigurations and potential attacks by logging failed requests. It talks about mining the logs and then alert the users of their misconfigurations. To assess the attacks, it not only just logs blocked connections, but also selected passed ones. For example, it talks about how it monitors servers for strange outbound requests that haven't been seen

before and correlated those with observed blocked connections to get an idea about the origin of these new outbound connections. Then use the combined information to determine whether the connections are benign or should be of concern.
There are some topics which are left in this paper, such as security visualization, forensic timeline analysis, log review, log correlation, and policy monitoring. The paper is useful for application developers it clearly draws a commensurate between application development, log file, cloud computing. Our work is diligently related with that of this paper, it clearly specifies the problems of log collection and how to resolve those snags.

2.2 DOMAIN SPECIFIC LANGUAGE

The thesis also refers to another paper which deals with the knowledge of domain specific languages, domain specific languages play a vital role because it will be the basis of development of the tool for log forensics. The paper is Domain Specific Languages by Duersen A.V. ,Klint P,Visser J. here they are describing about how the domain specific language give flexibility and reduces the complexity by taking the advantages of specific properties of a particular application domain. [5] It deals with implementation techniques, design methodologies and examples used for developing Domain Specific Languages. It says about Generic Solution and Specific Solution in which a generic solution is a solution which is sub-optimal and does not give a proper solution. It states that a more specific solution gives the optimal solution. The domain specific languages solve this problem. It states that at first all the languages are developed to deal with the problems which are specific in nature but slowly and gradually, people started developing more generic languages like C++ , Java. Domain specific language is a developed to solve a specific problem and provide specific solution. Domain specific language is a paradigm by which a new language is prepared specific to the application in which it is used. It states that language such as FORTRAN,COBOL[3] are not small, they are vast and powerful, as a result they cannot be regarded as domain specific languages. The key characteristic of Domain Specific Language is their focused expressive power. The paper describes how different languages can be used as a sublanguage in a Domain Specific Language. Languages such as COBOL and FORTRAN can be visualized as languages that are tailored for scientific purposes and business. The General purpose language used as sub-language of Domain specific language can extend the functionality of the domain specific language by incorporating more expressive power into it. Domain specific languages are usually declarative they can be used as a programming language. It then describes about the risk and opportunities of the Domain Specific Languages. [4] The given below are the risks and the opportunities of Domain Specific Languages that are pointed out in the thesis.
The benefits of Domain Specific Languages include:

- Domain Specific Languages allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. Consequently, domain experts themselves can understand, validate, modify, and often even develop Domain Specific Language programs. [1]

- Domain Specific Language programs are concise, self-documenting to a large extent, and can be reused for different purposes .
- Domain Specific Languages enhance productivity, reliability, maintainability, and portability.
- Domain Specific Languages embody domain knowledge, and thus enable the conservation and reuse of this knowledge.
- Domain Specific Languages allow validation and optimization at the domain level.

The disadvantages of the use of a Domain Specific Language are:

- The costs of designing, implementing and maintaining a Domain Specific Language.
- The costs of education for Domain Specific Language users. The limited availability of Domain Specific Languages. The difficulty of finding the proper scope for a Domain Specific Language.
- The difficulty of balancing between domain-specificity and general-purpose programming language constructs. The potential loss of efficiency when compared with hand-coded software.

It also gives some examples of Domain Specific Language, Domain Specific Languages, and then talks about the design methodologies of Domain Specific Language Design. It starts with analysis portion which includes identification a problem, gathering all relevant knowledge in the respective domain, clustering the knowledge based on semantics, Designing a Domain Specific Language that concisely describes the application in the respective domain. It then talks about the implementation, with construction of library that deals with the implementation details, such as designing a compiler that translates a Domain Specific Language program to corresponding sequence of library call. Then finally they are talking of writing the Domain Specific Language programs for desired applications and compiling the Domain Specific Language. The domain analysis is an important factor of Domain Specific Language; it is usually done by Domain Analysts. A domain analyst is like a system analyst who examines the needs of a system. The domain analyst is like a systems analyst, except that the goal is to support the development of families of related systems, not just one-of-a-kind productions. It also talks about Domain engineering; it originates from the research in the area of software usage. It clearly specifies that a prerequisite to developing a DOMAIN SPECIFIC LANGUAGES is mature domain knowledge.

Then it talks about the DOMAIN SPECIFIC LANGUAGES implementation details in which it clearly specifies that there are two types of implementation, firstly implementing a interpreter or implementing a compiler. The main advantage of building a compiler or interpreter is that the implementation is completely tailored towards the DOMAIN SPECIFIC LANGUAGES and no reductions are necessary regarding notation, primitives and the like. Also, at the domain level error detection, static analysis, and optimizations is done .An important problem is the cost of building a compiler or interpreter from scratch, and the lack of reuse from other DOMAIN SPECIFIC LANGUAGES implementations, although some DOMAIN SPECIFIC LANGUAGES tool sets are particularly designed to overcome such problems. ADOMAIN SPECIFIC LANGUAGES can also be implemented by extending a given base language. [2]A general-purpose language such as Java can be combined with domain-specific constructs. The main advantage of this approach is that all features of the base language remain available and need not be re-implemented. It said when such an extension of a language is used to build the DOMAIN SPECIFIC LANGUAGES there are three different ways in which it can be used.

- **Embedded language or domain specific languages**
  Prevailing mechanisms such as characterizations for functions or operators with user-defined syntax are used to construct a library of domain-specific operations. The syntactic mechanisms of the base language are used to express the idiom of the domain. A benefit of this approach is that the compiler or interpreter of the base language can be reprocessed for the DOMAIN SPECIFIC LANGUAGES. The main inadequacy of this approach is in the expressiveness of the syntactic mechanisms in the base language.
- **Pre Processing or Macro Processing**
  In this approach the new constructs are translated to statements in the base language by a preprocessor. The main advantage of this approach is simplicity. Its main disadvantage is that static checking and optimization are not done at the domain level. As a result, generated code is error prone, and the user is provided with feedback on these errors at the level of the base language, only at run-time.
- **Extensible compiler or Interpreter**
  This approach is similar to the previous one, but the preprocessing phase is now integrated in the compiler. The advantage is that more type checking and better optimization is possible.

The domain specific language is also associated with the term Aspect Oriented Programming in the paper, because Domain Specific Languages helps to describe the aspect of a systems behavior that is orthogonal to its main functionality. An aspect weaver is then used to generate domain specific code and merge it with the main code.

One more related work is there regarding Domain Specific Languages it is a paper called Comparing General-Purpose and Domain-Specific Languages: An Empirical Study by Kosar T, Oliviera N,Marnik M, Varanda M J, Matej C, Daniela da Cruz and Henriques P R.The paper states that the primary focus of a programming language is to make programming more efficient and provide good level of abstraction by hiding the needless details and emphasizing on finding the solution without compromising with the expressive power. It states that DOMAIN SPECIFIC LANGUAGES are more expressive with general purpose language this is difficult to achieve, since general purpose language does not offer solution specific to the problem. General Purpose

Languages are perfectly established in the life cycle [2] of software development. On the other hand integration of DOMAIN SPECIFIC LANGUAGES into the software development life cycle is not so smooth. The paper deals with comparing the effectiveness of domain specific languages over general purpose languages. Our work is mostly based on Domain Specific Languages so it is required to analyze how domain specific language works in respect to general purpose languages. The procedure they followed is empirical results that compare ten diverse implementation approaches for DOMAIN SPECIFIC LANGUAGESs, conducted on the same representative language, and are provided. Among the implementation approaches, the comparison also included the XML-based approach. From this study, it can be concluded that XML-based approach has some disadvantages . Although, XML usage and its tool support are spreading, this is one of the reasons that XAML, as a representative DOMAIN SPECIFIC LANGUAGES, has been chosen for this study. XAML, the Extensible Application Markup Language, is a language for construction of graphical user interfaces in Windows Presentation Foundation and Silver light applications of .NET Framework 3.5. C# Forms has been used for the comparison since it covers the same domain of graphical user interfaces. After this they are performing comparison on XAML and C#.

The Average programmers success rate for 35 programmers is given in the paper and in 10 out of the 11 specific questions the Domain Specific Language shows better result than General Purpose Language.

Then it shows how the Domain Specific Language varies with learn, perceive, evolve quotient of a programmer. It is also taken for 35 programmer. In the learning domain the mean for Domain Specific Language is 57.62%, Standard Deviation is21.90% and Standard Error Mean is 3.70% whereas the mean for general purpose language is 40.95%,the Standard Deviation is 22.99% and Standard error mean is 3.89%.

In the perceive domain the mean for Domain Specific Language is 64.57%, Standard Deviation is 19.45% and Standard Error Mean is 3.29% whereas in General Purpose Language Mean is 50.86%, Standard Deviation is 18.69%, Standard Deviation of Error is 3.16%.In the evolve domain the mean is 70.95%, Standard deviation is 20.35%, Standard error mean is 3.44% where as in general purpose language Mean is 33.34%, Standard Deviation is 26.20% and Standard Deviation Error is 4.43%.

In the aspects of closeness of mapping, viscosity, hidden dependencies, hard mental operations, imposed guess ahead, secondary notations, visibility, consistency, diffuseness, error-proneness, role expressiveness,
Abstraction gradient Domain Specific Languages scores more than General Purpose Language.

Domain Specific Languages use existing domain notation, which should be at an appropriate level of verbosity, so it is expected that they exhibit low diffuseness. On the other hand, it was shown in that plenty of low-level primitives, which are often purely syntactical, are one of the biggest cognitive barriers for end-user programmers. The paper promotes formal studies on the advantages of Domain Specific Languages over General Purpose Language explaining the difference between Domain Specific Languages and General Purpose

Languages program understanding, using the cognitive dimension framework. Questionnaires on understanding programs have been prepared and given to the programmers. Results showed that programmers’ success rate was around 15% better for Domain Specific Languages in all three groups of questions: learn, perceive and evolve, despite the fact that programmers were significantly less experienced in XAML than C# Forms. Further, the experiment measurement framework included cognitive dimensions to identify the aspects among these dimensions that are enhanced in the context of Domain Specific Language. It can be learned from the study that Domain Specific Languages are superior to General Purpose Languages in all cognitive dimensions. The cognitive dimensions, with the biggest influence in the experiment, are closeness of mappings, diffuseness, error-proneness, role expressiveness, and viscosity. Viscosity refers to the amount of effort that is needed to perform small changes. Thus with these related works we can now safely move ahead with our work in which we are doing some log analysis of cloud logs. [2] We need the ideas of Domain specific Languages as we will be working on it. There is a lot of data recorded in the log files so we need to analyze it before we start our work, the above works are helpful in our project because it gives an idea about what move to make and how to proceed with our works. The first paper clearly tells us how the work is to be organized, what is log forensics all about, what the problems in log forensics are and how they can be resolved. The next paper gives us a basic idea about a Domain Specific Language, the problems that can be faced and how they can be resolved. It discusses about the implementation details of a Domain Specific Language. The third papers justifies our use for Domain Specific Language, it explains why we should use a Domain Specific language and how it is better than General Purpose Language and empirically proves it. The tool which is being developed will be using Domain Specific Language to for log forensics. The mentioned above are very prime and rustic which needs to be integrated and polished. The knowledge gained from all of these will be later used in developing a tool which will be used for forensic study of the logs of cloud setup. The next chapter we will go into the details of the project following which we will start with the implementation of the project.

CHAPTER 3

DISASTER RECOVERY IN CLOUD

This chapter will describe the project, what is actually being done ,how is it helping and what are the issues that are faced. The project mainly deals with Disaster recovery (DR) portion of cloud computing, we will be developing a tool which will help us with tracing the life cycle of a cloud operation in a disaster recovery stack. So, it inculcates within itself some log forensic concepts. Actually log files are generated all throughout the cycle of a cloud. The project will be using VMware products such as VCenter, VCloudDirector, vSphere etc. While describing the implementation details, all these products will be discussed.

Disaster Recovery [13], as it is described before is a new type of service which is being provided by the cloud service provider. Basically during a disaster the cloud service provider moves the virtual machine from the local setup to some remote setup where some other cloud service provider offers a solution. This process of moving the virtual machines from local setup to remote setup is not only tedious but also involves a lot of complications. So, it leaves an open door for research [8].

So, starting off with Disaster Recovery first, why do we need a cloud based disaster recovery service.
There is no question that every business wants to protect their operations from downtime and loss of data. But many companies don't have the internal expertise or budget to implement the disaster recovery plan they need. Traditional disaster recovery solutions typically cost too much; they're too complex; and they are not always reliable. Cloud Disaster Recovery is preferred because:-.

1. Ease of Getting Started Deploying and managing a traditional disaster recovery plan can be complex and require time, budget, and staff that you may not have. Disaster Recovery provides an easy way to get started with an effective disaster recovery plan—without investing in any hardware, without hiring and training new specialists, and without having to invest in a secondary site. The service provides a simple, secure, automated process for replicating and recovering applications and data in the case of a local disaster or disruptive event.

2. Flexible, Lower Cost Alternative: The costs of traditional disaster recovery solutions can force you to make tradeoffs on what you can afford to protect versus what you need to protect. This can leave your organization vulnerable to having inadequate protection. Disaster Recovery addresses variable capacity requirements needed to support common DR use cases, such as replication, failover, and failback, at a significantly reduced price

point over traditional in-house disaster recovery solutions or managed service alternatives. You have the scalability to accommodate shifting requirements, you pay only for what you need, and you have flexible subscription options.

3. Simplified Environment, creating a comprehensive disaster recovery plan can be complex, whether you are trying to do it yourself or are choosing a managed service provider. Disaster Recovery is built on vSphere Replication and VCloud Air, providing a common hybrid platform from your on premises data center to the cloud. This means you can leverage the same tools, skill sets, and processes that you already use. And you'll have the confidence that you will get the same reliability, security, and support you know and trust from VMware.

The Disaster Recovery [14] service enables site administrators to protect their vSphere virtual workloads from a broad range of potential disruptions by asynchronously replicating those workloads from a source site to the cloud for recovery. The Disaster Recovery service uses vSphere Replication (host-based replication) to replicate virtual machines to VCloud Air. Since VMware products are used for functioning, so the disaster recovery details with respect to VMware products are explained. That is the disaster recovery option classifies into Recovery-as-a-Service category.

On a traditional x86 system operating system and hardware are tied together. Such as in a normal PC the windows XP is loaded on the infrastructure and works as a single unit. That is when we switch on the machine the operating system is booted, and the system starts. In order to ensure rapid recovery we need to duplicate the environment. That is if there is a failure the entire system can only be recovered if and only if there is a copy of the entire system. Now this means that everything needs to be doubled and so is the servers, storage, network, power and rackspace. And that is not all the Recovery site has to remain identical to the production site. The environment needs to be identical to the production site. Otherwise the recovery won't work.
This indirectly means that each software and hardware changes needs to replicate any minute change occurring in the hardware, such as increasing the RAM from 2gb to 8gb, or changing the network configuration, all these needs to be done in the recovery site and if there is a software change like installation of a word processing software in the computer, even that needs to be done in the recovery site.
Now in order to keep two sides same by replication can potentially fail, as it is a very challenging thing to do because of many factors such as it is a slow complex process with which involves uninstalling and reinstalling the operating system ,synchronization of time-If the time delay between the first site and the last site may not be synchronized, network complications-the network connectivity may be faulty and the transmission of information from one site to the other may be improper, authentication problems-there may be some

authentication problem, security problems and etc. So it is a tedious job for disaster recovery, and failures are bound to occur. This also makes it unreliable because any difference in the hardware and software of both sides can cause the recovery to fail.[9] Suppose in the original site the computer has a X86 architecture of the processor and in the recovery site the computer has X64 architecture, and in the original site there is a software which requires an X86 architecture but cannot be installed on a system which has a X64 processor, so then we cannot install the software in the replicated or rather the duplicated recovery system, as a result the recovery fails. Also there is another example, such as an image processing software requires a 2gb graphics card for proper running, the original system has 1gb graphics card and the recovery site has a system which has a 2gb graphics card then the image processing software will not perform that well in the original system, so recovery does not work properly. In addition, not only does it doubles the cost but also increases the task of maintenance and system change is doubled. In VMware the entire system the operating system, the hardware configuration and application software, the entire thing is taken and turned into a file or into few files, so the dependencies on individual system configuration separately like hardware, software, networking, power and etc. are reduced. Each physical system becomes a self-contained, hardware independent, virtual machine. Each of these files contains its own virtual hardware configuration, operating system installation and applications. VMware installs a thin layer of software known as the virtualization layer on top of the hardware. This thin layer of software makes it possible to run multiple virtual machines on the server safely and securely. This properties for virtualization has very important implication for disaster recovery because virtual machine stores an entire system in just some files and recovery is simple as recovering those few files. Thus recovery becomes substantially faster because there is no need for installation of an operating system or application software. A lot of time gets saved. Bare metal installation or manual process is no longer needed so a lot of time is saved. Virtual machines are hardware independent; there is no need for identical hardware now in original site and recovery site. It takes on an average 40+ hours for a physical to physical recovery, where as it takes only 4hours for a virtual to virtual recovery. The physical to physical recovery starts with configuring the hardware then installing the operating system, after this it configures the operating system, then it installs the backup agent and then starts “Single-step automatic recovery”. However, in a virtual to virtual recovery, firstly the vm configuration is restored and then the data recovery starts. The recovery is far more reliable. Testing to see the reliability of your disaster recovery is far easier, because exact copies of system can be created and can be run on the hardware for the purpose of disaster recovery without requiring extra additional hardware for the purpose of testing. Because virtual machines are hardware independent, expensive duplication of hardware at the production site is not required. It also reduces the expenses for storage, rackspace, power and server. The situation below will properly exemplify everything. Suppose a production environment has 200 servers at the disaster recovery site 200 identical physical recovery servers are required. Duplicate storage, power and networking is also needed for those 200 physical servers. With VMware software the 200 physical servers can be consolidate to 200 virtual machines. So it may only need 20 physical servers to hold 200 virtual machine. In this process there is no need for 200 physical servers, so even if 200 physical servers are there only 20 sufficient to host and 180 remains unused. So it requires no new servers, servers that are extra from the production is sufficient to do the disaster recovery. So there is a rapid and reliable recovery for disaster which is also at low cost. So, the recovery time is reduced, physical servers eliminated 300 and also consolidated a large number of server, a decreased server deployment time.

The disaster recovery in cloud is a not a single process involving only one product, it is a cycle starting from the on premise and ends in the private cloud. This cycle involves a large number of VMware products. The disaster recovery solution is provided by site recovery manager, it provides a solution for recovery, helps organizations set up a fully automated site recovery plan that will manage the failover virtual machine.

3.1 VMWARE APPLICATIONS TAKING PART IN THE DISASTER RECOVERY

3.1.1 SITE RECOVERY MANAGER

VMware has two technologies that allow to take advantage of replication. vSphere replication,(VSR).It does the process of spinning the replication by coping across the network. It is a free feature built in the vSphere. And then there is SITE RECOVERY MANAGER or the site recovery manager. It allows replication across the storage array layer and this allows to have much higher performance. SITE RECOVERY MANAGER, (SRM) is an end to end disaster recovery orchestration product; it automates the recovery of the virtual machines that resides on the replicated storage. There are basically four steps through which the disaster recovery orchestration is done by SITE RECOVERY MANAGER, these are as follows:-

1. At the protected site SITE RECOVERY MANAGER shuts down the virtual machines starting with the virtual machine having the lowest priority. This is because the way virtual machine write to the disk and ongoing changes happening in the background. So, SITE RECOVERY MANAGER tries to shut down the virtual machine so that no further changes can be made ,if that is not possible then the process continues as expected.

2. At the recovery site that is the designated recovery center it can be a datacenter next door or some datacenter in a remote location, Site Recovery Manager prepares the datastore groups for failover of the protected virtual machines. This would allow the ESXi servers in the recovery center to mount those replicated datacenter and take the necessary action to start up those virtual machines.

3. To provide more resources to the virtual machine to be powered on at the recovery site, SITE RECOVERY MANAGER suspend any virtual machine running at the recovery site that are designated as non-critical. Here the virtual machines which a marked as non-critical are shut down to make room for the other virtual machines that are considered to be machine critical to start up.

4. SITE RECOVERY MANAGER restarts virtual machine, at the recovery site, starting with virtual machines that are designated as high priority. Interesting thing about this is SITE RECOVERY MANAGER Administrators have full control in the recovery plans over designating start up orders, dependency sequences and other necessary things, so that the entire environment can come back essentially.

One importance of SITE RECOVERY MANAGER is that SITE RECOVERY MANAGER allows to share the recovery site as a result multiple virtual machine can run in the recovery site, and multiple here stands for virtual machines on different hosts or different data centers.
All of them can run on the same recovery site.
Some of the features of SITE RECOVERY MANAGER are as follows:-

1. FULL CENTRALIZED MANAGEMENT-It helps administrator to create, test and run recovery plans from a single point. It allows to do a series to customized recovery according to the administrator, and this later helps during the time of real disaster.
2. AUTOMATED DISASTER RECOVERY-It allows to build recovery plans in advance, to test those recovery plans and execute the recovery according to the recovery plans without any supervision during times of real disaster. In most cases there is a recovery plan in pen and paper, or may be a in a computer but it does not go into testing, while surprisingly while testing the recovery plan it fails, and now there is more down time than that would have been without the recovery plan.
3. SIMPLE INTEGRATION-The Site Recovery Manager easily integrates with the VCenter ,so there is no additional burden in setting up the SITE RECOVERY MANAGER, as a result allocating and managing the recovery resources become easier

Addition to Site Recovery Manager there is vSphere Replication, it allows ESXi servers to do replication on a virtual machine by virtual machine basis, and given that, this allows SITE RECOVERY MANAGER based recovery in small and medium scale. VSphere replication enables replication between heterogeneous data stores, the replication is managed as a property of the virtual machine, and also the replication performed by the vSphere Replication minimizes the effect on virtual machine workload. The SITE RECOVERY MANAGER helps in Planned Migration. The term is a bit complex, but the internal procedure is relatively simple. If anyone ever wants to do a datacenter migration, whether relocating a datacenter or reprogramming some systems from one data center to other, there are some challenges behind that. The planned migration takes care of these issues. The planned migration is different from disaster recovery because if there is an error on the way, then it stops, so that there is a reliable disaster recovery. The planned migration basically does all these in a two steps:-

1. Planned Migration shuts down the protected virtual machines and then synchronizes the disks.
2. Planned Migration stops if there is any error encountered in the process of recovery.

Site Recovery Manager also supports Automated Failback, if there is a failover, then the site recovery manager gives a help of a single button recovery, potentially can be referred as a failback. When a failover is done the virtual machines are not protected so it becomes necessary to rewrite recovery plans that will do the recovery. SITE RECOVERY MANAGER help in doing it easily. Firstly, it provides automated workflow to failback all recovered virtual machines, then it interfaces with storage to automatically reverse replication and finally replays existing recovery plans. New virtual machines are not part of failback. The benefits of automated Failback are that it simplifies the recovery process after disaster. It facilitates disaster recovery options for enterprises that are mandated to perform a true failover as a part of recovery process.

3.1.2 VSPHERE REPLICATION

The vSphere Replication working is similar to site recovery manager. It was introduced in site recovery manager. It acts as a software recovery orchestration tool. It is a part of the hypervisor. It is built in the platform and a couple of application is enough to manage in. It replicates individual virtual machine and not exactly same as array based replication where, there is a data store which is replicated. It is a vm by vm replication engine. Because it works in the hypervisor layer, it operates above the storage but below the hypervisor layer which means there is possibility of replicating between dissimilar and heterogeneous storage. So the replication can happen across any type of storage, VSA or local storage, irrespective of what that underlying technology is. The following picture shows a typical vSphere replication.

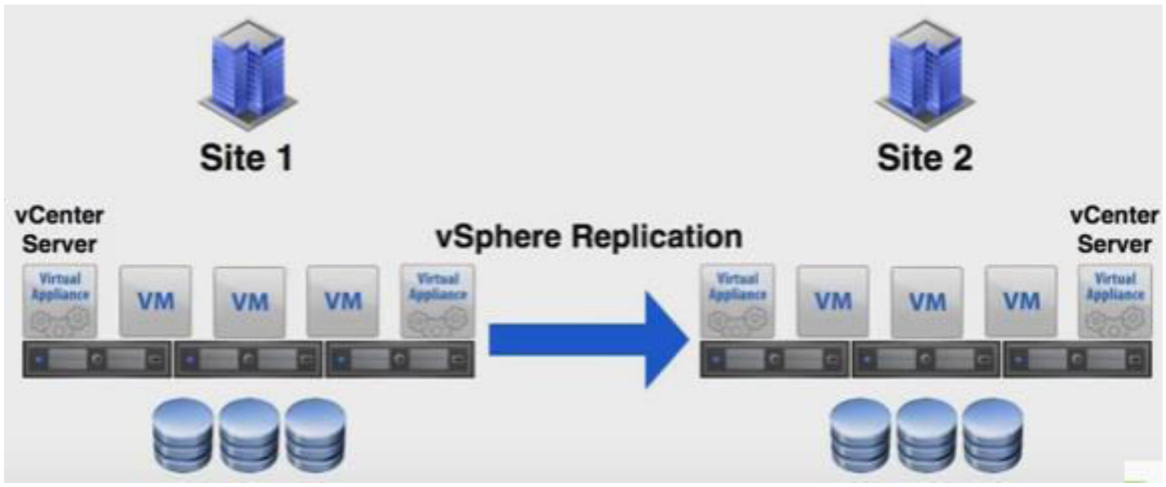


Figure 3 vSphere Replication Process

The replication happens from site number one to site number two. VSphere replication is configured on of the virtual machines. A vm in the client is clicked and then the action is clicked. From there all vSphere replication is selected and then configure replication is selected. This opens a configure replication wizard for that virtual machine. It shows two sites vc1 which is the local site and vc2 which is the site2.The second site is selected. After this a new window opens which shows the storage list, from here storage of the second site is selected. It is here the virtual machine will be replicated. After this the recovery point objective is set. It also has the provision of selecting the guest OS. At this point, the vSphere Replication will start replicating the virtual machine.

To verify the virtual machine is replicated, the vSphere replication is clicked in the vi client, and the required virtual machine can be found to be listed on the virtual machine list. The virtual machine has started its initial full synchronization. In vSphere replication, there is asynchronous replication, where there is an opportunity to choose recovery point objective between 15 minutes to 24 hours. And recovery testing and automation are done through SITE RECOVERY MANAGER Recovery plans. Basically SITE RECOVERY MANAGER sits on top of it. SITE RECOVERY MANAGER administers both array based replication and individual virtual machine replication like vSphere replication. So it is safe to say the VSphere Replication is an extension of Site recovery manager.

The architecture of the VSphere replication is provided below.

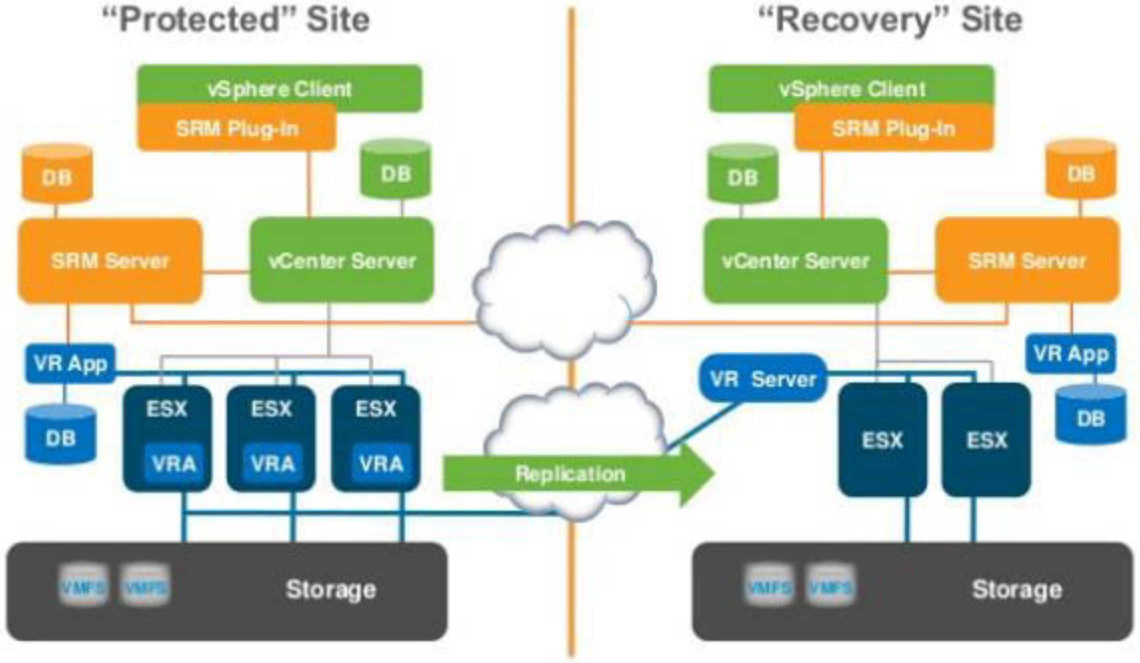


Figure 4 vSphere Replication Architecture

[https://www.google.co.in/search?q=vSphere+replication+architecture]

From SITE RECOVERY MANAGERs perspective there are two different SITE RECOVERY MANAGER Server in the two sites with a VCenter server. With SITE RECOVERY MANAGERs in this two sites two appliances are deployed VRMS-vSphere replication management server, they act as an interface between SITE RECOVERY MANAGERs and the agent it does not pertain in the replication or participate in another but what it does is that it handles the database that handles things like replication. There is VRa (Virtual Replication Appliance) which is the agent, it sits within the kernel, that is it sits within the vSphere Kernel itself and operates at the kernel level. There is also a scheduler that handles when it is time to take the pointer of the change blocks, gather them together, and hand them across the wire to the VRS at the remote site vSphere replication server. So VRS is nothing other than grab the blocks and write them down to disk using network files copied into the storage layer. VSphere replication is more efficient than normal array based replication because there is no replacement of the capabilities of storage or array based replication but it is designed to augment. Like if the entire branch is not needed to be replicated then this vSphere replication feature of site recovery manager is handful, which happens in majority of the cases, since replication is specific, so only a specific portion is needed to be replicated rather than the entire disk. Also people might not have multiple arrays. So it allows for replication within the capacity of the user with 15 minutes recovery point objective.

With vSphere replication it is possible to get only the protection but not the orchestration like testing, automation, planning. These things come with Site Recovery Manager. The replication engine is bundled with most of the vSphere edition. The virtual machine can be replicated from one location of the disk to another data store etc. It is standalone replication, because even though it comes with site recovery manager, it can be installed with as a standalone application and works absolutely fine. It is a simple recovery model, tightly integrated with the web client. Easily one can choose one or more virtual machines and choose to replicate them or protect them and then recover them. There is no orchestrated plans, one can easily right click the shadow copies and recovery them in a matter of minutes. It also supports automated protect and failback in SITE RECOVERY MANAGER for vSphere replication. So in past there is a concept called reprotect, where one could failover an entire environment or parts of an environment to another location and then simply click a button that would protect the environment back your primary site in an automated fashion. vSphere replication allows that provision of reprotect and failback and again mixing and matching vSphere replication and array based replication. It also includes an embedded database, the vSphere replication integration is very easy. Each and every individual component is deployed easily as one particular deployment. The vSphere replication is more tightly integrated with the capabilities of the VMware tools that are already doing application level quiescent. It does a synchronization of data variables from one side to another.

The following are the ways in which Site Recovery Manager compares with vSphere Replication.

1. vSphere Replication is independent of Site recovery manager, that is even though vSphere replication comes along with Site Recovery Manager, it can be installed separately and can be integrated with the vSphere client. The vSphere Replication can provide the necessary protection and replication, but it does not do testing, automated recovery. All these are done by Site Recovery Manager.
2. vSphere replication can replicate within a single VCenter, VCenter is a manager which manages multiple host, so basically it can replicate a virtual machine within any of the host that the VCenter manages. It will provide the targets from where it can be easily chosen. If only one appliance of vSphere replication is installed it can act as both management and target and replicated from on cluster to another giving a local copy that can be recovered in a matter of minutes.
3. Site Recovery manager when installed on top of vSphere Replication can easily integrate with the vSphere replication, it finds out all the replications, after the integration with the site recovery manager it can be used to gain automation, test recovery, failback, customization and recovery.

As the virtual machine writes down to disk, they Sphere replication the data passes through the hypervisor layer, the vSphere replication agent detects that it is writing through the hypervisor. And will keep track blocks of those that are written to the disks. Only the pointers to the blocks that have changed is kept, the blocks are not copied. According to the replication schedule, periodically these changes are grabbed, and ship that across to the remote appliance. With SITE RECOVERY MANAGER this load of replication can be distributed to other appliances. This load distribution option is not there in standalone vSphere Replication. Such kind of load distribution is needed when there is a huge amount of replication that needs to be done, such as replicating 500 virtual machine from one site to other. Another good feature of vSphere replication is that it allows to seed the initial copy of replication. If there are terabytes of data which needs to be replicated, it becomes a tedious task. So the user provide seed for the initial copy. The seed can be delivered through any out-of-band channel like the usb drive etc. The user directs the wizard to seed the file when configuring replication, in that way the replication becomes faster.

3.2 DISASTER RECOVERY TOPOLOGIES

1. **Active-passive Failover Process**-Here there is a production and a recovery site, for example a New York and a Chicago, so it is possible that New York will failover to Chicago or Chicago will failover to New York. The assumption is that both of them will not fail at the same time. Resources can be actively used in both sites. SITE RECOVERY MANAGER helps to shut down some low priority virtual machines and make room for some inbound virtual machine as and when it is required. It is the most common traditional Disaster Recovery Topology. It is expensive as it require dedicated resources.
2. **Active-active Failover**-This is a full bidirectional failover. Each site acts as a recovery site for the other. But one site is a dedicated production site and the other site is dedicated recovery site. It leverages recovery infrastructure for test development and training.
3. **Bidirectional Failover**-It is similar to the previous topology. Most customers won't see a difference between them. SITE RECOVERY MANAGER here deals with choosing some workloads to shut off to make room for incoming virtual machine request. It has production application at both ends. Here also each site acts as a recovery site for the other.
4. **Shared Recovery Sites**-Here the recovery site is shared among various production sites, there is a centralized recovery site and all the production site use that centralized recovery site for recovery purpose of their virtual machine. It is useful in case of many to one failover. It is particularly useful for remote office.

3.3 LIFE CYCLE OF DISASTER RECOVERY

The first step in the cycle of disaster recovery is Configure Replication. As mentioned above vSphere Replication is built into the platform, so it is integrated into the UI .Configure replication is as easy as finding the virtual machine in the inventory list, right clicking on it and saying configure replication. After this a wizard asks to set the Recovery Point Objective. It varies from 15 minutes to 24 hours. It also asks the destination where the replication is done. Disks can be chosen or dropped for replication. If there is more than one virtual machine to configure, then all the virtual machines can be selected, but the Recovery point objective will be set for all the virtual machine and not for a particular virtual machine. However the destinations can be kept same or can be made different for different virtual machine by essentially manipulating the data store mapping beforehand. For example, Data store A on the protected site gets mapped to data store D on the recovery site and Data store B on the protected site gets mapped to data store C on the recovery site, after this all the virtual machines residing in data store A gets automatically mapped to data store D and all the virtual machines residing in data store B gets automatically mapped to data store C.

Once configure replication happens the first thing which is done is an initial sync of source and target. The vm disk files in the target are checked, and the identities of those disks are compared to make sure that those disks are actually the same disks, to prevent overriding some other virtual machine disk files by accident. If the disks are the same then the blocks on the target site and the blocks on the source site is reviewed and the hashes on both sides are calculated and matched by exchanging the hashes to check which block is different. Through this way sending all the blocks over at one go is avoided. Only the changed blocks are sent. It only replicates the changed blocks necessary to align the virtual machine disk file. After the initial sync completes there is an exact replica on the other side. There is a process called full sync, it is used when something is modified or someone changes something in a disk, and there is no way to track the changed block then it becomes necessary to do a forced full sync .The process is same as initial sync but the hash matching is not done everything is replicated from one site to the other. During initial sync, the traffic goes through TCP port 31031.So to distinguish between initial sync and subsequent sync, different TCP port is configured for synchronization. Another type of synchronization is the lightweight delta sync. It is essentially similar to vm level snapshot but is very different from it. The reason why vm level snapshot is not used is because it has performance issues. The vm is stopped while taking a snapshot, so it has many issues. The delta sync is as essential as a snapshot. It is consistent with all the things in a virtual machine at a point of time. But while maintaining the consistency the I/O is not compromised with replication. Once the lightweight snapshot of the source vm is created, it is assured that the virtual machines are consistent. It basically checks for the changes in between the sync operations and send them to the recovery site. If a block is changed a multiple time between a replication only the final replication is done on the other side. The intermediate changes are not replicated. The delta sync happens over the TCP port 44046.After synchronization is switched to delta sync type after the initial sync. VSphere Replication agent tracks all changing blocks via VSCSI filter. Changed blocks are replicated as per the recovery point objective.

The vSphere replication at the very minimum provides the ‘point in time specific replica’ based on the RPO. This replica is going to be crash consistent which means that the write order is going to be preserved across all virtual disks in a vm. All the writes that happened before this time will be in the replica and all the writes that happen after this time will not be in the replica. It provides the crash consistent view of all the virtual disk in a given vm. On top of this it guarantees additional level of consistency such as file level consistency and disk level consistency. It also does an application level consistency. Every disk within the virtual machine will be consistent with the light weight delta that is shipped. If within the sync cycle the network fails the vSphere replication, snapshot the target virtual machine disk file which it discards only after a successful replication, with every sync the vSphere replication server writes a redo log for each replica. The redo log is snapshotted only after the synchronization is complete. Old replicas are collapsed only after a sync is completed. There is always at least one valid replica that corresponds to a valid light weight snapshot. If the network stops in the middle of the sync cycle the appliance knows that there is some blocks written on top of the redo logs. At the same time the protected block filter knows that it haven’t shipped all the blocks. So it basically invalidates saying that the sync cycle failed. The scheduler is going to initiate the next sync cycle almost immediately. Essentially the new sync cycle is going to write through the same redo logs. If the link goes down, the decision of when to do the replication cycle is taken by the scheduler in the source site. If the filter in the middle of the sync discovers that the connection drops, by timeout or some other kind of network failure detection technique, it will tell the scheduler that did not finish the sync. At this point the scheduler will do the smart thing to recover. Even though the blocks are retained after an incomplete sync, they are useless as the scheduler does not know the state of the block, after the link is up, since the scheduler does not know about the state of the blocks it will once again send all the blocks to the recovery sit.

Replication scheduler runs on each host. Imagine there is one vm which is configured for replication with an RPO of 15 minutes. It means that every 15 minutes it is synchronized. But if there are 100 virtual machines, if a synchronization takes a lot of bandwidth and a lot of time, and during this cycle the RPO comes in for the next virtual machine but resource for that synchronization is not available. In such a case it is possible to let the ongoing synchronization to complete but in that case the RPO is violated. This simplistic approach does not work efficiently. Instead the future is being predicted, all the past replication is analyzed for the last 48 hours, this analysis report is fed into a schedule map, using that as a data some of the sync are moved up, in place of 15 minutes they are done in 10 minutes and there is bandwidth left for other virtual machine replications for that host. The algorithm in the scheduler takes care about deciding the schedule, user cannot make a synchronization schedule. The SCSI filter keep track of the changed block. The vSphere Replication synchronizes the changed blocks.

During a failover, the virtual machine disk files in the recovery site is taken and an actual vm is surfaced out of it. Then integrate that with the VCenter. All the redo logs are stopped to ensure consistency. Site Recovery Manager gives an opportunity for doing tests. The steps performed are same as surfacing the vm. Instead of collapsing all the redo logs into a simple virtual machine disk file. A snapshot on site is created, this snapshot is a real snapshot, so the vm is writing to it, or it is modifying it, or its rebooting it etc. The vSphere Replication knows which particular application is used to create the replica. Vm never loses its base.

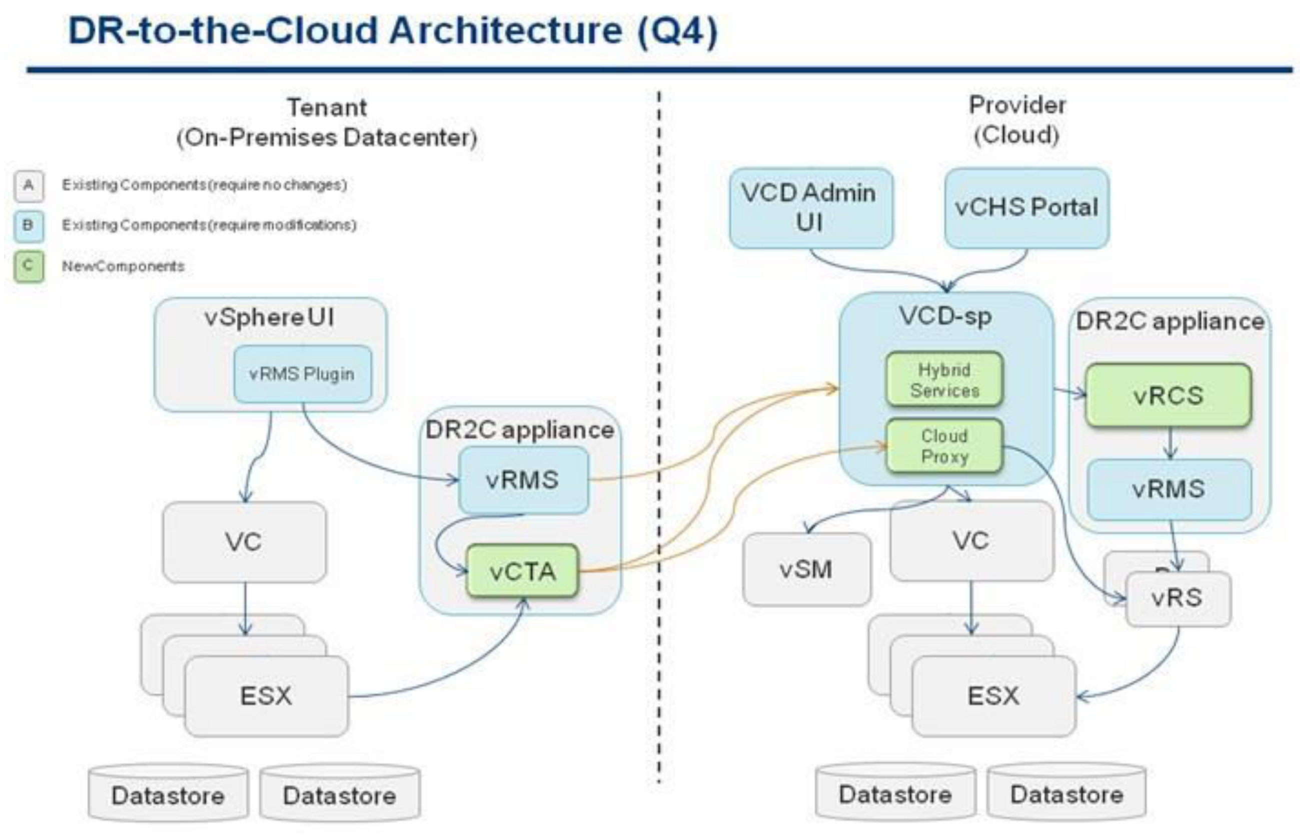


Figure 5 Architecture of Disaster Recovery in a cloud setup

Another feature of Site Recovery Manager is Automated Reprotect and Failback. If there is an actual disaster the failover might happen to the recovery site. But most of the times disasters can be predicted, in such a case the production site though may not be functional for a while, is working actually fine. In such a case there can be a planned migration to the recovery site, shutting down the protected site and waiting for the disaster to end, is what a person does. After that the protected site is powered on and the state before the disaster is what which needs to be reached. In reprotect, a configure replication is done from the recovery site back to the protected site. After the reprotect is started there will be an initial sync on what are the changes that are there in between the protected site vm’s and the recovery site vm’s, and only the changed blocks will be send over. It will not do the full terabytes of data transfer over the network. After the reprotect completes there will be some time for synchronization operations. After this short period of time, the two sites become consistent.

If a vm is selected for replication then it is also possible to replicate only a subset of the vm such as C: drive of the virtual machine can only be replicated, it is not necessary to replicated the whole virtual machine if only a particular subset of the virtual machine is what we need. The vSphere replication cannot exceed more than the bandwidth of the pipe. The moment the data exceeds the bandwidth the RPO fails. There is always one consistent image and one consistent redo logs, consistency is maintained by them. The life cycle of a disaster recovery process is described above. The products involved in this life cycle generates log files throughout the entire process of disaster recovery which, these log files will be analyzed for disaster recovery now. The log files have some issues , in the next section the issues regarding log file analysis will be discussed. The main purpose of the project is to create a tool which is going to do efficient, reliable and fast log analysis, which will help to trace the life cycle of the replication process in the stack. The tool is developed on Domain Specific Language. The Domain Specific Language is developed by extending the Groovy programming language. In the next chapter the details about logs and types of logs, which the products generate will be discussed

CHAPTER4

LOG ANALYSIS AND APPLICATIONS

4.1 CLOUD LOGS

A cloud-based application stores logs on multiple servers and in multiple log files. The volatile nature of these re-sources causes log files to be available only for a certain period of time. Each layer in the cloud application stack generates logs, the network, the operating system, the applications, databases, network services, etc. Once logs are collected, they need to be kept around for a specific time either for regulatory reasons or to support forensic investigations. We need to make the logs available to a number of constituencies; application developers, system administrators, security analysts, etc. They all need access, but only to a subset and not always all of the logs. Platform as a service (PaaS) providers often do not make logs available to their platform users at all. This can be a significant problem when trying to analyze application problems. For example, Amazon does not make the load balancer logs available to their users. And finally, critical components cannot or are not instrumented correctly to generate the logs necessary to answer specific questions. Even if logs are available, they come in all kinds of different formats that are often hard to process and analyze.

LOGGING GUIDELINES

To address the challenges associated with the information in log records, we need to establish a set of guidelines and we need to have our applications instrumented to follow these guidelines.

When to Log

When do applications generate log records? Making the decision when to write log records needs to be driven by use-cases.

These use-cases in cloud applications surface in four areas:

- Business relevant logging
- Operations based logging
- Security (forensics) related logging
- Regulatory and standards mandates

As a rule of thumb, at every return call in an application, the status should be logged, whether success or failure. Following this way errors are logged and activity throughout the application can be tracked.

Business

Business relevant logging covers features used and business metrics being tracked. Tracking features in a cloud application is extremely crucial for product management. It helps not only determine what features are currently used, it can also be used to make informed decisions about the future direction of the product.
Monitoring service level agreements (SLAs) fall under the topic of business relevant logging as well. Although some of the metrics are more of operational origin, such as application latencies.

Operational

Operational logging should be implemented for the following instances:

- Errors are problems that impact a single application user and not the entire platform.
- Critical conditions are situations that impacts all users of the application. They demand immediate attention
- System and application start, stop, and restart. Each of these events could indicate a possible problem. There is always a reason why a machine stopped or was restarted.
- Changes to objects track problems and attribute changes to an activity. Objects are entities in the application, such as users, invoices, or goods.

Other examples of changes that should be logged are:

1. Installation of a new application (generally logged on the operating system level).
2. Configuration change
3. Logging program code updates enable attribution of changes to developers.
4. Backup runs need to be logged to audit successful or failed backups.

Security

Security logging in cloud application is concerned with authentication and authorization, as well as forensics support. In addition to these three cases, security tools (e.g. Intrusion detection or prevention system or anti virus tools) will log all kinds of other security related issues, such as attempted attacks or the detection of a virus on a system.

Cloud-applications should focus on the following use-cases:

1. Login / logout (local and remote)
2. Password changes / authorization changes
3. Failed resource access (denied authorization)

4. All activity executed by a privileged account Privileged accounts, admins, or root users are the ones that have control of a system or application. They have privileges to change most of the parameters in the application. It therefore is crucial for security purposes to monitor very closely what these accounts are doing

4.1.1 CHALLENGES IN LOG ANALYSIS

If log analysis is the solution to many of our needs in cloud application development and delivery, we need to have a closer look at the challenges that are associated with it.
The following is a list of challenges associated with cloud-based log analysis and forensics:

- Decentralization of logs
- Volatility of logs
- Multiple tiers and layers
- Archival and retention
- Accessibility of logs
- Nonexistence of logs
- Absence of critical information in logs
- Non compatible / random log formats

The first five challenges can be solved through log management. The remaining three are more intrinsic problems and have to be addressed through defining logging guidelines and standards

4.1.2 LOG MANAGEMENT

Solving the cloud logging problems outlined in the last section requires a log management solution or architecture to support the following list of features:

- Centralization of all logs
- Scalable log storage
- Fast data access and retrieval
- Support for any log format
- Running data analysis jobs
- Retention of log records
- Archival of old logs and restoring on demand

- Segregated data access through access control
- Preservation of log integrity
- Audit trail for access to logs

These requirements match up with the challenges defined in the last section. However, they do not address the last three challenges of missing and non-standardized log records.
For example, if machines are pegging at a very high load, new machines can be booted up or machines are terminated if they are not needed anymore without prior warning.
Note that in some cases, it is not possible to change anything about the logging behavior as we cannot control the code of third-party applications.
A general rule of thumb states that a log record should be both understandable by a human and easily machine process able.

4.1.3 LOGGING ARCHITECTURE

A log management system is the basis for enabling log analysis and solving the goals introduced in the previous sections. Setting up a logging framework involves the following steps:

- Enable logging in each infrastructure and application component
- Setup and configure log transport
- Tune logging configurations

4.1.3.1 ENABLE LOGGING

As a first step, we need to enable logging on all infrastructure components that we need to collect logs from. Note that this might sound straight forward, but it is not always easy to do so. Operating systems are mostly simple to configure. In the case of UNIX, syslog [15] is generally already setup
and logs can be found in/var/log
The hard part with OS logs is tuning.
Configuring in databases is a level harder than logging in operating systems. Configuration can be very tricky and complicated.
Each of them has their own sets of features, advantages, and disadvantages. It gets worse though; logging
from within your applications is most likely non-existent. Or if it exists, it is likely not configured the way your use-cases demand; log records are likely missing and the existing log records are missing critical information.

4.1.3.2 LOG TRANSPORT

Setting up log transport covers issues related to how logs are transferred from the sources to a central log collector.

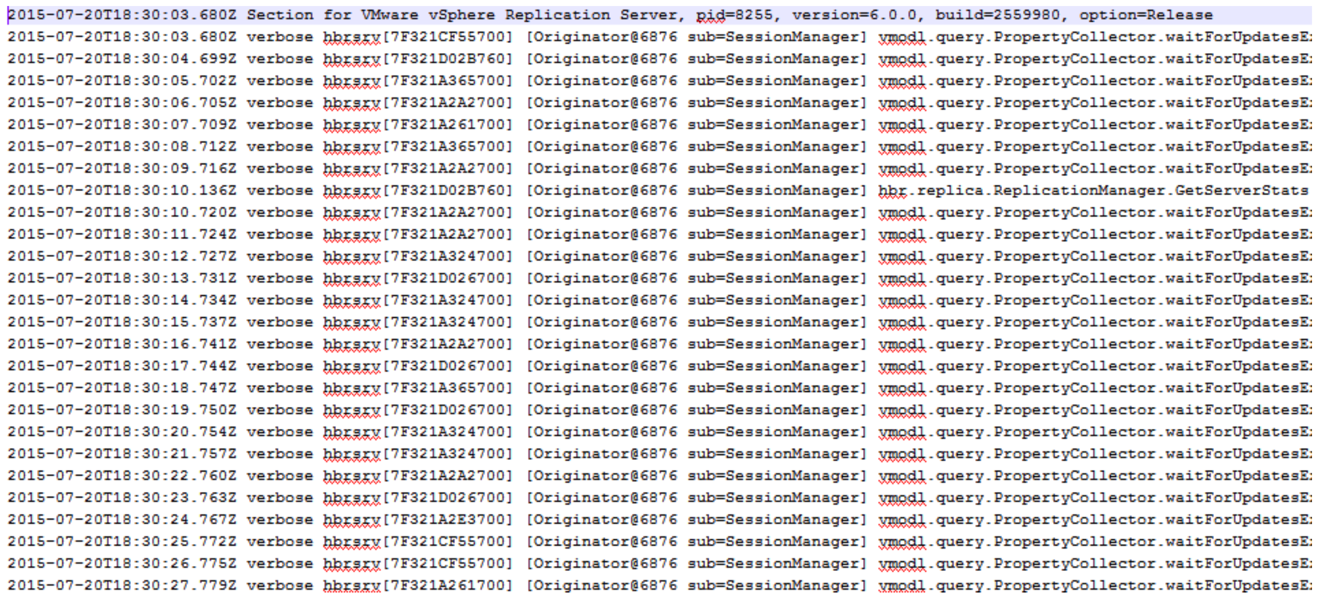
Here are issues to consider when setting up the infrastructure:

- Synchronized clocks across components
- Reliable transport protocol
- Compressed communication to preserve bandwidth
- Encrypted transport to preserve confidentiality and integrity

4.1.3.3 LOG TUNING

Log data is now centralized and we have to tune log sources to make sure we get the right Type of logs and the right details collected. Each logging component needs to be visited and tuned based on the use-case.

Given below is a image of a log file which is the VMware replication server in short it is HBR server.



```
2015-07-20T18:30:03.680Z Section for VMware vSphere Replication Server, pid=8255, version=6.0.0, build=2559980, option=Release
2015-07-20T18:30:03.680Z verbose hbrsrv[7F321CF55700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:04.699Z verbose hbrsrv[7F321D02B760] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:05.702Z verbose hbrsrv[7F321A365700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:06.705Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:07.709Z verbose hbrsrv[7F321A261700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:08.712Z verbose hbrsrv[7F321A365700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:09.716Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:10.136Z verbose hbrsrv[7F321D02B760] [Originator@6876 sub=SessionManager] hbr.replica.ReplicationManager.GetServerStats
2015-07-20T18:30:10.720Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:11.724Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:12.727Z verbose hbrsrv[7F321A324700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:13.731Z verbose hbrsrv[7F321D026700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:14.734Z verbose hbrsrv[7F321A324700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:15.737Z verbose hbrsrv[7F321A324700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:16.741Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:17.744Z verbose hbrsrv[7F321D026700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:18.747Z verbose hbrsrv[7F321A365700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:19.750Z verbose hbrsrv[7F321D026700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:20.754Z verbose hbrsrv[7F321A324700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:21.757Z verbose hbrsrv[7F321A324700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:22.760Z verbose hbrsrv[7F321A2A2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:23.763Z verbose hbrsrv[7F321D026700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:24.767Z verbose hbrsrv[7F321A2E3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:25.772Z verbose hbrsrv[7F321CF55700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:26.775Z verbose hbrsrv[7F321CF55700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
2015-07-20T18:30:27.779Z verbose hbrsrv[7F321A261700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE:
```

Figure 6 Portion of a log file

4.1.4 THINGS TO LOG

We are now leaving the high-level use-cases and infrastructure setup to dive into the individual log records.

How does an individual record have to look?

At a minimum, the following fields need to be present in every log record: Timestamp, Application, User, Session ID, CGID, OPID, Operation ID, Severity, Reason, Categorization. These fields help answer the questions: when, what, who, and why. Furthermore, they are responsible for providing all the information demanded.

A timestamp is necessary to identify when a log record or the recorded event happened. Timestamps are logged in a standard format. The application field identifies the producer of the log entry. A user field is necessary to identify which user has triggered an activity. Use unique user names or IDs to distinguish users from each other. A OPID helps track a single request across different applications and tiers. The challenge is to share the same ID across all components. A severity is logged to filter logs based on their importance. A severity system needs to be established. For example: debug, info, warn, error, and crit. The same schema should be used across all applications and tiers. A reason is often necessary to identify why something has happened. For example, access was denied due to insufficient privileges or a wrong password. The reason identifies why. As a last set of mandatory field, category or taxonomy fields should be logged.

Categorization is a method commonly used to augment information in log records to allow addressing similar events in a common way. This is highly useful in, for example, reporting. Think about a report that shows all failed logins. One could try to build a really complicated search pattern that finds failed login events across all kinds of different applications or one could use a common category field to address all those records.

4.2 APPLICATION USED TO BUILT THE TOOL

The following is a brief overview of the application components that are touched upon

4.2.1 GROOVY

Groovy is an object-oriented programming language for the Java platform. It is a dynamic language with features similar to those of Python, Ruby. Groovy code is compiled to byte code that is executed by the Java Virtual Machine (JVM). We choose Groovy as the DOMAIN SPECIFIC LANGUAGES programming language for its simplicity and flexibility, as well as the performance and stability of the JVM. Because Groovy is compiled to byte code that runs on the JVM Java Virtual Machine (JVM), 99% of Java code is valid Groovy. Groovy offers programmers the ability to toss aside all the humdrum conventions of brackets and semicolons, to write simpler programs that can leverage all that existing Java code. Everything runs on the JVM. Not only that, everything links tightly to Java JARs, so you can enjoy your existing code. The Groovy code runs like a dynamically typed scripting language with full access to the data in statically typed Java objects. There's all of the immense power of the Java code base with all of the fun of using closures, operator overloading, and polymorphic iteration -- not to mention the simplicity of using the question mark to indicate a check for null pointers. It's so much simpler than typing another if-then statement to test nullity. Naturally, all of this flexibility tends to create as much logic with a tiny fraction of the keystrokes. Given below is the image how groovy compiles a code and how it is similar to java and other languages.

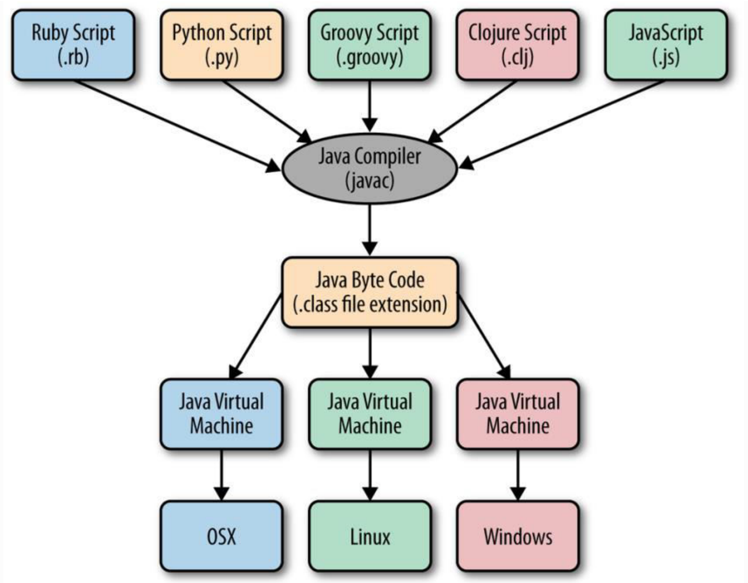


Figure 7 Groovy language compiler comparison with similar languages compilers and how it relates to Java

4.2.2 DOMAIN SPECIFIC LANGUAGES

DOMAIN SPECIFIC LANGUAGES stands for domain specific language. Languages like C++, Java, C# are all going to allow us the same computations, that is if it is possible to do it in one language then it is also possible to do it in other language. But that does not say anything about how easy it is, how much help the language will give you about making the mistake and how quick the final result will be. The domain specific language provides

- Ease of coding, it includes various coding features in it kind closures etc.
- It provides a even higher level of abstraction and hence helps in eliminating the mistakes
- It gives the result faster. The ability of the compiler to do run time optimization.

Matlab which is very good in doing mathematical operations, or matrix multiplication, the same coding of matrix multiplication can be done in languages like C, C++ or C#. But languages like matlab which uses a * for matrix multiplication is going to be more concise. Languages specifically designed for mathematics will easily find out mistakes if it is in that domain. For example, in C and C++ often a two dimensional matrix in array is really just a single dimensional array. And it is really easy to make mistakes to pass in array that have the wrong dimension in matrix multiplication during row major or column major order. C,C++ etc won’t give the support for matrix implementation that is being used, it will stay silent and compute the wrong answer. Matlab will catch those errors. Thirdly optimization, the higher the level of instruction given to an interpreter or a compiler the more is the scope for creativity. If the matrix multiplication is written in three for loops, then the compiler is forced to generate code for only that particular implementation DOMAIN SPECIFIC LANGUAGES allows to express things like matrix transposition and multiplication at a higher level, and thus end up generating better code for the new target architecture. So less time writing, concise, faster and more creativity

4.2.3 PIQL

Performance Insightful Query Languages. PIQL is designed specifically for the development of large-scale data-intensive web sites and applications. PIQL enables the creation of compelling applications without requiring developers to worry about scaling. The PIQL compiler takes in an application spec and produces a library jar that can be used from any JVM based language to interact with a cluster of Storage Nodes . [9] Entity Sets are typed collections of attributes, analogous to relations in a traditional relational database. An important distinction, however, is that PIQL allows only the atomic update of a single entity, instead of the general update mechanism present in SQL. Relationships are defined by specifying a foreign key that points to another entity type. This is different from SQL, [10] where the join predicates can be defined at query time. The reason for requiring this explicit specification is to allow the user to place cardinality constraints on specific relationships. These constraints can be specified in the form of a integer constant, 5000 in the above example.[11]

The following point shows how PIQL relates to SQL:

- It is based on a subset of SQL that enables accurate performance predication.
- All queries must specified ahead of time and validated against a developer-specified service level objective (SLO) .
- It has direct support for manipulating the entities and relationships that are commonly used in web applications .
- Its implementation is based on automatically selected indexes.
- It can be adapted to run on many existing key-value stores

PIQL, a Performance Insightful Query Language that allows developers to express many of the queries found on these websites, while still providing strict bounds on the number of I/O operations for any query.

CHAPTER 5

IMPLEMENTATION

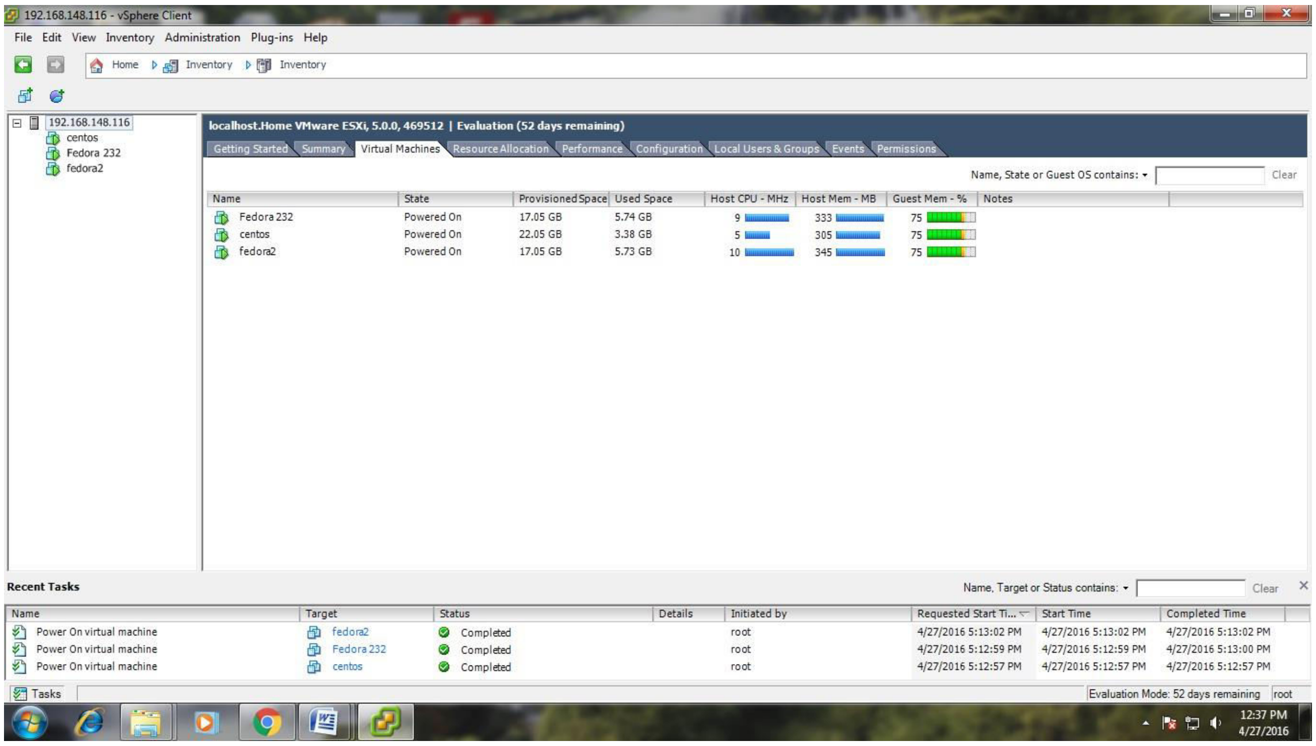
5.1 EXPERIMENTAL SETUP

In this chapter we will discuss about the experimental setup and implantation of the tool, and finally we will see the outputs generated by the tool. Our objective is to trace the lifecycle of a cloud operation across the log files throughout all the products involved in the disaster recovery. Firstly we need the entire disaster recovery set up. That is we need a protected site and a recovery site. In the protected side we will create virtual machines and we will replication them to the recovery site.

After the replication is done we will undergo other cloud operations like

- Planned Migration
- Test Failover
- Test Cleanup
- Reverse Replication
- Sync

For this we first load the ESXi, that is the hypervisor on a host, hypervisor is a software which is installed on bare metal. It virtualizes the resources such as storage, networking, and memory. After this we install a vSphere client on any machine which has network connectivity with that host, it is a client used to access the host. In vSphere we install VCenter server appliance as a plug-in, this plug-in is installed in two parts one part is installed in client and other part is installed in the host, as a virtual machine on the host. The VCenter server acts as a manager and manages the virtual machines on the host. This is the set up for the protected site. The following figure shows how it will eventually look like.


192.168.148.116 - vSphere Client
File Edit View Inventory Administration Plug-ins Help
Home Inventory Inventory
192.168.148.116
centos
Fedora 232
fedora2
Getting Started Summary Virtual Machines Resource Allocation Performance Configuration Local Users & Groups Events Permissions
Name, State or Guest OS contains: Clear
Name, Target or Status contains: Clear
Tasks
12:37 PM 4/27/2016

localhost.Home VMware ESXi, 5.0.0, 469512 | Evaluation (52 days remaining)

Name	State	Provisioned Space	Used Space	Host CPU - MHz	Host Mem - MB	Guest Mem - %	Notes
Fedora 232	Powered On	17.05 GB	5.74 GB	9	333	75	
centos	Powered On	22.05 GB	3.38 GB	5	305	75	
fedora2	Powered On	17.05 GB	5.73 GB	10	345	75	

Recent Tasks

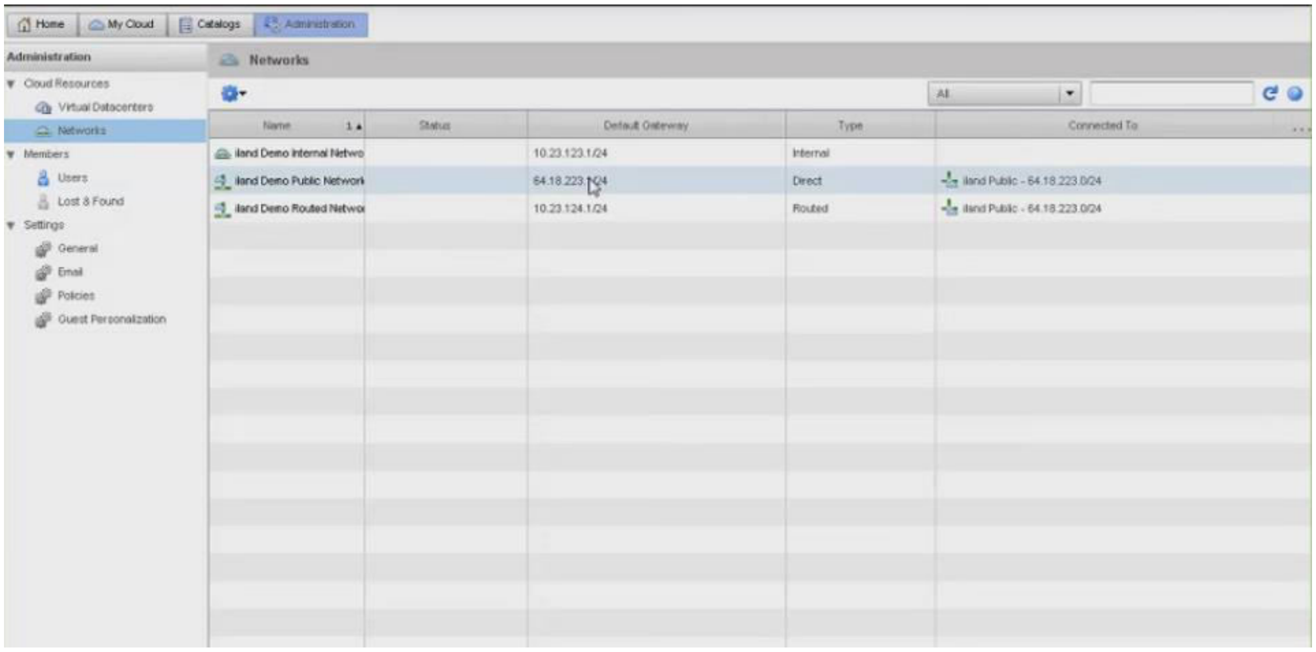
Name	Target	Status	Details	Initiated by	Requested Start Ti...	Start Time	Completed Time
Power On virtual machine	fedora2	Completed		root	4/27/2016 5:13:02 PM	4/27/2016 5:13:02 PM	4/27/2016 5:13:02 PM
Power On virtual machine	Fedora 232	Completed		root	4/27/2016 5:12:59 PM	4/27/2016 5:12:59 PM	4/27/2016 5:13:00 PM
Power On virtual machine	centos	Completed		root	4/27/2016 5:12:57 PM	4/27/2016 5:12:57 PM	4/27/2016 5:12:57 PM

Evaluation Mode: 52 days remaining root

Figure 8 In the above figure there are three virtual machines that are created namely centos,Fedora 232 and fedora2. is showing the statistics of those virtual machines

In the recovery site that is the site which is managed by the cloud organization, the setup is same, there is an ESXi which virtualizes the physical resources, they are VCenter and VSphere client to manage the resources, but since the organization has a large pool of resources, so there are significantly large number of hosts, so there are more and more number of VCenter. To manage these VCenter in a organizational cloud VCloud Director is installed on top. It has a number of API's which helps in larger resource abstraction and management. Mainly VCloud Director is needed in enterprise computing.

The figure below shows how a VCloud director looks like



Name	Status	Default Gateway	Type	Connected To
iland Demo Internal Netwo		10.23.123.1/24	Internal	
iland Demo Public Network		64.18.223.1/24	Direct	iland Public - 64.18.223.0/24
iland Demo Routed Netwo		10.23.124.1/24	Routed	iland Public - 64.18.223.0/24

Figure 9 VCloud Director Administrator UI

The VCloud Director Administrator UI looks like the above, on the right hand side there are the data centers or organizations of data centers which the user can provide as recovery site, in the above figure there are three of them, along with the status, gateway, type and whether the recovery site provided is connected to any protected site or not. The left hand column lists the Virtual datacenters, networking details setting etc.

Once this is done the Site recovery manager plug-in is installed in the protected side VCenter. After installation of the Site Recovery Manager the setup is ready. Virtual machines with different configurations such are created on the protected site, each virtual machine is selected, and then configured for replication. Once this is done the virtual machines are replicated on the recovery site and the setup is ready.

We will perform the operations on individual virtual machines, and collect the log files of the following products.

- ESXi in the protected site
- ESXi in the recovery site
- VCenter on the secondary site
- VCloudDirector on the secondary site
- VSphere Replication Server(hbrserv) on the secondary site
- VSphere Replication Management server(hms) on the primary site
- VSphere Replication management server(hms) on the secondary site

Once the log files are collected the logs are tagged and uploaded with proper manifest to the LogInsight Server. LogInsight Server is the database server on which the log files are queried with PIQL query language.

To configure and recover virtual machines protected by the Disaster Recovery service, perform the following tasks in vSphere Replication and VCloud Air:

1 Using vSphere Replication, we replicate the virtual machines from source site to VCloud Air site. Replication to the cloud is initiated by using vSphere Replication at your source site because replication between your source site and the cloud is not symmetrical, like it is when using VMware Site Recovery Manager. Replication is started from your source site but, for security reasons, there is no communication with the virtual machines at source site from the cloud.

2 After replicating your virtual machines to the cloud, VCloud Air is logged to view virtual data center enabled for disaster recovery and the placeholder virtual machines that you selected for replication.

3 Using vSphere Replication or VCloud Air, test recovery for a virtual machine and cleanup the test after execution.

4 In the event that your source site becomes unavailable, VCloud Air can be logged and virtual machines can be reconfigured.

5 Once source site is available, reverse replicate virtual machines from the VCloud Air Web UI.

LOG FILE COLLECTION FROM VARIOUS PRODUCTS

The Log File collection procedure of various products are given below.The Log files will be collected in a bundle using automated log bundle generator.

1 VSphere Replication Management Server.

To collect logs from the VRMS server:

1. Open the console for the VRMS server appliance and log in as the root user..
2. Confirm PermitRootLogin is set to yes in the SSH configuration file.

   To set the PermitRootLogin parameter to yes:

   a. Open the /etc/ssh/sshd_config file using a text editor.
   b. Locate the line that begins with PermitRootLogin.
   c. Set this parameter to yes.
   d. Save and close the file.
   e. Restart the sshd service using this command:
      service sshd restart

3. Start generating the support bundle using this command:
   /opt/VMware/hms/generatesupportbundle.sh
   In /opt/VMware/hms/bin directory instead of/opt/VMware/hms/
   The completed log bundle is located at:
   /opt/vmware/hms/support/HSB-hms-<uuid>/bundle.tar.gz
4. Connect to the VRMS server appliance using an SFTP client, such as FileZilla, and download the log bundle.

By default, the VRMS server logs (hms*.log) are located at /opt/vmware/hms/logs.

2 VSphere Replication Server

To collect logs from the VR server:

1. Open the console for the VR server appliance and log in as the root user.
2. Start generating the support bundle using this command:

   /usr/bin/hbrsrv-support-bundle.sh
   The completed log bundle is located in:
   /tmp/hbrsrv-2011-08-20-14-34-6.tgz
3. Connect to the VR server appliance using an SFTP client, such as FileZilla, and download the log bundle.
4. By default, the VR server logs (hbrsrv*.log) are located at /var/log/vmware.

3 ESXi and VCenter Server

To collect ESX/ESXi and VCenter Server diagnostic data:

1. Start the vSphere Web Client and log in to the VCenter Server system.
2. Under Inventory Lists, select VCenter Servers.
3. Click the VCenter Server that contains the ESX/ESXi hosts from which you want to export logs.
4. Click the Monitor tab and click System Logs.
5. Click Export System Logs.
6. Select the ESX/ESXi hosts from which you want to export logs.
7. Select the Include VCenter Server and vSphere Web Client logs option. This step is optional.
8. Click Next.
9. Select the system logs that are to be exported.
10. Select Gather performance data to include performance data information in the log files. Click Next.

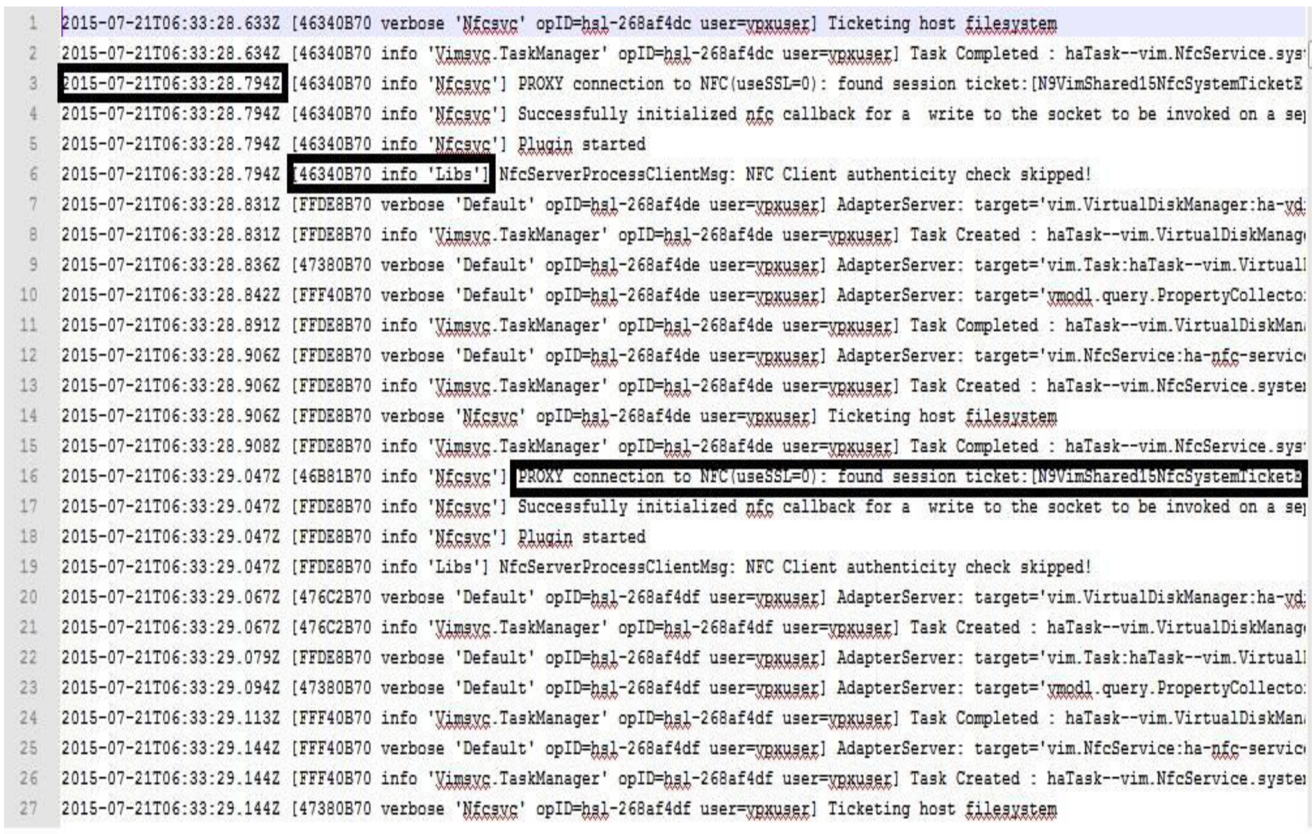
11. Click Generate Log Bundle. The Download Log Bundles dialog appears when the Generating Diagnostic Bundle task completes.
12. Click Download Log Bundle to save it to your local computer.
    The host or VCenter Server generates .zip bundles containing the log files.
13. After the download completes, click Finish or generate another log bundle.

4 VCloud Director

To collect the log files in VCloudDirector, open the terminal and run /opt/vmware/cloud-director/bin/vmware-vcd-support

The logs for VMware VCloud Director are located in /opt/vmware/vCloud-director/logs

Below figure show the log files collected from the ESXi host. These log files are tagged and then uploaded to the LogInsight server.

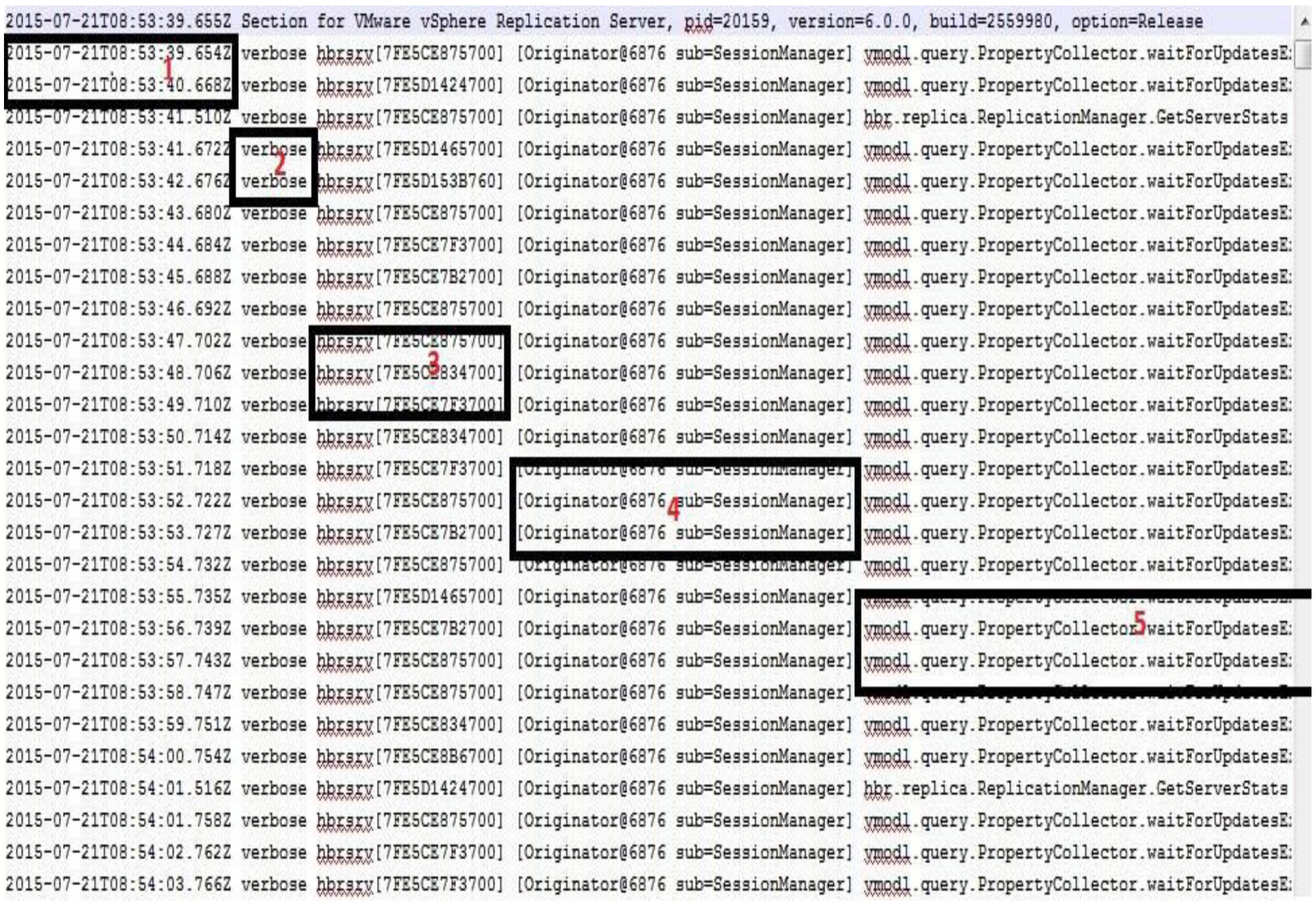


```
2015-07-21T06:33:28.633Z [46340B70 verbose 'Nfcsvc' opID=hsl-268af4dc user=vpxuser] Ticketing host filesystem
2015-07-21T06:33:28.634Z [46340B70 info 'Vimsvc.TaskManager' opID=hsl-268af4dc user=vpxuser] Task Completed : haTask--vim.NfcService.sys
2015-07-21T06:33:28.794Z [46340B70 info 'Nfcsvc'] PROXY connection to NFC(useSSL=0): found session ticket:[N9VimShared15NfcSystemTicketE
2015-07-21T06:33:28.794Z [46340B70 info 'Nfcsvc'] Successfully initialized nfc callback for a  write to the socket to be invoked on a se
2015-07-21T06:33:28.794Z [46340B70 info 'Nfcsvc'] Plugin started
2015-07-21T06:33:28.794Z [46340B70 info 'Libs'] NfcServerProcessClientMsg: NFC Client authenticity check skipped!
2015-07-21T06:33:28.831Z [FFDE8B70 verbose 'Default' opID=hsl-268af4de user=vpxuser] AdapterServer: target='vim.VirtualDiskManager:ha-vd
2015-07-21T06:33:28.831Z [FFDE8B70 info 'Vimsvc.TaskManager' opID=hsl-268af4de user=vpxuser] Task Created : haTask--vim.VirtualDiskManag
2015-07-21T06:33:28.836Z [47380B70 verbose 'Default' opID=hsl-268af4de user=vpxuser] AdapterServer: target='vim.Task:haTask--vim.Virtual
2015-07-21T06:33:28.842Z [FFF40B70 verbose 'Default' opID=hsl-268af4de user=vpxuser] AdapterServer: target='vmodl.query.PropertyCollecto
2015-07-21T06:33:28.891Z [FFDE8B70 info 'Vimsvc.TaskManager' opID=hsl-268af4de user=vpxuser] Task Completed : haTask--vim.VirtualDiskMan
2015-07-21T06:33:28.906Z [FFDE8B70 verbose 'Default' opID=hsl-268af4de user=vpxuser] AdapterServer: target='vim.NfcService:ha-nfc-servic
2015-07-21T06:33:28.906Z [FFDE8B70 info 'Vimsvc.TaskManager' opID=hsl-268af4de user=vpxuser] Task Created : haTask--vim.NfcService.syste
2015-07-21T06:33:28.906Z [FFDE8B70 verbose 'Nfcsvc' opID=hsl-268af4de user=vpxuser] Ticketing host filesystem
2015-07-21T06:33:28.908Z [FFDE8B70 info 'Vimsvc.TaskManager' opID=hsl-268af4de user=vpxuser] Task Completed : haTask--vim.NfcService.sys
2015-07-21T06:33:29.047Z [46B81B70 info 'Nfcsvc'] PROXY connection to NFC(useSSL=0): found session ticket:[N9VimShared15NfcSystemTicketE
2015-07-21T06:33:29.047Z [FFDE8B70 info 'Nfcsvc'] Successfully initialized nfc callback for a  write to the socket to be invoked on a se
2015-07-21T06:33:29.047Z [FFDE8B70 info 'Nfcsvc'] Plugin started
2015-07-21T06:33:29.047Z [FFDE8B70 info 'Libs'] NfcServerProcessClientMsg: NFC Client authenticity check skipped!
2015-07-21T06:33:29.067Z [476C2B70 verbose 'Default' opID=hsl-268af4df user=vpxuser] AdapterServer: target='vim.VirtualDiskManager:ha-vd
2015-07-21T06:33:29.067Z [476C2B70 info 'Vimsvc.TaskManager' opID=hsl-268af4df user=vpxuser] Task Created : haTask--vim.VirtualDiskManag
2015-07-21T06:33:29.079Z [FFDE8B70 verbose 'Default' opID=hsl-268af4df user=vpxuser] AdapterServer: target='vim.Task:haTask--vim.Virtual
2015-07-21T06:33:29.094Z [47380B70 verbose 'Default' opID=hsl-268af4df user=vpxuser] AdapterServer: target='vmodl.query.PropertyCollecto
2015-07-21T06:33:29.113Z [FFF40B70 info 'Vimsvc.TaskManager' opID=hsl-268af4df user=vpxuser] Task Completed : haTask--vim.VirtualDiskMan
2015-07-21T06:33:29.144Z [FFF40B70 verbose 'Default' opID=hsl-268af4df user=vpxuser] AdapterServer: target='vim.NfcService:ha-nfc-servic
2015-07-21T06:33:29.144Z [FFF40B70 info 'Vimsvc.TaskManager' opID=hsl-268af4df user=vpxuser] Task Created : haTask--vim.NfcService.syste
2015-07-21T06:33:29.144Z [47380B70 verbose 'Nfcsvc' opID=hsl-268af4df user=vpxuser] Ticketing host filesystem
```

Figure 10 log file of ESXi

In the above figure the log file of ESXi is shown, the top leftmost box is the timestamp the format of the timestamp in the log is of the form YYYY-MM-DDTHH:MM:SS.sssZ, well this is the format of the timestamp in the log files, there are different format of timestamps for different products, tagging need to be done accordingly, the second box in the middle just below and immediate right of the timestamp describing box ,describes an element and usually given between the [15] brackets, the third box is to the bottom right is records the log message. The tagging needs to be done appropriately while uploading the logs to the data base server.

The below figure shows the log files collected from VSphere replication Server



```
2015-07-21T08:53:39.655Z Section for VMware vSphere Replication Server, pid=20159, version=6.0.0, build=2559980, option=Release
2015-07-21T08:53:39.654Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:40.668Z verbose hbrsrv[7FE5D1424700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:41.510Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] hbr.replica.ReplicationManager.GetServerStats
2015-07-21T08:53:41.672Z verbose hbrsrv[7FE5D1465700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:42.676Z verbose hbrsrv[7FE5D153B760] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:43.680Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:44.684Z verbose hbrsrv[7FE5CE7F3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:45.688Z verbose hbrsrv[7FE5CE7B2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:46.692Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:47.702Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:48.706Z verbose hbrsrv[7FE5CE834700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:49.710Z verbose hbrsrv[7FE5CE7F3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:50.714Z verbose hbrsrv[7FE5CE834700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:51.718Z verbose hbrsrv[7FE5CE7F3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:52.722Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:53.727Z verbose hbrsrv[7FE5CE7B2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:54.732Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:55.735Z verbose hbrsrv[7FE5D1465700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:56.739Z verbose hbrsrv[7FE5CE7B2700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:57.743Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:58.747Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:53:59.751Z verbose hbrsrv[7FE5CE834700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:54:00.754Z verbose hbrsrv[7FE5CE8B6700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:54:01.516Z verbose hbrsrv[7FE5D1424700] [Originator@6876 sub=SessionManager] hbr.replica.ReplicationManager.GetServerStats
2015-07-21T08:54:01.758Z verbose hbrsrv[7FE5CE875700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:54:02.762Z verbose hbrsrv[7FE5CE7F3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
2015-07-21T08:54:03.766Z verbose hbrsrv[7FE5CE7F3700] [Originator@6876 sub=SessionManager] vmodl.query.PropertyCollector.waitForUpdatesE
```

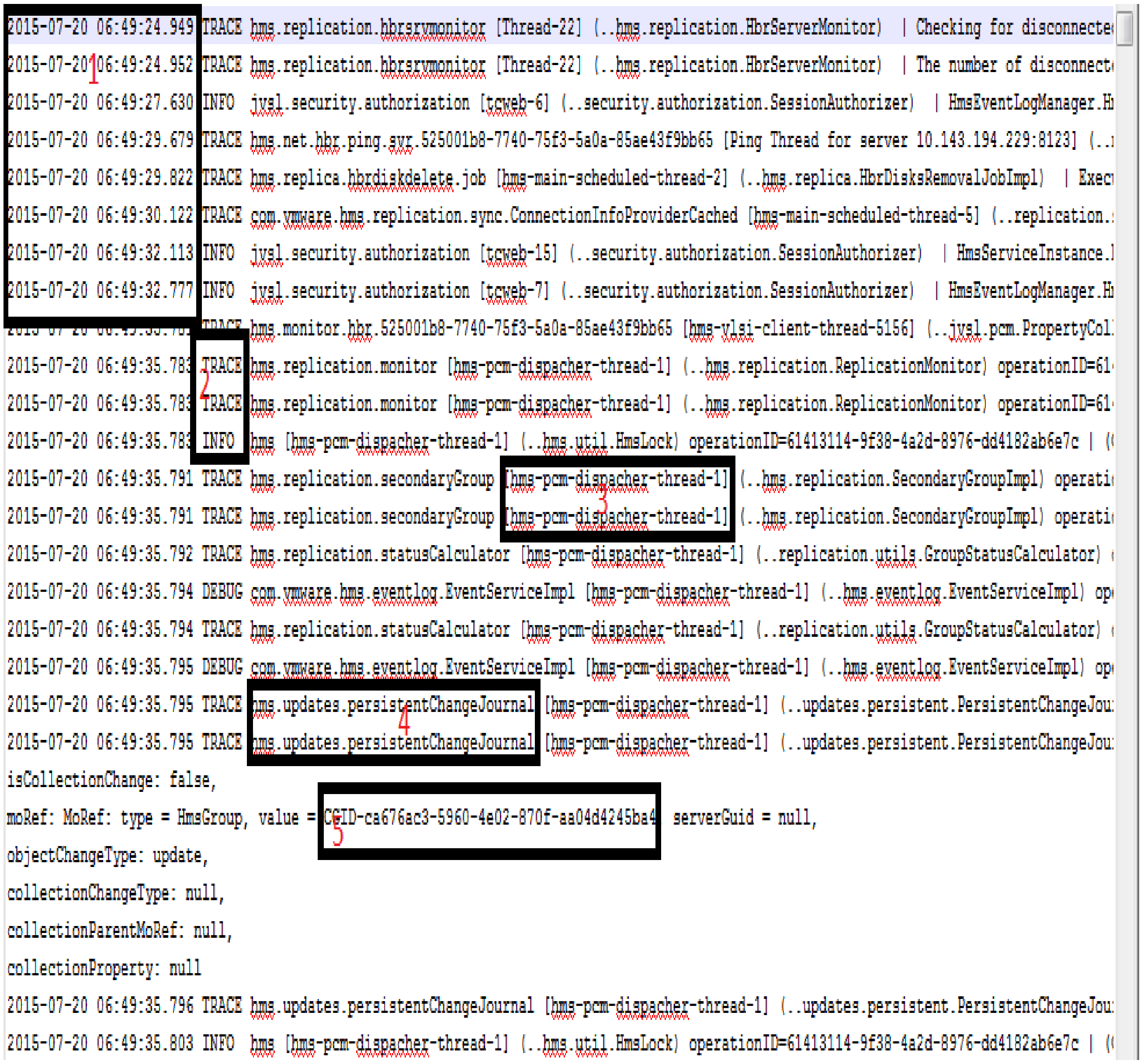
Figure 11 log file collected from a VSphere Replication Server

The figure above shows the log file collected from a VSphere Replication Server

In the figure, 1.Indicates the timestamp, the format of the timestamp is the same,2.Indicates the type of the message recorded in the log line,3.Indicates an element, the first part is hbrsrv followed by a code in square brackets,4.Indicates another element ,5.Is the message of the log.

The log files is tagged accordingly and uploaded in the LogInsight server.

Given below is the figure of Log file of the VSphere Replication Management server

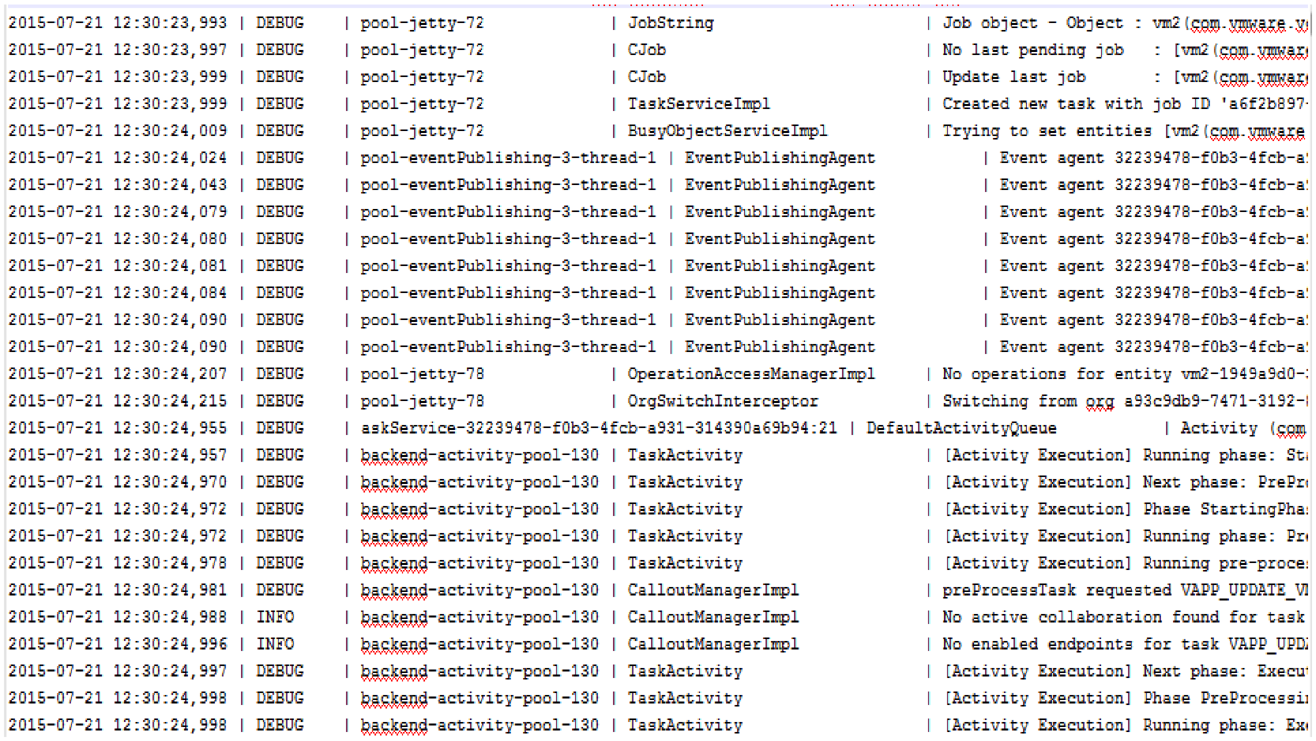


```
2015-07-20 06:49:24.949 TRACE hms.replication.hbrsrvmonitor [Thread-22] (..hms.replication.HbrServerMonitor)  | Checking for disconnecte
2015-07-20 06:49:24.952 TRACE hms.replication.hbrsrvmonitor [Thread-22] (..hms.replication.HbrServerMonitor)  | The number of disconnect
2015-07-20 06:49:27.630 INFO  jvsl.security.authorization [tcweb-6] (..security.authorization.SessionAuthorizer)  | HmsEventLogManager.H
2015-07-20 06:49:29.679 TRACE hms.net.hbr.ping.svr.525001b8-7740-75f3-5a0a-85ae43f9bb65 [Ping Thread for server 10.143.194.229:8123] (..
2015-07-20 06:49:29.822 TRACE hms.replica.hbrdiskdelete.job [hms-main-scheduled-thread-2] (..hms.replica.HbrDisksRemovalJobImpl)  | Exec
2015-07-20 06:49:30.122 TRACE com.vmware.hms.replication.sync.ConnectionInfoProviderCached [hms-main-scheduled-thread-5] (..replication.
2015-07-20 06:49:32.113 INFO  jvsl.security.authorization [tcweb-15] (..security.authorization.SessionAuthorizer)  | HmsServiceInstance.
2015-07-20 06:49:32.777 INFO  jvsl.security.authorization [tcweb-7] (..security.authorization.SessionAuthorizer)  | HmsEventLogManager.H
2015-07-20 06:49:35.781 TRACE hms.monitor.hbr.525001b8-7740-75f3-5a0a-85ae43f9bb65 [hms-vlsi-client-thread-5156] (..jvsl.pcm.PropertyCol
2015-07-20 06:49:35.783 TRACE hms.replication.monitor [hms-pcm-dispacher-thread-1] (..hms.replication.ReplicationMonitor) operationID=61
2015-07-20 06:49:35.783 TRACE hms.replication.monitor [hms-pcm-dispacher-thread-1] (..hms.replication.ReplicationMonitor) operationID=61
2015-07-20 06:49:35.783 INFO  hms [hms-pcm-dispacher-thread-1] (..hms.util.HmsLock) operationID=61413114-9f38-4a2d-8976-dd4182ab6e7c | (
2015-07-20 06:49:35.791 TRACE hms.replication.secondaryGroup [hms-pcm-dispacher-thread-1] (..hms.replication.SecondaryGroupImpl) operati
2015-07-20 06:49:35.791 TRACE hms.replication.secondaryGroup [hms-pcm-dispacher-thread-1] (..hms.replication.SecondaryGroupImpl) operati
2015-07-20 06:49:35.792 TRACE hms.replication.statusCalculator [hms-pcm-dispacher-thread-1] (..replication.utils.GroupStatusCalculator)
2015-07-20 06:49:35.794 DEBUG com.vmware.hms.eventlog.EventServiceImpl [hms-pcm-dispacher-thread-1] (..hms.eventlog.EventServiceImpl) op
2015-07-20 06:49:35.794 TRACE hms.replication.statusCalculator [hms-pcm-dispacher-thread-1] (..replication.utils.GroupStatusCalculator)
2015-07-20 06:49:35.795 DEBUG com.vmware.hms.eventlog.EventServiceImpl [hms-pcm-dispacher-thread-1] (..hms.eventlog.EventServiceImpl) op
2015-07-20 06:49:35.795 TRACE hms.updates.persistentChangeJournal [hms-pcm-dispacher-thread-1] (..updates.persistent.PersistentChangeJou
2015-07-20 06:49:35.795 TRACE hms.updates.persistentChangeJournal [hms-pcm-dispacher-thread-1] (..updates.persistent.PersistentChangeJou
isCollectionChange: false,
moRef: MoRef: type = HmsGroup, value = CGID-ca676ac3-5960-4e02-870f-aa04d4245ba4  serverGuid = null,
objectChangeType: update,
collectionChangeType: null,
collectionParentMoRef: null,
collectionProperty: null
2015-07-20 06:49:35.796 TRACE hms.updates.persistentChangeJournal [hms-pcm-dispacher-thread-1] (..updates.persistent.PersistentChangeJou
2015-07-20 06:49:35.803 INFO  hms [hms-pcm-dispacher-thread-1] (..hms.util.HmsLock) operationID=61413114-9f38-4a2d-8976-dd4182ab6e7c | (
```

Figure 12 log file of a VSphere Replication Management Server

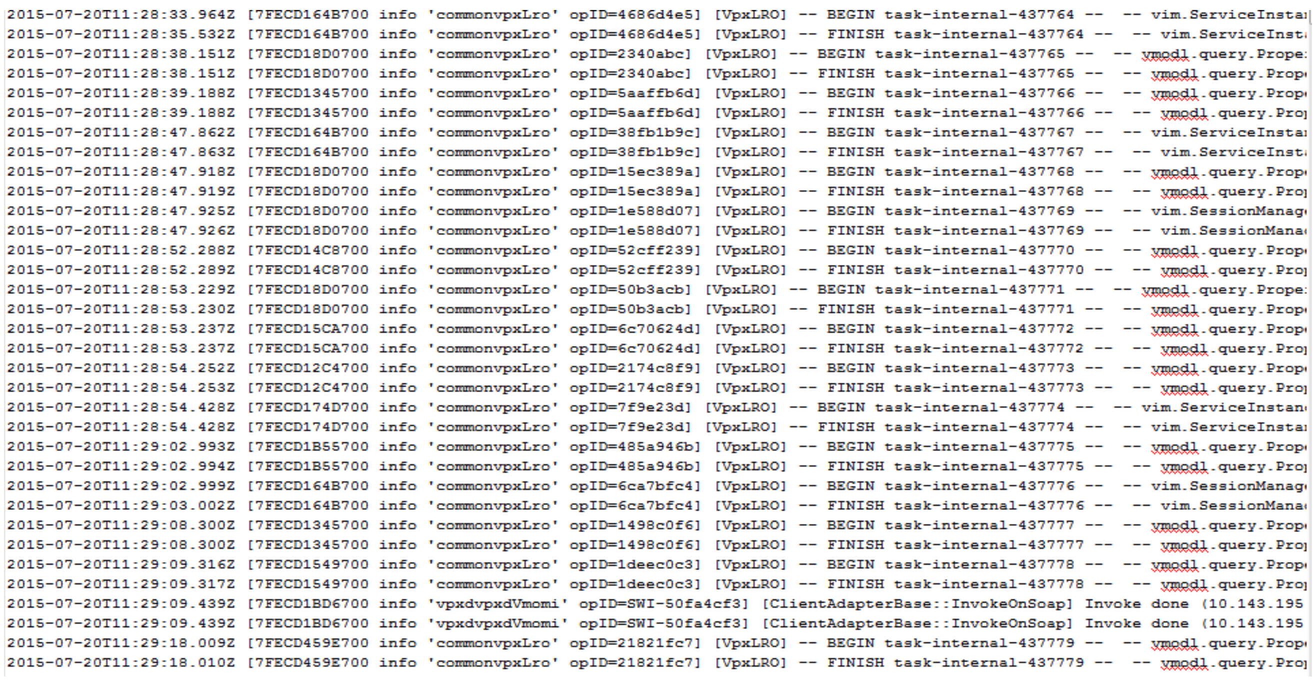
The above figure describe the log file of a VSphere Replication Management Server. In the above figure 1. Indicates the timestamp, notice the format of the timestamp is different here it is YYYY-MM-DD HH:MM:SS:sss, 2. Indicates the type of message recorded in the log line, 3. Indicates an element which stores inside it the thread name, 4. Indicates the class called, 5. Here is the CGID Common Group identification. This id will be used later for analysis

Next we will see the log files of VCloud Director and VCenter server



```
2015-07-21 12:30:23,993 | DEBUG    | pool-jetty-72             | JobString                | Job object - Object : vm2(com.vmware.v
2015-07-21 12:30:23,997 | DEBUG    | pool-jetty-72             | CJob                     | No last pending job   : [vm2(com.vmwar
2015-07-21 12:30:23,999 | DEBUG    | pool-jetty-72             | CJob                     | Update last job       : [vm2(com.vmwar
2015-07-21 12:30:23,999 | DEBUG    | pool-jetty-72             | TaskServiceImpl          | Created new task with job ID 'a6f2b897
2015-07-21 12:30:24,009 | DEBUG    | pool-jetty-72             | BusyObjectServiceImpl    | Trying to set entities [vm2(com.vmware
2015-07-21 12:30:24,024 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,043 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,079 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,080 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,081 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,084 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,090 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,090 | DEBUG    | pool-eventPublishing-3-thread-1 | EventPublishingAgent     | Event agent 32239478-f0b3-4fcb-a
2015-07-21 12:30:24,207 | DEBUG    | pool-jetty-78             | OperationAccessManagerImpl | No operations for entity vm2-1949a9d0-
2015-07-21 12:30:24,215 | DEBUG    | pool-jetty-78             | OrgSwitchInterceptor     | Switching from org a93c9db9-7471-3192-
2015-07-21 12:30:24,955 | DEBUG    | askService-32239478-f0b3-4fcb-a931-314390a69b94:21 | DefaultActivityQueue     | Activity (com
2015-07-21 12:30:24,957 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Running phase: St
2015-07-21 12:30:24,970 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Next phase: PrePr
2015-07-21 12:30:24,972 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Phase StartingPha
2015-07-21 12:30:24,972 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Running phase: Pr
2015-07-21 12:30:24,978 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Running pre-proce
2015-07-21 12:30:24,981 | DEBUG    | backend-activity-pool-130 | CalloutManagerImpl       | preProcessTask requested VAPP_UPDATE_V
2015-07-21 12:30:24,988 | INFO     | backend-activity-pool-130 | CalloutManagerImpl       | No active collaboration found for task
2015-07-21 12:30:24,996 | INFO     | backend-activity-pool-130 | CalloutManagerImpl       | No enabled endpoints for task VAPP_UPD
2015-07-21 12:30:24,997 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Next phase: Execu
2015-07-21 12:30:24,998 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Phase PreProcessi
2015-07-21 12:30:24,998 | DEBUG    | backend-activity-pool-130 | TaskActivity             | [Activity Execution] Running phase: Ex
```

Figure 13 The above figure is a portion within the log file generated by a VCloud Director



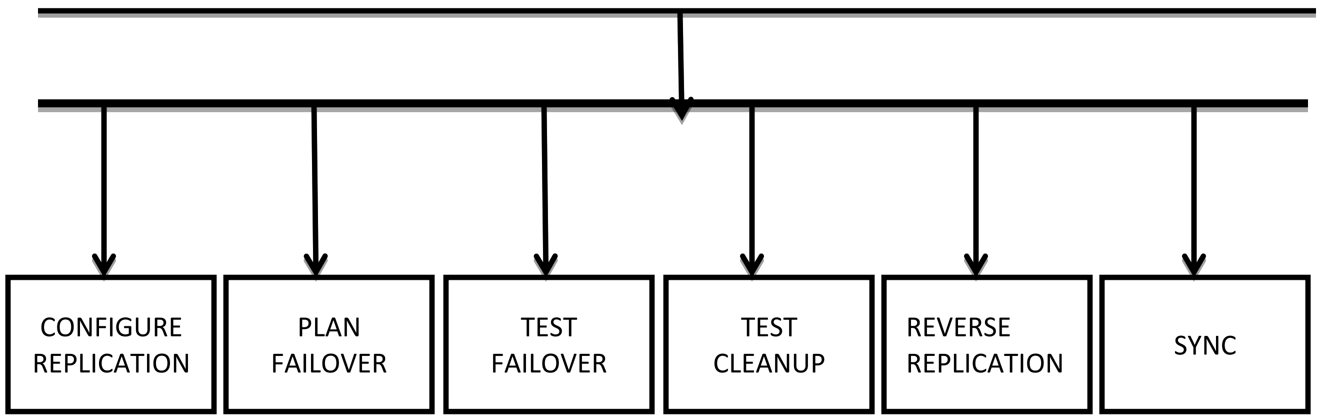
```
2015-07-20T11:28:33.964Z [7FECD164B700 info 'commonvpxLro' opID=4686d4e5] [VpxLRO] -- BEGIN task-internal-437764 --  -- vim.ServiceInsta
2015-07-20T11:28:35.532Z [7FECD164B700 info 'commonvpxLro' opID=4686d4e5] [VpxLRO] -- FINISH task-internal-437764 --  -- vim.ServiceInst
2015-07-20T11:28:38.151Z [7FECD18D0700 info 'commonvpxLro' opID=2340abc] [VpxLRO] -- BEGIN task-internal-437765 --  -- vmodl.query.Prope
2015-07-20T11:28:38.151Z [7FECD18D0700 info 'commonvpxLro' opID=2340abc] [VpxLRO] -- FINISH task-internal-437765 --  -- vmodl.query.Prop
2015-07-20T11:28:39.188Z [7FECD1345700 info 'commonvpxLro' opID=5aaffb6d] [VpxLRO] -- BEGIN task-internal-437766 --  -- vmodl.query.Prop
2015-07-20T11:28:39.188Z [7FECD1345700 info 'commonvpxLro' opID=5aaffb6d] [VpxLRO] -- FINISH task-internal-437766 --  -- vmodl.query.Pro
2015-07-20T11:28:47.862Z [7FECD164B700 info 'commonvpxLro' opID=38fb1b9c] [VpxLRO] -- BEGIN task-internal-437767 --  -- vim.ServiceInsta
2015-07-20T11:28:47.863Z [7FECD164B700 info 'commonvpxLro' opID=38fb1b9c] [VpxLRO] -- FINISH task-internal-437767 --  -- vim.ServiceInst
2015-07-20T11:28:47.918Z [7FECD18D0700 info 'commonvpxLro' opID=15ec389a] [VpxLRO] -- BEGIN task-internal-437768 --  -- vmodl.query.Prop
2015-07-20T11:28:47.919Z [7FECD18D0700 info 'commonvpxLro' opID=15ec389a] [VpxLRO] -- FINISH task-internal-437768 --  -- vmodl.query.Pro
2015-07-20T11:28:47.925Z [7FECD18D0700 info 'commonvpxLro' opID=1e588d07] [VpxLRO] -- BEGIN task-internal-437769 --  -- vim.SessionManag
2015-07-20T11:28:47.926Z [7FECD18D0700 info 'commonvpxLro' opID=1e588d07] [VpxLRO] -- FINISH task-internal-437769 --  -- vim.SessionMana
2015-07-20T11:28:52.288Z [7FECD14C8700 info 'commonvpxLro' opID=52cff239] [VpxLRO] -- BEGIN task-internal-437770 --  -- vmodl.query.Prop
2015-07-20T11:28:52.289Z [7FECD14C8700 info 'commonvpxLro' opID=52cff239] [VpxLRO] -- FINISH task-internal-437770 --  -- vmodl.query.Pro
2015-07-20T11:28:53.229Z [7FECD18D0700 info 'commonvpxLro' opID=50b3acb] [VpxLRO] -- BEGIN task-internal-437771 --  -- vmodl.query.Prope
2015-07-20T11:28:53.230Z [7FECD18D0700 info 'commonvpxLro' opID=50b3acb] [VpxLRO] -- FINISH task-internal-437771 --  -- vmodl.query.Prop
2015-07-20T11:28:53.237Z [7FECD15CA700 info 'commonvpxLro' opID=6c70624d] [VpxLRO] -- BEGIN task-internal-437772 --  -- vmodl.query.Prop
2015-07-20T11:28:53.237Z [7FECD15CA700 info 'commonvpxLro' opID=6c70624d] [VpxLRO] -- FINISH task-internal-437772 --  -- vmodl.query.Pro
2015-07-20T11:28:54.252Z [7FECD12C4700 info 'commonvpxLro' opID=2174c8f9] [VpxLRO] -- BEGIN task-internal-437773 --  -- vmodl.query.Prop
2015-07-20T11:28:54.253Z [7FECD12C4700 info 'commonvpxLro' opID=2174c8f9] [VpxLRO] -- FINISH task-internal-437773 --  -- vmodl.query.Pro
2015-07-20T11:28:54.428Z [7FECD174D700 info 'commonvpxLro' opID=7f9e23d] [VpxLRO] -- BEGIN task-internal-437774 --  -- vim.ServiceInstan
2015-07-20T11:28:54.428Z [7FECD174D700 info 'commonvpxLro' opID=7f9e23d] [VpxLRO] -- FINISH task-internal-437774 --  -- vim.ServiceInsta
2015-07-20T11:29:02.993Z [7FECD1B55700 info 'commonvpxLro' opID=485a946b] [VpxLRO] -- BEGIN task-internal-437775 --  -- vmodl.query.Prop
2015-07-20T11:29:02.994Z [7FECD1B55700 info 'commonvpxLro' opID=485a946b] [VpxLRO] -- FINISH task-internal-437775 --  -- vmodl.query.Pro
2015-07-20T11:29:02.999Z [7FECD164B700 info 'commonvpxLro' opID=6ca7bfc4] [VpxLRO] -- BEGIN task-internal-437776 --  -- vim.SessionManag
2015-07-20T11:29:03.002Z [7FECD164B700 info 'commonvpxLro' opID=6ca7bfc4] [VpxLRO] -- FINISH task-internal-437776 --  -- vim.SessionMana
2015-07-20T11:29:08.300Z [7FECD1345700 info 'commonvpxLro' opID=1498c0f6] [VpxLRO] -- BEGIN task-internal-437777 --  -- vmodl.query.Prop
2015-07-20T11:29:08.300Z [7FECD1345700 info 'commonvpxLro' opID=1498c0f6] [VpxLRO] -- FINISH task-internal-437777 --  -- vmodl.query.Pro
2015-07-20T11:29:09.316Z [7FECD1549700 info 'commonvpxLro' opID=1deec0c3] [VpxLRO] -- BEGIN task-internal-437778 --  -- vmodl.query.Prop
2015-07-20T11:29:09.317Z [7FECD1549700 info 'commonvpxLro' opID=1deec0c3] [VpxLRO] -- FINISH task-internal-437778 --  -- vmodl.query.Pro
2015-07-20T11:29:09.439Z [7FECD1BD6700 info 'vpxdvpxdVmomi' opID=SWI-50fa4cf3] [ClientAdapterBase::InvokeOnSoap] Invoke done (10.143.195
2015-07-20T11:29:09.439Z [7FECD1BD6700 info 'vpxdvpxdVmomi' opID=SWI-50fa4cf3] [ClientAdapterBase::InvokeOnSoap] Invoke done (10.143.195
2015-07-20T11:29:18.009Z [7FECD459E700 info 'commonvpxLro' opID=21821fc7] [VpxLRO] -- BEGIN task-internal-437779 --  -- vmodl.query.Prop
2015-07-20T11:29:18.010Z [7FECD459E700 info 'commonvpxLro' opID=21821fc7] [VpxLRO] -- FINISH task-internal-437779 --  -- vmodl.query.Pro
```

Figure 14 The above figure is a portion of log file generated by a VCenter Server

5.2 ALGORITHM

After uploading the log files to the LogInsight server, coding in Domain Specific languages is done, through these codes the pattern is searched in the log files which is unique to a particular operation, and the entire operation performed in the virtual machine is traced. In the following sections will describe the implementation based on the operations, the coding is done in operation specific way so that the tool can be kept as much modular as possible. This is required so that it can be changed later if and when required. In the Domain Specific Languages the elements within the angular brackets are treated as variables, the Domain here is pattern searching in log file. The initial DOMAIN SPECIFIC LANGUAGES will be the same for all that is the home DOMAIN SPECIFIC LANGUAGES

```
INPUT:REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH FIVE COLUMNS,CLICKING EACH OF THEM WILL LEAD
1. CREATE FIVE VARIABLES PLANFAILOVER,TESTFAILOVER,CONFIGURE
REPLICATION,SYNC,REVERSE REPLICATION
2. STORE A STRING IN EACH OF THEM AND MAKE THEM HYPERLINK
TO THE RESPECTIVE OPERATIONS.
```



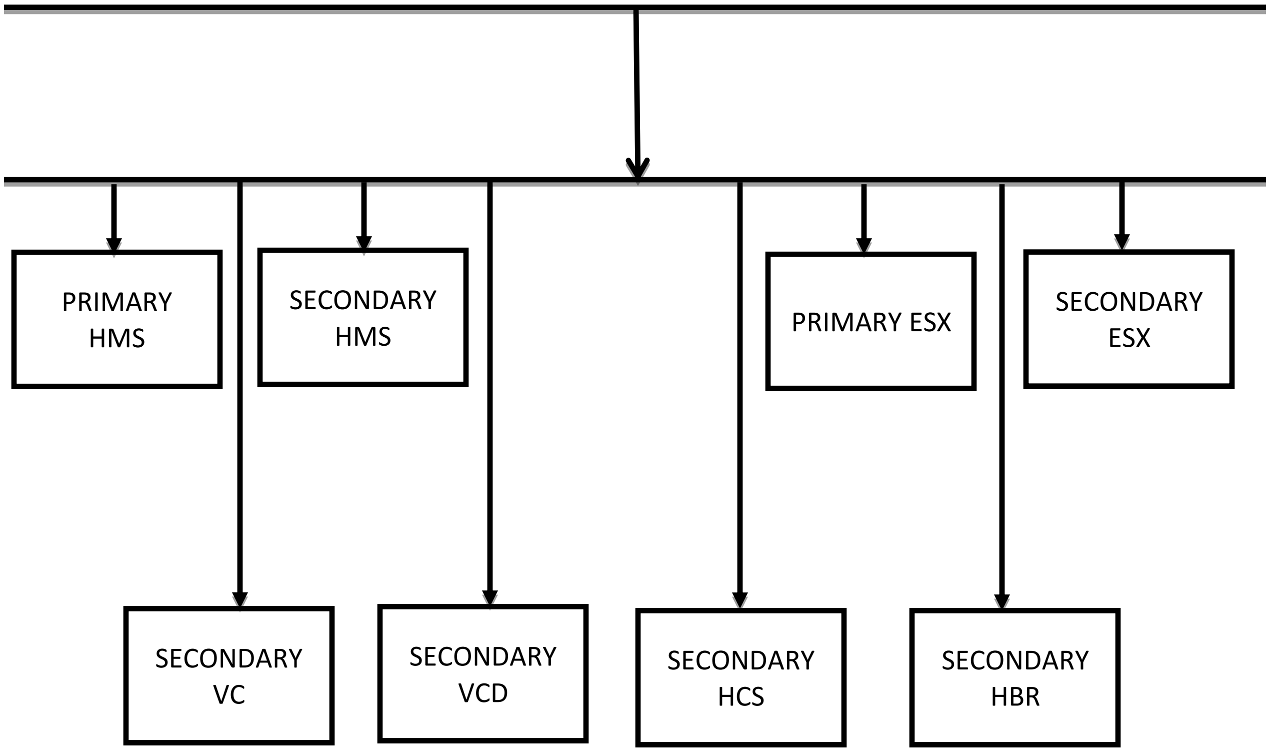
TRACING THE CONFIGURE REPLICATION OPERATION

```
INPUT:REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND CONFIGURE REPLICATION
1. CREATE PATTERN ’hms.job [<<event>>] (<<class_name>>) '+'operationID=<<op_id>> | <<msg>>$'
2.GET THE LOG LINES IN THE DATABASE WHICH CONTAINS THE GIVEN PATTERN AND STORE
THEN IN A TABLE
3.FOR EACH LINE IN THE TABLE,
        3.1CREATE VARIABLES
                line WHICH STORES LOGLINE MESSAGE
                opidWHICH STORES LOGLINE OPID
                op1 WHICH STORES STRING 'Creating replications for group'
        3.2IF line CONTAINS op1
                3.2.1CREATE PATTERN 'hms [<<thread_name>>] (<<class_name>>) operationID=' + opid
                + ' | (<<cutit>>:run(<<CGID>>), elapsed: <<time>>) obtains the lock'
                3.2.2SEARCH THE DATABASE WITH THE PATTERN AND STORE IT IN A NEW
                TABLE
                3.2.3FOR EACH LINE IN THIS NEW TABLE FETCH cgid AND vmname
                3.2.4CREATE A LIST WITH vmname,cgid AND opid
                3.2.5PASS THIS LIST TO SUBSEQUENT DOMAIN SPECIFIC LANGUAGES WITH THE
                NAME CONFIGURE REPLICATION
        ENDIF
END FOR
```

```
INPUT:REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH FOUR
COLUMNSTARTTIME,ENDTIME,STATUS,TRACESTACK
1.STORE THE ELEMENTS RECEIVED BY THE LIST PASSED INTO INDIVIDUAL VARIABLES
vmName ,opid .cgid
2.CREATE PATTERN '[<<event>>] (<<class_name>>) operationID='+opid+' | Creating  Configure
replication to cloud  task'
3.SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
NOW TRAVERSE THE TABLE AND STORE THE TIMESTAMP OF THE LOGLINES IN A VARIABLE
startTime
4.CREATE PATTERN '[<<event>>] (<<class_name>>) operationID='+opid+' | State update succeeded for
task <<HTID>>. State = <<msg2>>$' from 'hms-primary'
5.SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
6.TRAVERSE THE TABLE
        6.1 IF THE LOGLINE MESSAGE CONTAINS SUCCESS
                THEN STATUS = SUCCESS
                STORE THE TIMESTAMP IN A VARIABLE endtime
        6.2 IF THE LOGLINE MESSAGE CONTAINS ERROR
                THEN STATUS = ERROR
                STORE THE TIMESTAMP IN A VARIABLE endtime
                ADD ALERT
        END IF
END TRAVERSE
7.CREATE A LIST Trace_Stack WITH VARIABLES startTime, endTime, opid, cgid, vmName
8. PASS THIS LIST TO SUBSEQUENT DOMAIN SPECIFIC LANGUAGES WITH THE NAME
CONFIGURE REPLICATION PRODUCT
```

```
INPUT:REQUIRES THE HMS-PRIMARY LOGFILE
```

```
OUTPUT: A TABLE IS CREATED WITH EIGHT COLUMNS IS CREATEDVARIABLES
PRIMARY_HMS ,SECONDARY_HMS  ,SECONDARY_VCD ,SECONDARY_VC ,PRIMARY_ESX
,SECONDARY_ESX ,PRIMARY_HBR, SECONDARY_HBR
1. STORE THE ELEMENTS RECEIVED BY THE LIST PASSED INTO INDIVIDUAL VARIABLES
startTime , endTime
,opid, cgid , vmName
2.CREATE VARIABLES
primary_hms,secondary_hms,secondary_vcd,secondary_vc,primary_esx,secondary_esx,primary_hbr AND
STORE opid IN THEM
3.CREATE LIST secondary_hbr AND STORE opid, startTime, endTime, cgid IN IT
4.PASS THE VARIABLES AND THE LIST TO SUBSEQUENT DOMAIN SPECIFIC LANGUAGES
```



PRIMARY ESX-HYPERVISOR ON SITE 1

```
INPUT:REQUIRES THE ESX-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH THREE COLUMNS MSG , MSG2 AND TEST2
  1. CREATE PATTERN primary_esx+""+'-*'
  2. .SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
     TABLE
  3. TRAVERSE THE TABLE
         3.1CREATE A VARIABLE MSG WHICH STORE logLine.vdiag_caseid
         3.2CREATE A VARIABLE MSG2 WHICH STORE NULL
         3.3IF (logLine.vmw_esxi_vmfs_volume != NULL)
             THEN
                 CONCATENATE MSG2 WITH  logLine.vmw_esxi_vmfs_volume
             ENDIF
         3.4IF (logLine.vmdt_hostd_task_name != NULL)
```

```
          THEN
                    CONCATENATE MSG2 WITH  logLine.vmdt_hostd_task_name
          ENDIF

3.5IF (logLine.dr2c_hostd_hbrTaskName_1!= NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_hbrTaskName_1
          ENDIF
3.6IF (logLine.vmw_esxi_vm_name != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.vmw_esxi_vm_name
          ENDIF
3.7IF (logLine.vmdt_hostd_event_name!= NULL)
          THEN
                    CONCATENATE MSG2 WITHlogLine.vmdt_hostd_event_name
          ENDIF
3.8IF (logLine.dr2c_hostd_sync_stask_tatus != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_sync_stask_tatus
          ENDIF
3.9IF (logLine.dr2c_hostd_sync_initiator_1 != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_sync_initiator_1
          ENDIF
3.10IF (logLine.dr2c_hostd_vmName_1 != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_vmName_1
          ENDIF
3.11IF (logLine.dr2c_hostd_vmReplicationID_1 != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_vmReplicationID_1
          ENDIF
3.12IF (logLine.dr2c_hostd_delta_instanceID_1 != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.dr2c_hostd_delta_instanceID_1
          ENDIF
3.13IF (logLine.vmw_vc_task_method != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.vmw_vc_task_method
          ENDIF
3.14IF (logLine.vmw_task_status!= NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.vmw_task_status
          ENDIF
3.15IF (logLine.vmw_vc_taskid != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.vmw_vc_taskid
          ENDIF
3.16IF (logLine.view_broker_createnewvm != NULL)
          THEN
                    CONCATENATE MSG2 WITH  logLine.view_broker_createnewvm
```

```
            ENDIF
        3.17IF (logLine.plog_error_status!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.plog_error_status
            ENDIF
        3.18IF (logLine.hbr_msg!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.hbr_msg
            ENDIF
    END TRAVERSE
```

SECONDARY ESX-HYPERVISOR ON SITE 2

```
INPUT:REQUIRES THE ESX-SECONDARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH THREE COLUMNS MSG , MSG2 AND TEST2
  1. CREATE A CLOSURE
     1.1secondary_esx ->defretv = secondary_esx +'.+'.toString()
     1.2 retv.getMetaClass().is_regex = true
     1.3 retv.getMetaClass().column = 'text'
     1.4RETURN retv

  2. CREATE A VARIBLE test2 STORE THE VALUE RETURED BY THE CLOSURE IN THE
     VARIABLE
  3. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
     TABLE
  4. TRAVERSE THE TABLE
        4.1CREATE A VARIABLE MSG WHICH STORE logLine.vdiag_caseid
        4.2CREATE A VARIABLE MSG2 WHICH STORE NULL
        4.3IF (logLine.vsan_vsantrace_traceid != NULL)
            THEN
                CONCATENATE MSG2 logLine.vsan_vsantrace_traceid
            ENDIF
        4.4IF (logLine.vmw_vc_task_methodvmw_vc_task_method != NULL)
            THEN
                CONCATENATE MSG2 WITH
        logLine.vmw_vc_task_methodvmw_vc_task_method
            ENDIF

        4.5IF (logLine.dr2c_hostd_hbrTaskName_1!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.dr2c_hostd_hbrTaskName_1
            ENDIF
        4.6IF (logLine.vmw_esxi_vm_name != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.vmw_esxi_vm_name
            ENDIF
        4.7IF (logLine.vmw_task_status!= NULL)
            THEN
                CONCATENATE MSG2 WITHlogLine.vmw_task_status
            ENDIF
        4.8IF (logLine.vsan_vsantrace_role != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.vsan_vsantrace_role
            ENDIF
        4.9IF (logLine.dr2c_hostd_sync_initiator_1 != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.dr2c_hostd_sync_initiator_1
            ENDIF
        4.10IF (logLine.dr2c_hostd_vmName_1 != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.dr2c_hostd_vmName_1
            ENDIF
        4.11IF (logLine.dr2c_hostd_vmReplicationID_1 != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.dr2c_hostd_vmReplicationID_1
            ENDIF
        4.12IF (logLine.dr2c_hostd_delta_instanceID_1 != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.dr2c_hostd_delta_instanceID_1
            ENDIF
        4.13IF (logLine.vmw_vc_task_method != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.vmw_vc_task_method
            ENDIF
        4.14IF (logLine.vmw_task_status!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.vmw_task_status
            ENDIF
        4.15IF (logLine.vmw_vc_taskid != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.vmw_vc_taskid
            ENDIF
        4.16IF (logLine.view_broker_createnewvm != NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.view_broker_createnewvm
            ENDIF
        4.17IF (logLine.plog_error_status!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.plog_error_status
            ENDIF
        4.18IF (logLine.hbr_msg!= NULL)
            THEN
                CONCATENATE MSG2 WITH  logLine.hbr_msg
            ENDIF
     END TRAVERSE
```

SECONDARY HBR-VSPHERE REPLICATION SERVER ON SITE 2

INPUT:REQUIRES THE HBR-SECONDARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIME , PTIME AND MSG

```
1. STORE THE ELEMENTS RECEIVED BY THE LIST PASSED INTO INDIVIDUAL VARIABLES
   startTime , endTime ,opid, cgid
2. PARSE THE startTime,endTime INTO SIMPLEDATEFORMAT
3. CREATE PATTERN "Configured disks for group "+cgid+":" from LOGINSIGHT('LogInsight') where
   vdiag_product: 'hbr-secondary'
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
   TABLE
5. TRAVESE THE LOG FILE GET THE TIMESTAMP OF THE LOG LINE WHICH IS GREATER
   THAN THE startTime
       STORE IT IN A VARIABLE
       END TRAVESRSE
6. CREATE A CLOSURE
       6.1defretv = cgid +'.+'.toString()
       6.2 retv.getMetaClass().is_regex = true
       6.3 retv.getMetaClass().column = 'text'
       6.4RETURN retv
7. CREATE A VARIBLE cgidt2 STORE THE VALUE RETURED BY THE CLOSURE IN THE
   VARIABLE
8. CREATE PATTERN "ResumeRpoViolationReporting for group" from LOGINSIGHT('LogInsight')
   where vdiag_product: 'hbr-secondary'
9. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
   TABLE FILTER BY cgid
10. TRAVERSE THE TABLE FILE GET THE TIMESTAMP OF THE LOG LINE WHICH IS LESSER
    THAN THE endTime
11. CREATE PATTERN "[Originator@6876" from LOGINSIGHT('LogInsight') where vdiag_product: 'hbr-
    secondary',timestamp:[ge:greaterTime,le:lesserTime]
12. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
    TABLE
13. TRAVERSE THE LOG FILE
       13.1.   CREATE THREE VARIABLES time WHICH STORE THE TIMESTAMP OF THE
            LOGLINE ,ptime WHICH STORES THE BROKER TIMESTAMP AND msg WHICH STORE
            THE LOGLINE MESSAGE.
       END TRAVERSE
```

SECONDARY HCS

INPUT:REQUIRES THE ESX-SECONDARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIMESTAMP , ID AND MSG

```
    1. CREATE A CLOSURE
       1.1secondary_esx ->defretv = secondary_esx +'.+'.toString()
       1.2 retv.getMetaClass().is_regex = true
       1.3 retv.getMetaClass().column = 'text'
       1.4RETURN retv
  2. CREATE PATTERN "operationID" from 'hcs-secondary' filter by:[HMS_OPID(secondary_hcs)]
  3. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
  TABLE
  4. TRAVERSE THE LOG FILE
       4.1. CREATE A VARIABLEid AND STORE OPEERATION ID OF THE LOGLINE IN IT
       4.2 GET LOGLINE MESSAGE AND STORE IT IN A VARIABLE CALLED msg
       4.3 GET TIMESTAMP OF THE LOGLINE
```

```
        4.4 CREATE A REGULAR EXPRESSION WITH THE WORD ERROR AND SEARCH IT IN msg
        4.1 IF msg CONTAINS THE REGEX
                ADD ALERT
            ELSE
                DO NOTHING
            END IF
    5.END TRAVERSE
```

SECONDARY HMS-VSPHERE REPLICATION MANAGEMENT SERVER ON SITE 2

```
INPUT:REQUIRES THE HMS-SECONDARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIMESTAMP ,OPID  AND  MSG
    1. CREATE PATTERN '<<var1>> [<<event>>] (<<class_name>>) operationID=<<op_id>> |
       <<msg>>$' of secondary_hms from 'hms-secondary'
    2. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
       TABLE
    3. TRAVERSE THE LOG FILE
               STORE THE TIMESTAMP OF THE LOGLINE
```

PRIMARYHMS-VSPHERE REPLICATION MANAGEMENT SERVER ON SITE 1

```
INPUT:REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIMESTAMP ,OPID  AND  MSG
    1. CREATE PATTERN '[<<event>>] (<<class_name>>) operationID=<<op_id>> | <<msg>>$' of
       primary_hms from 'hms-primary'
    2. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
       TABLE
    3. TRAVERSE THE LOG FILE
                3.1. STORE THE TIMESTAMP OF THE LOGLINE
                3.2. CREATE A REGULAR EXPRESSION WITH THE WORD ERROR AND SEARCH IT
                     IN msg
                3.3.  IF msg CONTAINS THE REGEX
                        ADD ALERT
                        ELSE
                                DO NOTHING
                        END IF
       END TRAVERSE
```

SECONDARYVC-VCENTER SERVER ON SITE 2

```
INPUT:REQUIRES THE VC SECONDARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIMESTAMP ,OPID  AND  MSG
    1. CREATE A CLOSURE
        1.1  secondary_vc ->defstr = 'vcd-'+secondary_vc
        1.2  defretv = secondary_vc +'.+'.toString()
        1.3. retv.getMetaClass().is_regex = true
        1.4. retv.getMetaClass().column = 'text'
        1.5. RETURN retv
    2.  CREATE A VARIBLE test2 STORE THE VALUE RETURED BY THE CLOSURE IN THE
        VARIABLE
    3.  CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
    4.  SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
        TABLE
    5.  TRAVERSE THE LOG FILE
            5.1. STORE THE TIMESTAMP OF THE LOGLINE
            5.2. CREATE A REGULAR EXPRESSION WITH THE WORD ERROR AND SEARCH IT IN
                 msg
            5.3. IF msg CONTAINS THE REGEX
                        ADD ALERT
                        ELSE
                                DO NOTHING
                        END IF
        END TRAVERSE
```

SECONDARY VCD -VCLOUD DIRECTOR ON SITE 2

```
INPUT:REQUIRES THE VC SECONDARY LOGFILE
OUTPUT: A TABLE IS CREATED WITH THREE COLUMNS TIMESTAMP ,OPID  AND  MSG
    1.  CREATE A CLOSURE
          secondary_vcd ->defretv = secondary_vcd +'.+'.toString()
          1.3. retv.getMetaClass().is_regex = true
          1.4. retv.getMetaClass().column = 'text'
          1.5. RETURN retv
    2.  CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
        VARIABLE
    3.  CREATE PATTERN "requestId" from 'vcd-secondary' filter by:[HMS_OPID(secondary_vcd)]
    4.  SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
        TABLE
    5.  TRAVERSE THE LOG FILE
            5.1. STORE THE TIMESTAMP OF THE LOGLINE
            5.2. CREATE A REGULAR EXPRESSION WITH THE WORD ERROR AND SEARCH IT IN
                 msg
            5.3. IF msg CONTAINS THE REGEX
                        ADD ALERT
                        ELSE
                                DO NOTHING
                        END IF
        END TRAVERSE
```

TRACING THE PLANNED FAILOVER OPERATION

```
INPUT: REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND PLANNED FAILOVER
1.CREATE A CLOSURE
```

```
        1.1. opid ->defretv = opid +'.+'.toString()
        1.2. retv.getMetaClass().is_regex = true
        1.3. retv.getMetaClass().column = 'text'
        1.4. RETURN retv
2. CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
VARIABLE
3. CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
5. TRAVERSE THE LOG FILE
        5.1. CREATE VARIABLES
                        cgid WHICH STORES LOGLINE CGID
                        opidWHICH STORES LOGLINE OPID
        5.2. CREATE PATTERN 'ReplicationConfig file'
        5.3. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES
            IN A TABLE
        5.4. TRAVERSE THE LOG FILE
                        5.4.1.  EXTRACT THE VM NAME AND STORE IT IN A VARIABLE vmname
                        5.4.2.  CREATE A LIST
                        5.4.3.  IF THE LIST CONTAINS THE VM
                                        THEN DO NOTHING
                                ELSE
                                        ADD VM IN THE LIST
                                ENDIF
        END TRAVERSE
        5.5. EXTRACT THE VM NAME IN ORDER FROM THE LIST
END TRAVERSE
```

TRACING THE TEST FAILOVER OPERATION

```
INPUT: REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND TEST FAILOVER
1.CREATE A CLOSURE
        1.1. opid ->defretv = opid +'.+'.toString()
        1.2. retv.getMetaClass().is_regex = true
        1.3. retv.getMetaClass().column = 'text'
        1.4. RETURN retv
2. CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
VARIABLE
3. CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
5. TRAVERSE THE LOG FILE
        5.1. CREATE VARIABLES
                        cgid WHICH STORES LOGLINE CGID
                        opidWHICH STORES LOGLINE OPID
        5.3. CREATE PATTERN 'ReplicationConfig file'
        5.6. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES
            IN A TABLE
        5.7. TRAVERSE THE LOG FILE
```

```
                    5.7.1.  EXTRACT THE VM NAME AND STORE IT IN A VARIABLE vmname
                    5.7.2.  CREATE A LIST
                    5.7.3.  IF THE LIST CONTAINS THE VM
                                THEN DO NOTHING
                            ELSE
                                ADD VM IN THE LIST
                            ENDIF
    END TRAVERSE
    5.8. EXTRACT THE VM NAME IN ORDER FROM THE LIST
END TRAVERSE
```

TRACING THE REVERSE REPLICATION OPERATION

```
INPUT: REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND PLANNED FAILOVER
1.CREATE A CLOSURE
    1.1. opid ->defretv = opid +'.+'.toString()
    1.2. retv.getMetaClass().is_regex = true
    1.3. retv.getMetaClass().column = 'text'
    1.4. RETURN retv
2. CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
VARIABLE
3. CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
5. TRAVERSE THE LOG FILE
    5.1. CREATE VARIABLES
                    cgid WHICH STORES LOGLINE CGID
                    opidWHICH STORES LOGLINE OPID
    5.4. CREATE PATTERN 'ReplicationConfig file'
    5.9. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES
         IN A TABLE
    5.10.   TRAVERSE THE LOG FILE
                    5.10.1. EXTRACT THE VM NAME AND STORE IT IN A VARIABLE vmname
                    5.10.2. CREATE A LIST
                    5.10.3. IF THE LIST CONTAINS THE VM
                                THEN DO NOTHING
                            ELSE
                                ADD VM IN THE LIST
                            ENDIF
    END TRAVERSE
    5.11.   EXTRACT THE VM NAME IN ORDER FROM THE LIST
END TRAVERSE
```

TRACING THE TEST FAILOVER OPERATION

```
INPUT: REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND TEST CLEANUP
1.CREATE A CLOSURE
    1.1. opid ->defretv = opid +'.+'.toString()
```

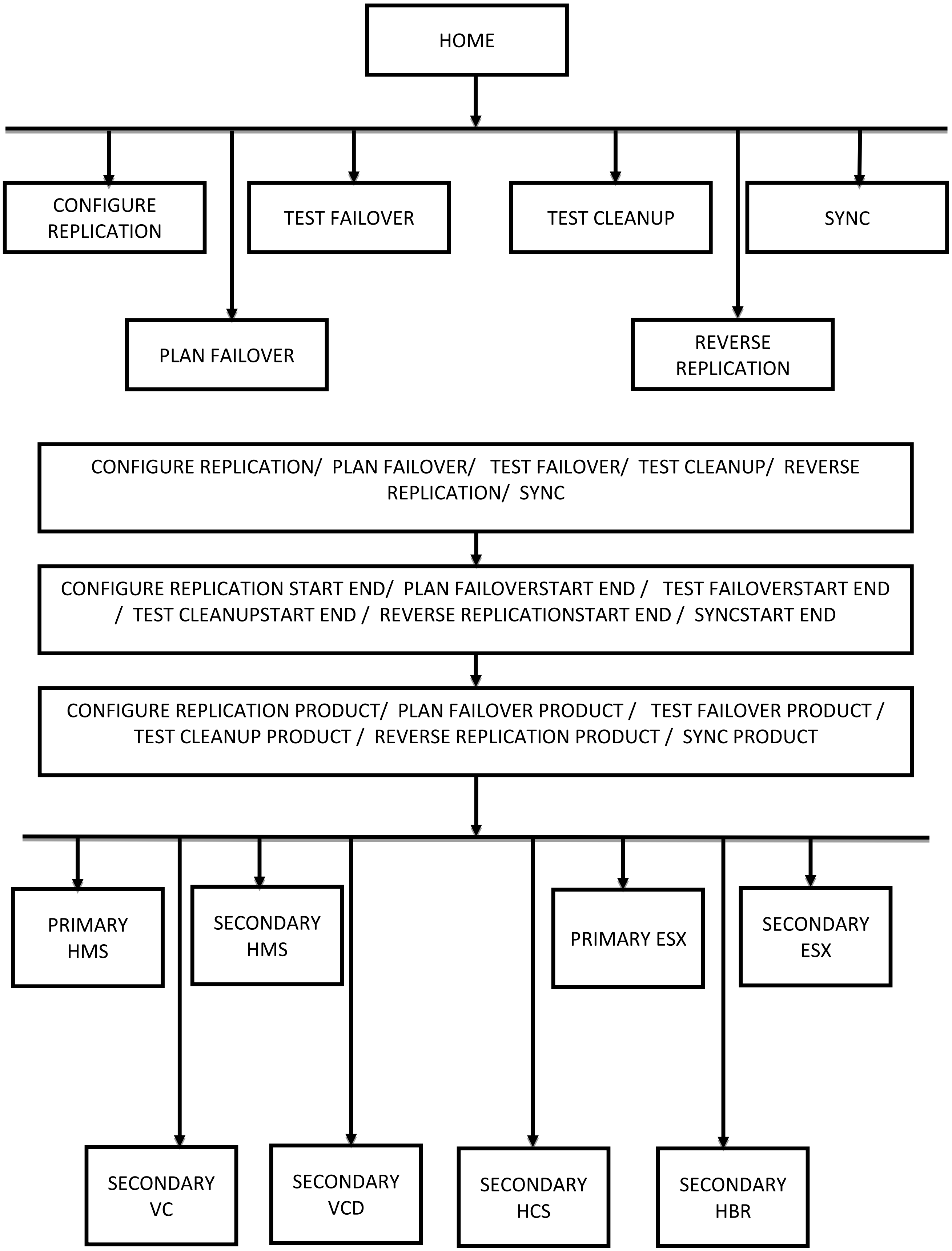
```
        1.2. retv.getMetaClass().is_regex = true
        1.3. retv.getMetaClass().column = 'text'
        1.4. RETURN retv
2. CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
VARIABLE
3. CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
5. TRAVERSE THE LOG FILE
        5.1. CREATE VARIABLES
                        cgid WHICH STORES LOGLINE CGID
                        opidWHICH STORES LOGLINE OPID
        5.5. CREATE PATTERN 'ReplicationConfig file'
        5.12.   SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING
            LOGLINES IN A TABLE
        5.13.   TRAVERSE THE LOG FILE
                        5.13.1.  EXTRACT THE VM NAME AND STORE IT IN A VARIABLE vmname
                        5.13.2.  CREATE A LIST
                        5.13.3.  IF THE LIST CONTAINS THE VM
                                         THEN DO NOTHING
                                 ELSE
                                         ADD VM IN THE LIST
                                 ENDIF
        END TRAVERSE
        5.14.   EXTRACT THE VM NAME IN ORDER FROM THE LIST
END TRAVERSE
```

TRACING THE SYNC OPERATION

```
INPUT: REQUIRES THE HMS-PRIMARY LOGFILE
OUTPUT:A TABLE IS CREATED WITH TWOCOLUMN VM AND SYNC
1.CREATE A CLOSURE
        1.1. opid ->defretv = opid +'.+'.toString()
        1.2. retv.getMetaClass().is_regex = true
        1.3. retv.getMetaClass().column = 'text'
        1.4. RETURN retv
2. CREATE A VARIBLE HMS_OPID STORE THE VALUE RETURED BY THE CLOSURE IN THE
VARIABLE
3. CREATE PATTERN "opID" from 'vc-secondary' filter by:[VC_OPID(secondary_vc)]
4. SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING LOGLINES IN A
TABLE
5. TRAVERSE THE LOG FILE
        5.1. CREATE VARIABLES
                        cgid WHICH STORES LOGLINE CGID
                        opidWHICH STORES LOGLINE OPID
        5.6. CREATE PATTERN 'ReplicationConfig file'
        5.15.   SEARCH THE DATABASE WITH THE PATTERN AND STORE MATCHING
            LOGLINES IN A TABLE
        5.16.   TRAVERSE THE LOG FILE
                        5.16.1.  EXTRACT THE VM NAME AND STORE IT IN A VARIABLE vmname
```

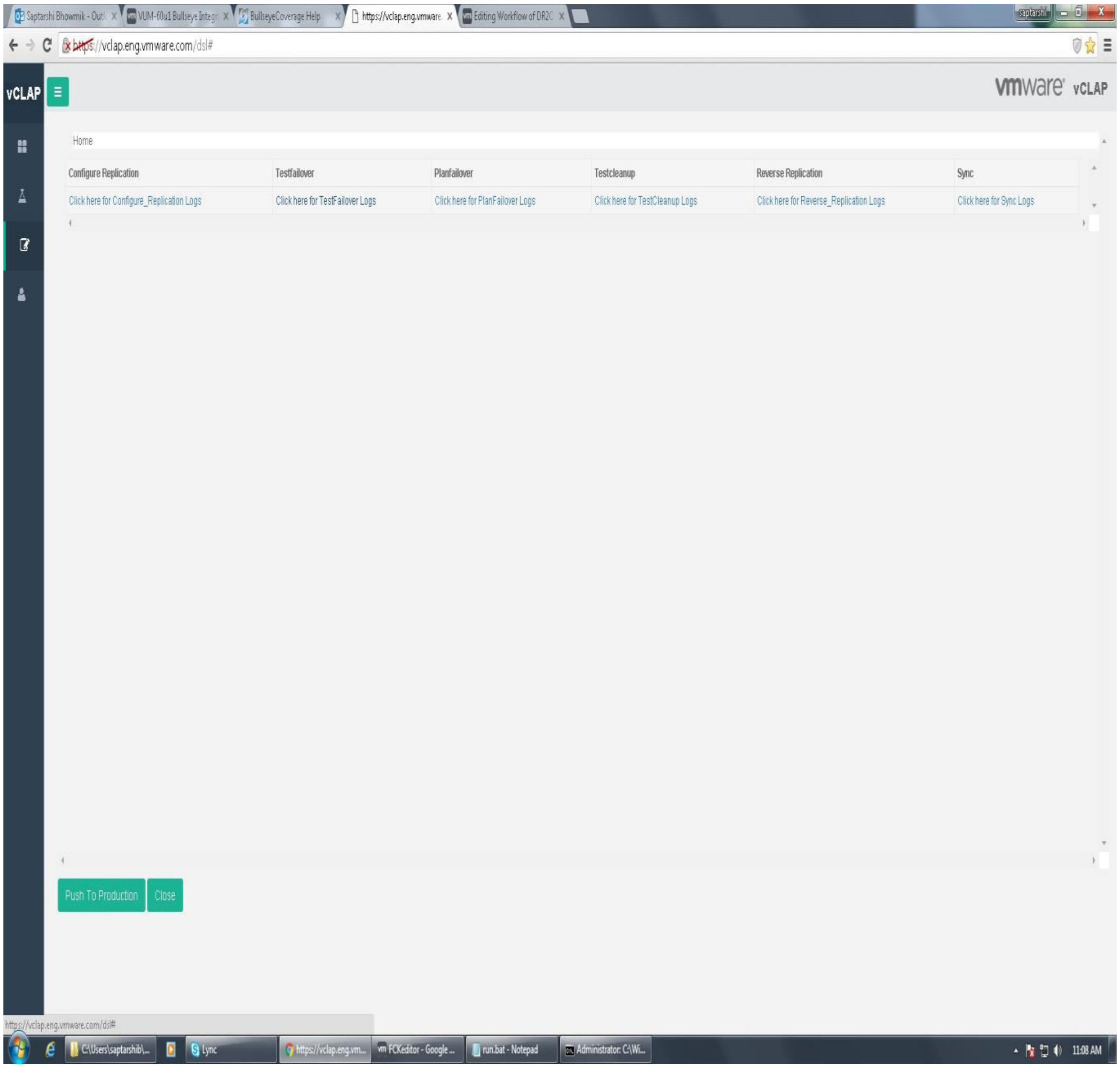
```
                    5.16.2.  CREATE A LIST
                    5.16.3.  IF THE LIST CONTAINS THE VM
                                 THEN DO NOTHING
                             ELSE
                                 ADD VM IN THE LIST
                             ENDIF
    END TRAVERSE
    5.17.    EXTRACT THE VM NAME IN ORDER FROM THE LIST
END TRAVERSE
```

BLOCK DIAGRAM


HOME
CONFIGURE REPLICATION
TEST FAILOVER
TEST CLEANUP
SYNC
PLAN FAILOVER
REVERSE REPLICATION
CONFIGURE REPLICATION/ PLAN FAILOVER/ TEST FAILOVER/ TEST CLEANUP/ REVERSE REPLICATION/ SYNC
CONFIGURE REPLICATION START END/ PLAN FAILOVERSTART END / TEST FAILOVERSTART END / TEST CLEANUPSTART END / REVERSE REPLICATIONSTART END / SYNCSTART END
CONFIGURE REPLICATION PRODUCT/ PLAN FAILOVER PRODUCT / TEST FAILOVER PRODUCT / TEST CLEANUP PRODUCT / REVERSE REPLICATION PRODUCT / SYNC PRODUCT
PRIMARY HMS
SECONDARY HMS
PRIMARY ESX
SECONDARY ESX
SECONDARY VC
SECONDARY VCD
SECONDARY HCS
SECONDARY HBR

5.3 RESULTS

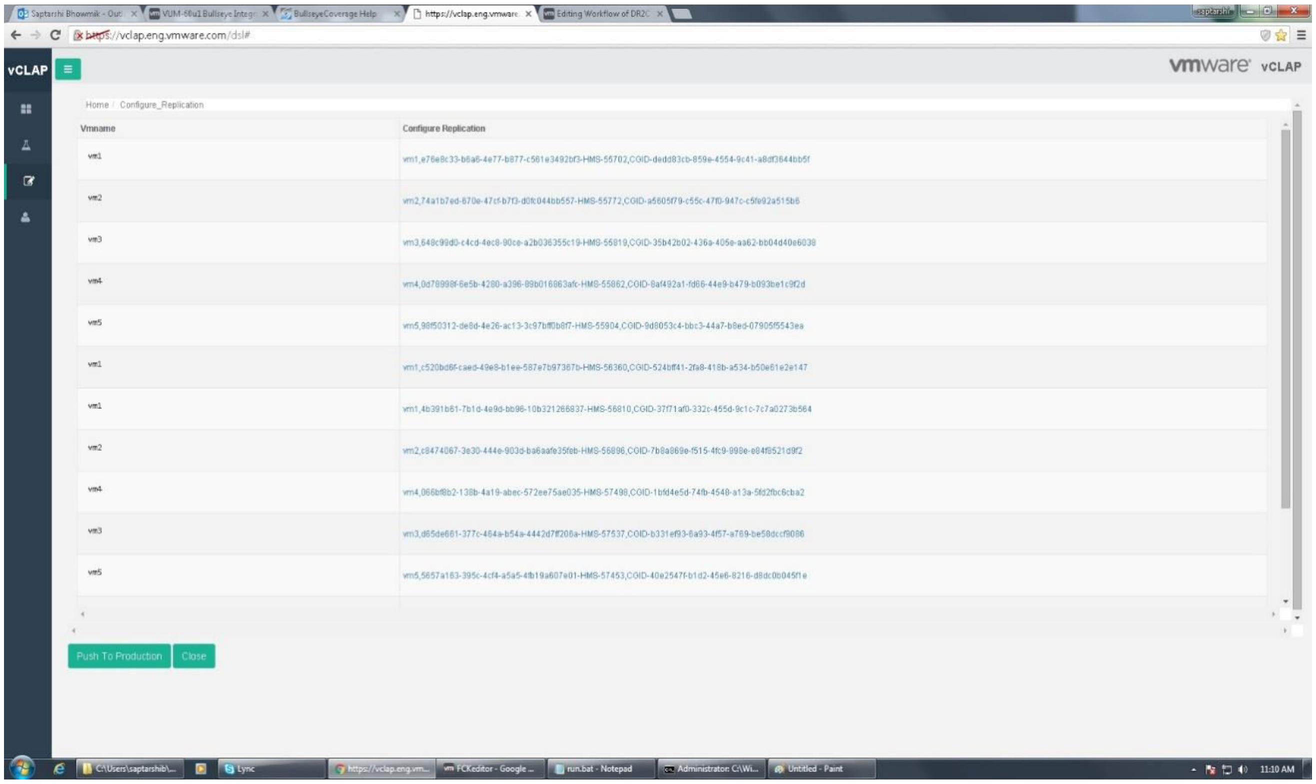
5.3.1 CONFIGURE REPLICATION OPERATION



Configure Replication	Testfailover	Planfailover	Testcleanup	Reverse Replication	Sync
Click here for Configure_Replication Logs	Click here for TestFailover Logs	Click here for PlanFailover Logs	Click here for TestCleanup Logs	Click here for Reverse_Replication Logs	Click here for Sync Logs

Figure 15 The home page from where a particular operation can be traced

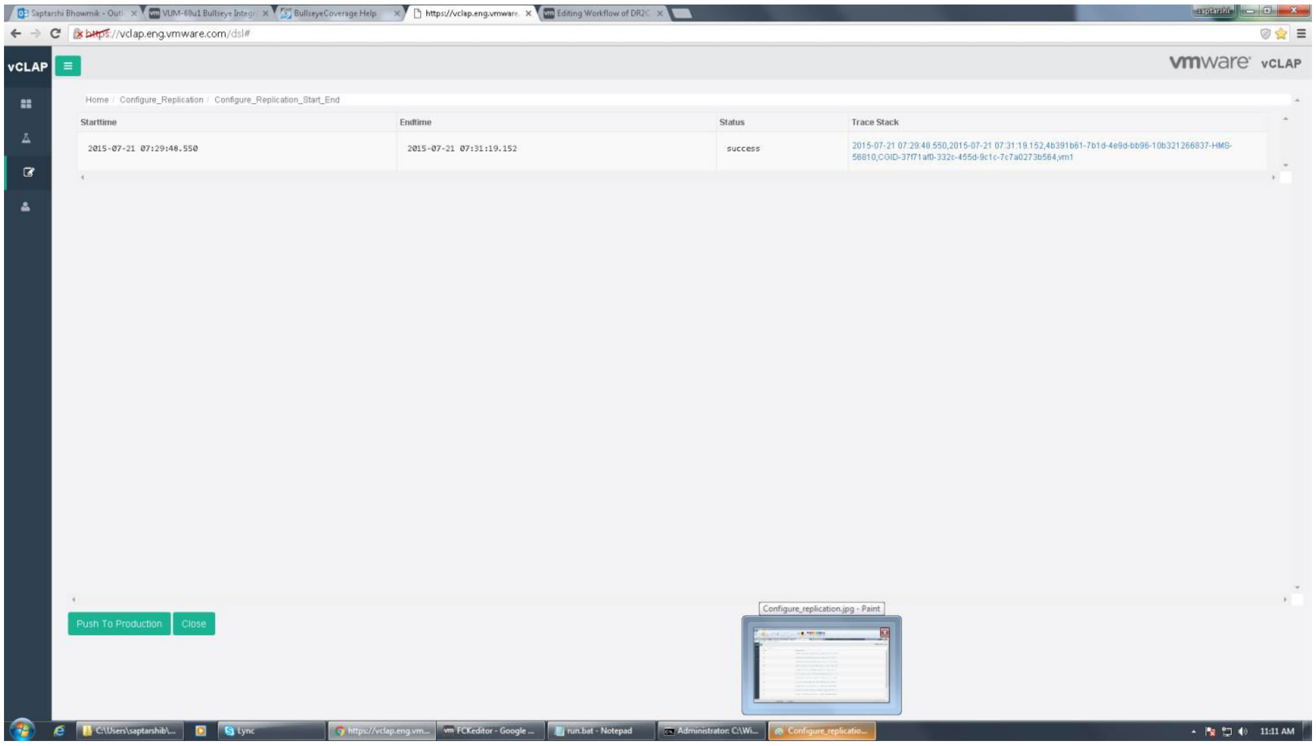
Since the tool is developed as a web application, the results produced by it will be displayed as a web page. This is the very first page which is shown when the tool is given run. It has listed six operations namely Configure Replication, Planned Failover, Test Failover, Test Cleanup, Reverse Replication and Sync. On clicking the above function, tool starts tracing the operation in the database which contains the log files.



Home / Configure_Replication

Vmname	Configure Replication
vm1	vm1,e76e8c33-b6a6-4e77-b877-c561e3492bf3-HMS-55702,CGID-dedd83cb-859e-4554-9c41-a8df3644bb5f
vm2	vm2,74a1b7ed-670e-47cf-b7f3-d0fc044bb557-HMS-55772,CGID-a5605f79-c55c-47f0-947c-c5fe92a515b6
vm3	vm3,648c99d0-c4cd-4ec8-90ce-a2b036355c19-HMS-55819,CGID-35b42b02-436a-405e-aa62-bb04d40e6038
vm4	vm4,0d78998f-6e5b-4280-a396-89b016863afc-HMS-55862,CGID-8af492a1-fd66-44e9-b479-b093be1c9f2d
vm5	vm5,98f50312-de8d-4e26-ac13-3c97bff0b8f7-HMS-55904,CGID-9d8053c4-bbc3-44a7-b8ed-07905f5543ea
vm1	vm1,c520bd6f-caed-49e8-b1ee-587e7b97367b-HMS-56360,CGID-524bff41-2fa8-418b-a534-b50e61e2e147
vm1	vm1,4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810,CGID-37f71af0-332c-455d-9c1c-7c7a0273b564
vm2	vm2,c8474067-3e30-444e-903d-ba6aafe35feb-HMS-56896,CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2
vm4	vm4,066bf8b2-138b-4a19-abec-572ee75ae035-HMS-57498,CGID-1bfd4e5d-74fb-4548-a13a-5fd2fbc6cba2
vm3	vm3,d65de661-377c-464a-b54a-4442d7ff206a-HMS-57537,CGID-b331ef93-6a93-4f57-a769-be58dccf9086
vm5	vm5,5657a163-395c-4cf4-a5a5-4fb19a607e01-HMS-57453,CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e

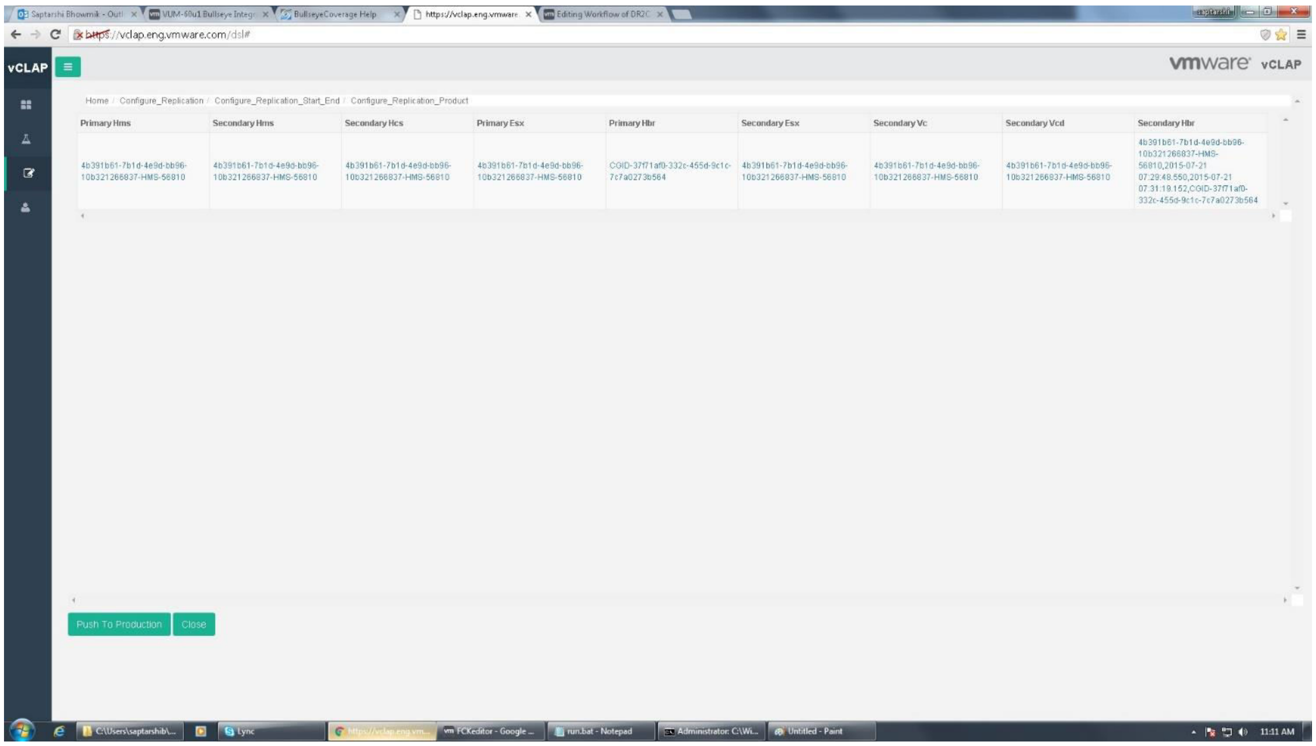
Figure 16 The Configure Replication page showing the names of the vm that performed that operation



Home / Configure_Replication / Configure_Replication_Start_End

Starttime	Endtime	Status	Trace Stack
2015-07-21 07:29:48.550	2015-07-21 07:31:19.152	success	2015-07-21 07:29:48.550,2015-07-21 07:31:19.152,4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810,CGID-37f71af0-332c-455d-9c1c-7c7a0273b564,vm1

Figure 17 Showing the start time and end time of Configure Replication operation in a particular vm



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product

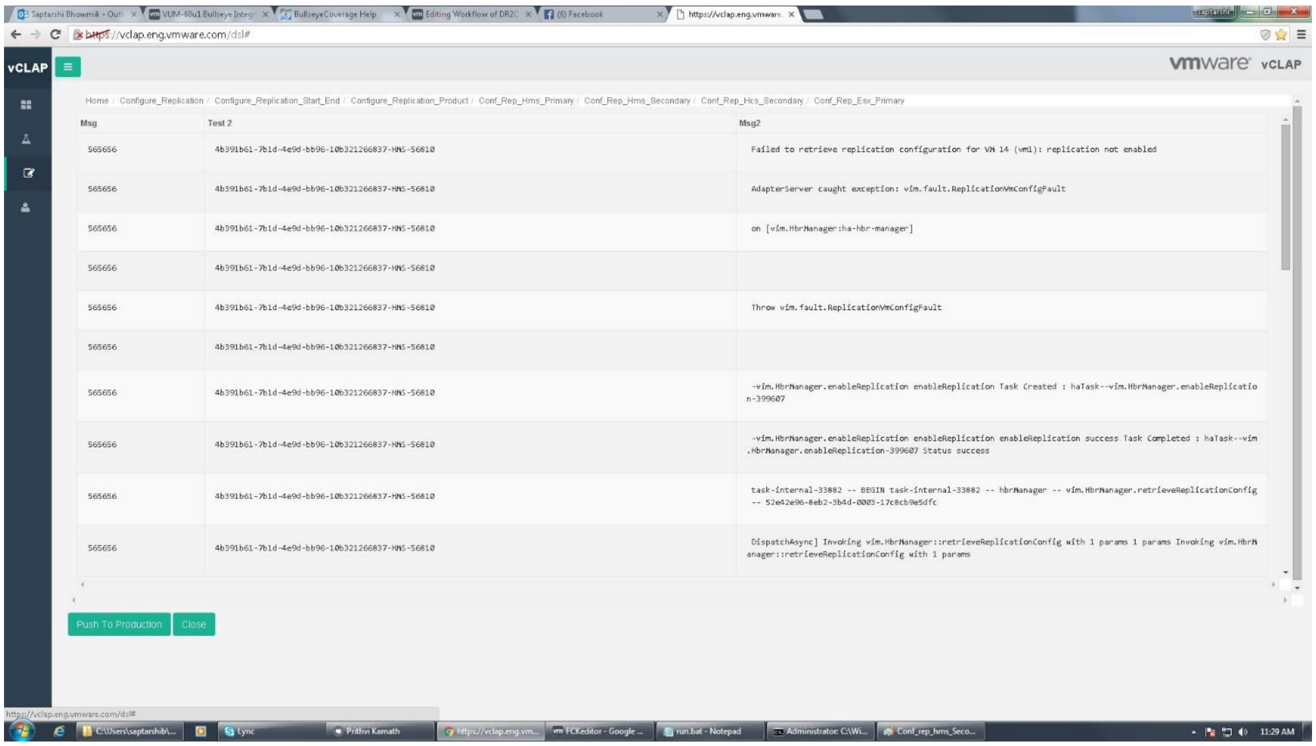
Primary Hms	Secondary Hms	Secondary Hcs	Primary Esx	Primary Hbr	Secondary Esx	Secondary Vc	Secondary Vcd	Secondary Hbr
4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	CGID-37f71af0-332c-455d-9c1c-7c7a0273b564	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810,2015-07-21 07:29:48.550,2015-07-21 07:31:19.152,CGID-37f71af0-332c-455d-9c1c-7c7a0273b564

Figure 18 The products that were involved in Configure Replication operation

In, Figure 16, The is the vm names along with the information that are unique to that vm, this information are a combination of operation ID, start time of that operation for that vm, end time of the operation for that vm etc. These information will help in further analysis of the logs in the database and extracting information specific to the vm.

On clicking a particular vm in the above web page what we get is another web page with the start time end time, status and some information which are printed in the column of trace stack. Figure 17 shows how it looks like.

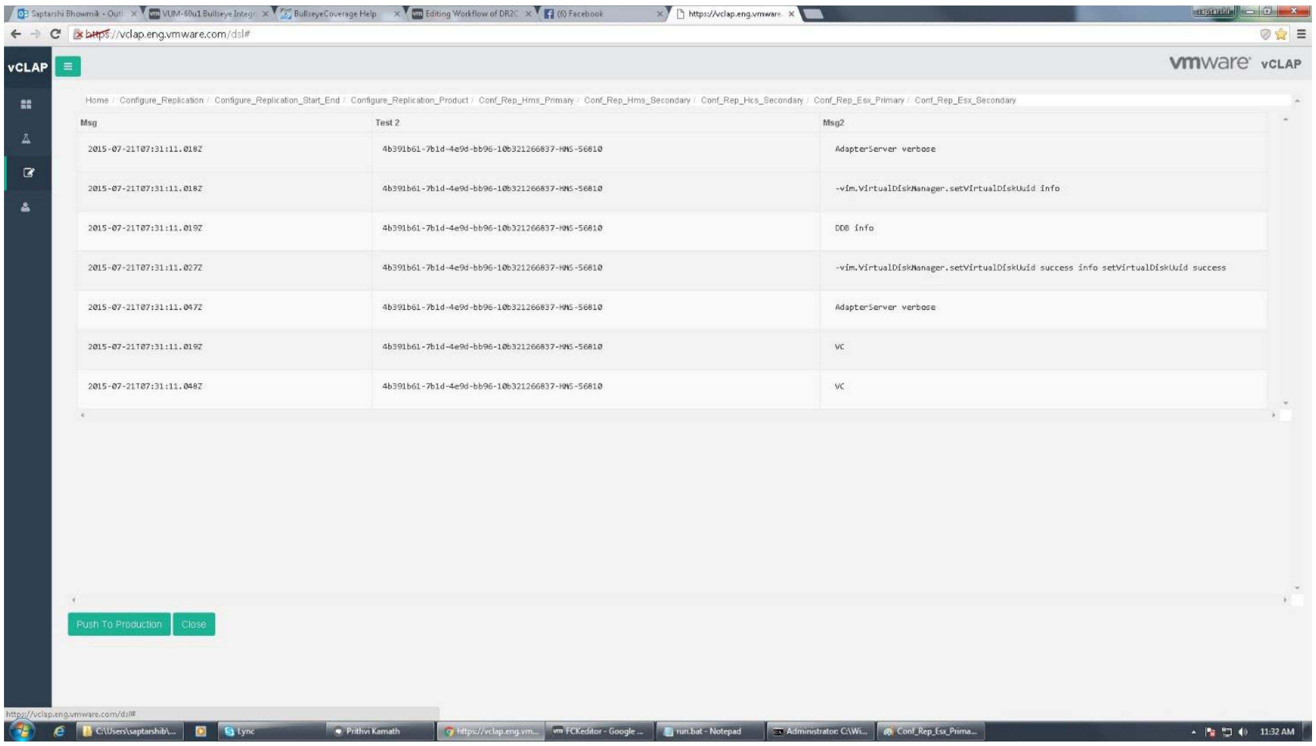
From there, clicking the trace stack column leads to another web page, which has the list of products, that are involved in the process of disaster recovery. Figure 18 clearly shows the result. These products on the web page, when clicked shows their corresponding log analysis result



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary / Conf_Rep_Hcs_Secondary / Conf_Rep_Esx_Primary

Msg	Test 2	Msg2
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Failed to retrieve replication configuration for VM 14 (vm1): replication not enabled
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	AdapterServer caught exception: vim.fault.ReplicationVmConfigFault
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	on [vim.HbrManager:ha-hbr-manager]
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Throw vim.fault.ReplicationVmConfigFault
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-vim.HbrManager.enableReplication enableReplication Task Created : haTask--vim.HbrManager.enableReplication-399607
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-vim.HbrManager.enableReplication enableReplication enableReplication success Task Completed : haTask--vim.HbrManager.enableReplication-399607 Status success
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	task-internal-33882 -- BEGIN task-internal-33882 -- hbrManager -- vim.HbrManager.retrieveReplicationConfig -- 52e42e96-8eb2-3b4d-0003-17c8cb9e5dfc
565656	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	DispatchAsync] Invoking vim.HbrManager::retrieveReplicationConfig with 1 params 1 params Invoking vim.HbrManager::retrieveReplicationConfig with 1 params

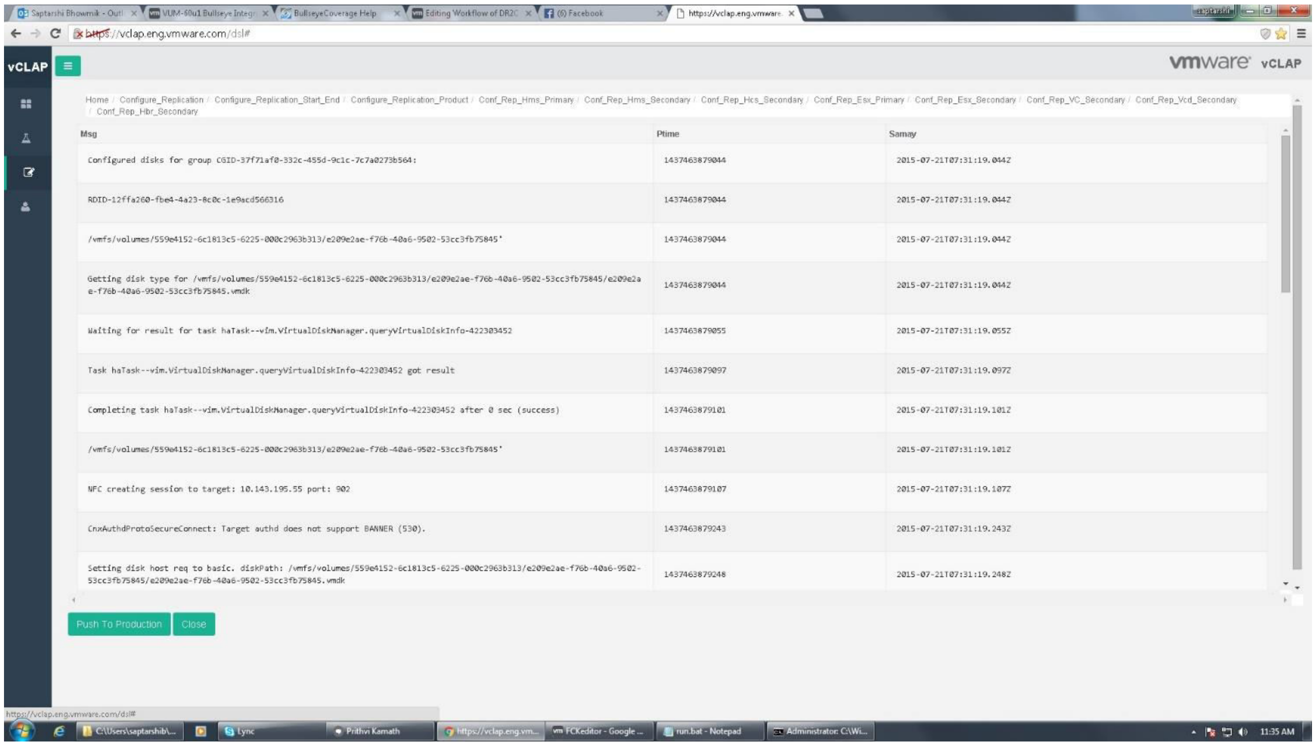
Figure 19 The output produced by the tool while tracing Configure Replication in ESXi Primary



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary / Conf_Rep_Hcs_Secondary / Conf_Rep_Esx_Primary / Conf_Rep_Esx_Secondary

Msg	Test 2	Msg2
2015-07-21T07:31:11.018Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	AdapterServer verbose
2015-07-21T07:31:11.018Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-vim.VirtualDiskManager.setVirtualDiskUuid info
2015-07-21T07:31:11.019Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	DDB info
2015-07-21T07:31:11.027Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-vim.VirtualDiskManager.setVirtualDiskUuid success info setVirtualDiskUuid success
2015-07-21T07:31:11.047Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	AdapterServer verbose
2015-07-21T07:31:11.019Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	VC
2015-07-21T07:31:11.048Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	VC

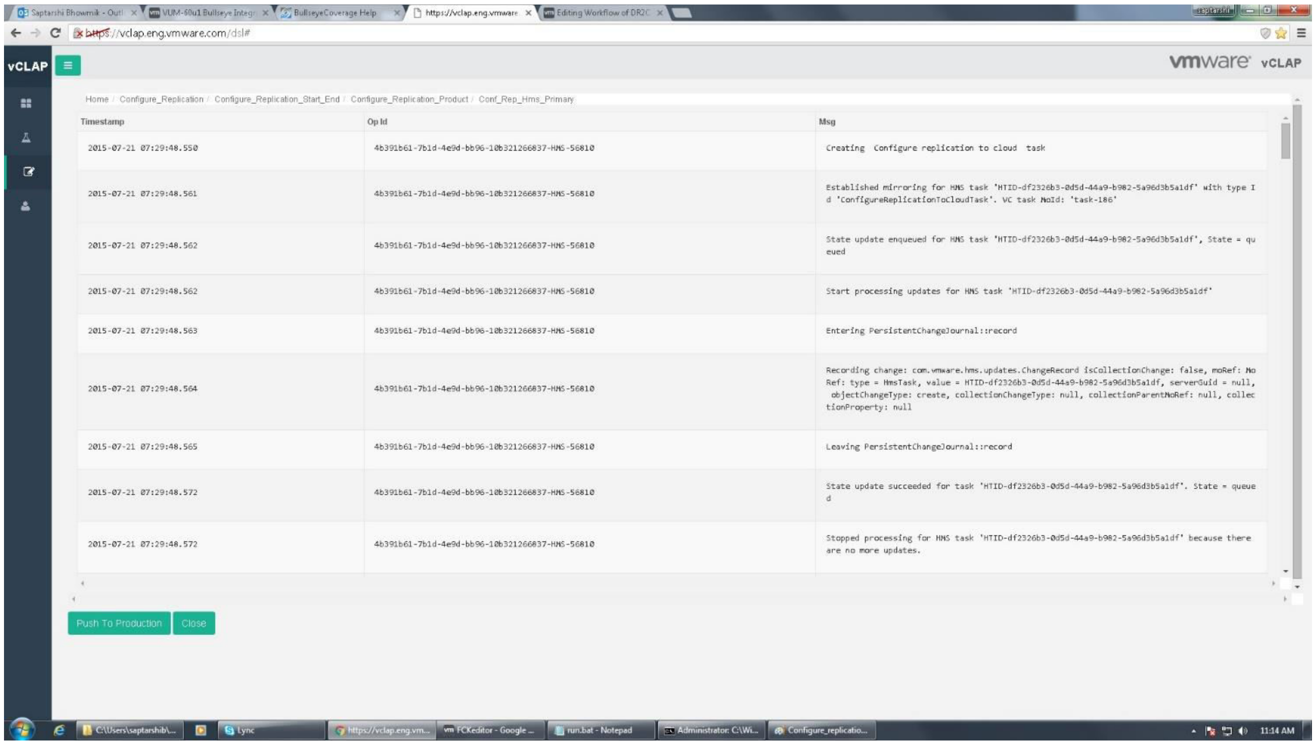
Figure 20 The output produced by the tool while tracing Configure Replication in ESXi Secondary



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary / Conf_Rep_Hcs_Secondary / Conf_Rep_Esx_Primary / Conf_Rep_Esx_Secondary / Conf_Rep_VC_Secondary / Conf_Rep_Vcd_Secondary / Conf_Rep_Hbr_Secondary

Msg	Ptime	Samay
Configured disks for group CGID-37f71af0-332c-455d-9c1c-7c7a0273b564:	1437463879044	2015-07-21T07:31:19.044Z
RDID-12ffa260-fbe4-4a23-8c0c-1e9acd566316	1437463879044	2015-07-21T07:31:19.044Z
/vmfs/volumes/559e4152-6c1813c5-6225-000c2963b313/e209e2ae-f76b-40a6-9502-53cc3fb75845'	1437463879044	2015-07-21T07:31:19.044Z
Getting disk type for /vmfs/volumes/559e4152-6c1813c5-6225-000c2963b313/e209e2ae-f76b-40a6-9502-53cc3fb75845/e209e2ae-f76b-40a6-9502-53cc3fb75845.vmdk	1437463879044	2015-07-21T07:31:19.044Z
Waiting for result for task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422303452	1437463879055	2015-07-21T07:31:19.055Z
Task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422303452 got result	1437463879097	2015-07-21T07:31:19.097Z
Completing task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422303452 after 0 sec (success)	1437463879101	2015-07-21T07:31:19.101Z
/vmfs/volumes/559e4152-6c1813c5-6225-000c2963b313/e209e2ae-f76b-40a6-9502-53cc3fb75845'	1437463879101	2015-07-21T07:31:19.101Z
NFC creating session to target: 10.143.195.55 port: 902	1437463879107	2015-07-21T07:31:19.107Z
CnxAuthdProtoSecureConnect: Target authd does not support BANNER (530).	1437463879243	2015-07-21T07:31:19.243Z
Setting disk host req to basic. diskPath: /vmfs/volumes/559e4152-6c1813c5-6225-000c2963b313/e209e2ae-f76b-40a6-9502-53cc3fb75845/e209e2ae-f76b-40a6-9502-53cc3fb75845.vmdk	1437463879248	2015-07-21T07:31:19.248Z

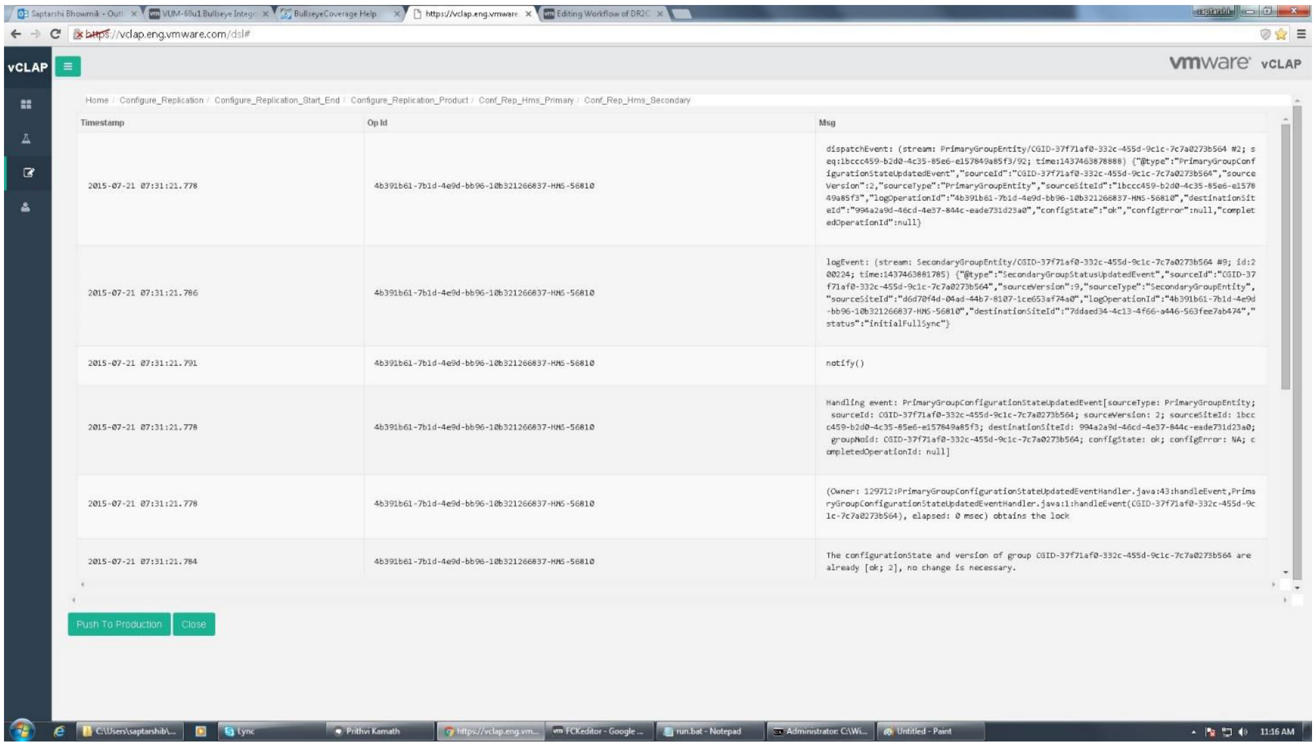
Figure 21 The output produced by the tool while tracing Configure Replication in HBR Secondary



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary

Timestamp	Op Id	Msg
2015-07-21 07:29:48.550	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Creating Configure replication to cloud task
2015-07-21 07:29:48.561	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Established mirroring for HMS task 'HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df' with type Id 'ConfigureReplicationToCloudTask'. VC task MoId: 'task-186'
2015-07-21 07:29:48.562	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	State update enqueued for HMS task 'HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df', State = queued
2015-07-21 07:29:48.562	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Start processing updates for HMS task 'HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df'
2015-07-21 07:29:48.563	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Entering PersistentChangeJournal::record
2015-07-21 07:29:48.564	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df, serverGuid = null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 07:29:48.565	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Leaving PersistentChangeJournal::record
2015-07-21 07:29:48.572	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	State update succeeded for task 'HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df'. State = queued
2015-07-21 07:29:48.572	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Stopped processing for HMS task 'HTID-df2326b3-0d5d-44a9-b982-5a96d3b5a1df' because there are no more updates.

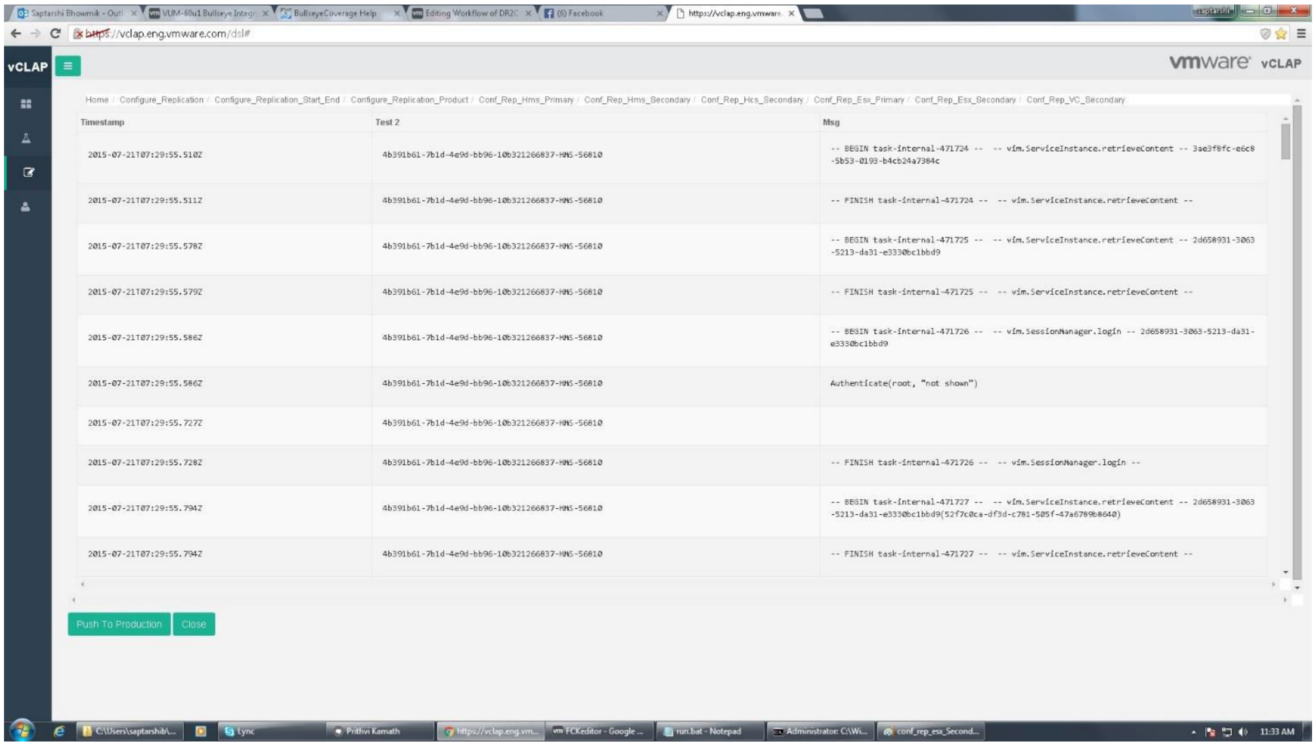
Figure 22 The output produced by the tool while tracing Configure Replication in HMS Primary


vCLAP
vmware vCLAP
https://vclap.eng.vmware.com/dsl#
Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary
Push To Production
Close

Timestamp	Op Id	Msg
2015-07-21 07:31:21.778	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	dispatchEvent: (stream: PrimaryGroupEntity/CGID-37f71af0-332c-455d-9c1c-7c7a0273b564 #2; seq:1bccc459-b2d0-4c35-85e6-e157849a85f3/92; time:1437463878888) {"@type":"PrimaryGroupConfigurationStateUpdatedEvent","sourceId":"CGID-37f71af0-332c-455d-9c1c-7c7a0273b564","sourceVersion":2,"sourceType":"PrimaryGroupEntity","sourceSiteId":"1bccc459-b2d0-4c35-85e6-e157849a85f3","logOperationId":"4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810","destinationSiteId":"994a2a9d-46cd-4e37-844c-eade731d23a0","configState":"ok","configError":null,"completedOperationId":null}
2015-07-21 07:31:21.786	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	logEvent: (stream: SecondaryGroupEntity/CGID-37f71af0-332c-455d-9c1c-7c7a0273b564 #9; id:200224; time:1437463881785) {"@type":"SecondaryGroupStatusUpdatedEvent","sourceId":"CGID-37f71af0-332c-455d-9c1c-7c7a0273b564","sourceVersion":9,"sourceType":"SecondaryGroupEntity","sourceSiteId":"d6d70f4d-04ad-44b7-8107-1ce653af74a0","logOperationId":"4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810","destinationSiteId":"7ddaed34-4c13-4f66-a446-563fee7ab474","status":"initialFullSync"}
2015-07-21 07:31:21.791	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	notify()
2015-07-21 07:31:21.778	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Handling event: PrimaryGroupConfigurationStateUpdatedEvent[sourceType: PrimaryGroupEntity; sourceId: CGID-37f71af0-332c-455d-9c1c-7c7a0273b564; sourceVersion: 2; sourceSiteId: 1bccc459-b2d0-4c35-85e6-e157849a85f3; destinationSiteId: 994a2a9d-46cd-4e37-844c-eade731d23a0; groupMoid: CGID-37f71af0-332c-455d-9c1c-7c7a0273b564; configState: ok; configError: NA; completedOperationId: null]
2015-07-21 07:31:21.778	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	(Owner: 129712:PrimaryGroupConfigurationStateUpdatedEventHandler.java:43:handleEvent,PrimaryGroupConfigurationStateUpdatedEventHandler.java:1:handleEvent(CGID-37f71af0-332c-455d-9c1c-7c7a0273b564), elapsed: 0 msec) obtains the lock
2015-07-21 07:31:21.784	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	The configurationState and version of group CGID-37f71af0-332c-455d-9c1c-7c7a0273b564 are already [ok; 2], no change is necessary.

Figure 23 The output produced by the tool while tracing Configure Replication in HMS Secondary

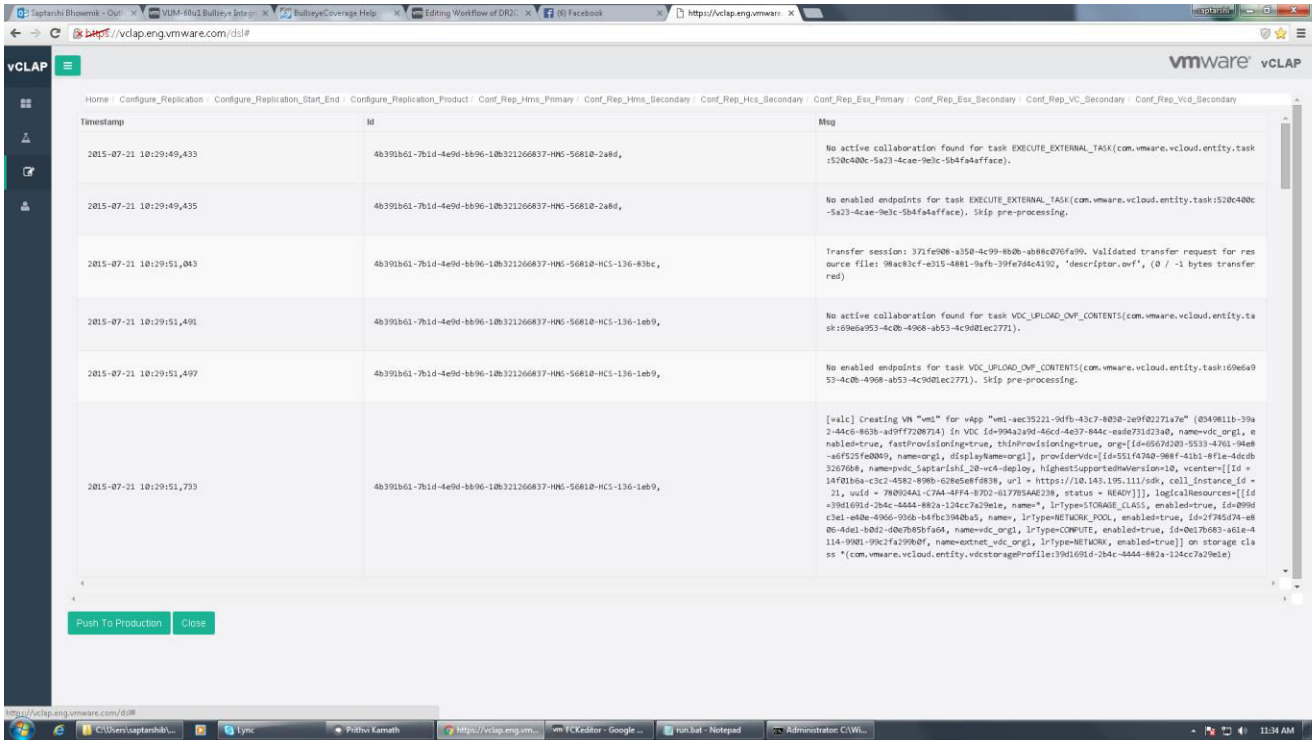
In, Figure 19 the result shows what record the ESXi Primary, which is the hypervisor on the first site or he protected site leaves when Configure Replication operation is done. The tool displays the operation ID in one column and the messages that are recorded in the log files. Figure 20 similarly shows the same thing but here the product is ESXi Secondary that is the hypervisor on the recovery site. Even it also shows for Configure Replication operation. Figure 21 the result shows what record the HBR or better to say vSphere Replication Server on the Recovery site leaves when Configure Replication operation is done. It has three columns and it also shows time, operation ID and log message. The next two images that is, Figure 22 and Figure 23 shows the record the HMS, which is the vSphere Replication Management Server on the protected site and the recovery site leaves when Configure Replication operation is done. Figure 24 shows the record left by vCenter on the recovery site and Figure 25 shows the log records of Configure Replication on vCloud Direcctor.



Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary / Conf_Rep_Hcs_Secondary / Conf_Rep_Esx_Primary / Conf_Rep_Esx_Secondary / Conf_Rep_VC_Secondary

Timestamp	Test 2	Msg
2015-07-21T07:29:55.510Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- BEGIN task-internal-471724 -- -- vim.ServiceInstance.retrieveContent -- 3ae3f8fc-e6c8-5b53-0193-b4cb24a7384c
2015-07-21T07:29:55.511Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- FINISH task-internal-471724 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T07:29:55.578Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- BEGIN task-internal-471725 -- -- vim.ServiceInstance.retrieveContent -- 2d658931-3063-5213-da31-e3330bc1bbd9
2015-07-21T07:29:55.579Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- FINISH task-internal-471725 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T07:29:55.586Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- BEGIN task-internal-471726 -- -- vim.SessionManager.login -- 2d658931-3063-5213-da31-e3330bc1bbd9
2015-07-21T07:29:55.586Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	Authenticate(root, "not shown")
2015-07-21T07:29:55.727Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	
2015-07-21T07:29:55.728Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- FINISH task-internal-471726 -- -- vim.SessionManager.login --
2015-07-21T07:29:55.794Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- BEGIN task-internal-471727 -- -- vim.ServiceInstance.retrieveContent -- 2d658931-3063-5213-da31-e3330bc1bbd9(52f7c0ca-df3d-c781-505f-47a6789b8640)
2015-07-21T07:29:55.794Z	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810	-- FINISH task-internal-471727 -- -- vim.ServiceInstance.retrieveContent --

Figure 24 The output produced by the tool while tracing Configure Replication in VC Secondary

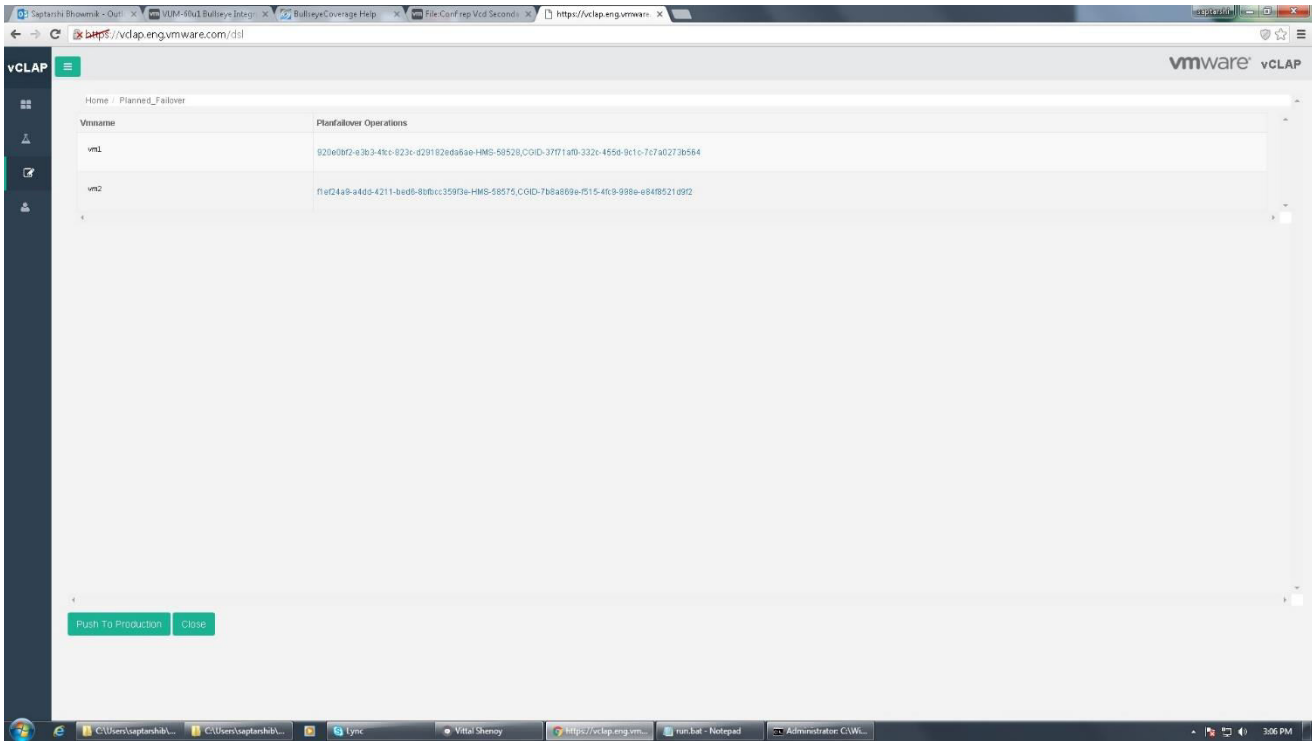


Home / Configure_Replication / Configure_Replication_Start_End / Configure_Replication_Product / Conf_Rep_Hms_Primary / Conf_Rep_Hms_Secondary / Conf_Rep_Hcs_Secondary / Conf_Rep_Esx_Primary / Conf_Rep_Esx_Secondary / Conf_Rep_VC_Secondary / Conf_Rep_Vcd_Secondary

Timestamp	Id	Msg
2015-07-21 10:29:49,433	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-2a8d,	No active collaboration found for task EXECUTE_EXTERNAL_TASK(com.vmware.vcloud.entity.task:520c400c-5a23-4cae-9e3c-5b4fa4afface).
2015-07-21 10:29:49,435	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-2a8d,	No enabled endpoints for task EXECUTE_EXTERNAL_TASK(com.vmware.vcloud.entity.task:520c400c-5a23-4cae-9e3c-5b4fa4afface). Skip pre-processing.
2015-07-21 10:29:51,043	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-HCS-136-83bc,	Transfer session: 371fe908-a350-4c99-8b0b-ab88c076fa99. Validated transfer request for resource file: 98ac83cf-e315-4881-9afb-39fe7d4c4192, 'descriptor.ovf', (0 / -1 bytes transferred)
2015-07-21 10:29:51,491	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-HCS-136-1eb9,	No active collaboration found for task VDC_UPLOAD_OVF_CONTENTS(com.vmware.vcloud.entity.task:69e6a953-4c0b-4968-ab53-4c9d01ec2771).
2015-07-21 10:29:51,497	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-HCS-136-1eb9,	No enabled endpoints for task VDC_UPLOAD_OVF_CONTENTS(com.vmware.vcloud.entity.task:69e6a953-4c0b-4968-ab53-4c9d01ec2771). Skip pre-processing.
2015-07-21 10:29:51,733	4b391b61-7b1d-4e9d-bb96-10b321266837-HMS-56810-HCS-136-1eb9,	[valc] Creating VM "vm1" for vApp "vm1-aec35221-9dfb-43c7-8030-2e9f02271a7e" (0349811b-39a2-44c6-863b-ad9ff7208714) in VDC id=994a2a9d-46cd-4e37-844c-eade731d23a0, name=vdc_org1, enabled=true, fastProvisioning=true, thinProvisioning=true, org=[id=6567d203-5533-4761-94e8-a6f525fe0049, name=org1, displayName=org1], providerVdc=[id=551f4740-988f-41b1-8f1e-4dcdb32676b8, name=pvdc_Saptarishi_20-vc4-deploy, highestSupportedHwVersion=10, vcenter=[[Id = 14f01b6a-c3c2-4582-898b-628e5e8fd838, url = https://10.143.195.111/sdk, cell_instance_id = 21, uuid = 780924A1-C7A4-4FF4-87D2-61778SAAE238, status = READY]]], logicalResources=[[id=39d1691d-2b4c-4444-882a-124cc7a29e1e, name=*, lrType=STORAGE_CLASS, enabled=true, id=099dc3e1-e40e-4966-936b-b4fbc3940ba5, name=, lrType=NETWORK_POOL, enabled=true, id=2f745d74-e806-4de1-b0d2-d0e7b85bfa64, name=vdc_org1, lrType=COMPUTE, enabled=true, id=0e17b683-a61e-4114-9901-99c2fa299b0f, name=extnet_vdc_org1, lrType=NETWORK, enabled=true]] on storage class *(com.vmware.vcloud.entity.vdcstorageProfile:39d1691d-2b4c-4444-882a-124cc7a29e1e)

Figure 25 The output produced by the tool while tracing Configure Replication in VCD Secondary

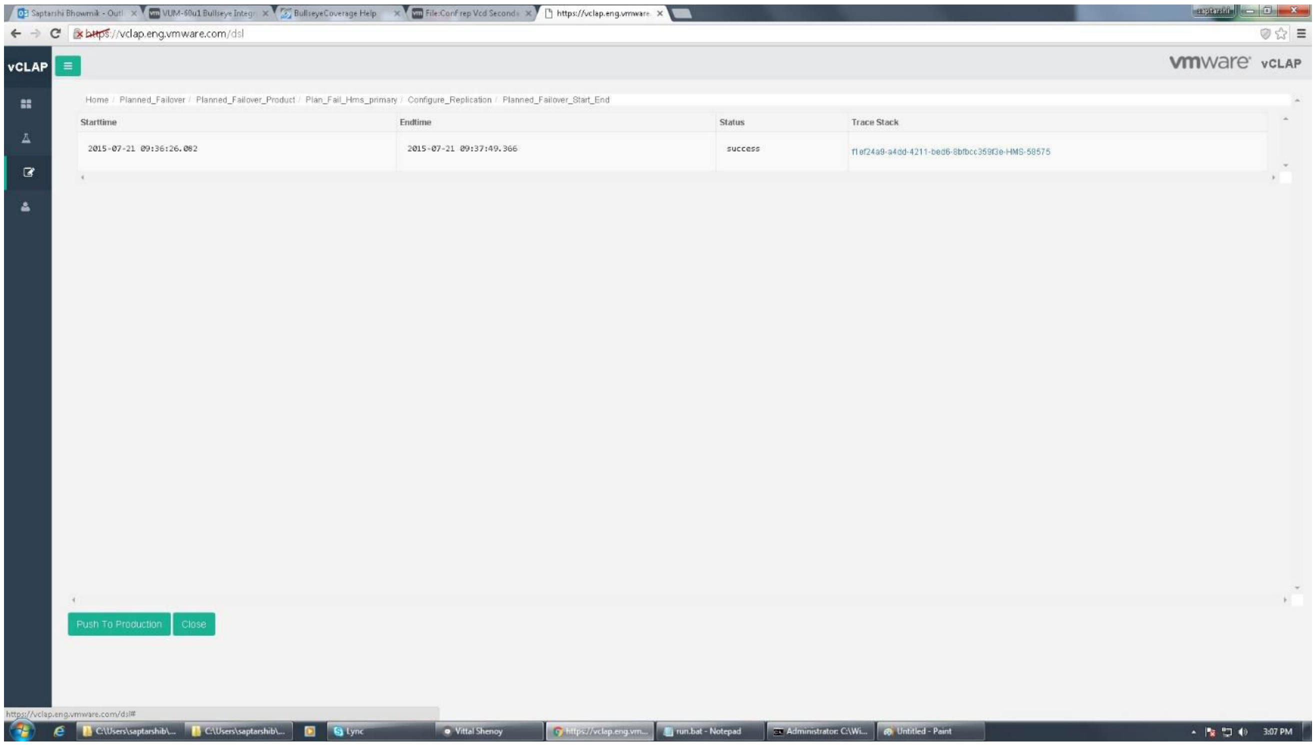
5.3.2 PLANNED FAILOVER OPERATION



Home / Planned_Failover

Vmname	Planfailover Operations
vm1	920e0bf2-e3b3-4fcc-823c-d29182eda6ae-HMS-58528,CGID-37f71af0-332c-455d-9c1c-7c7a0273b564
vm2	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575,CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2

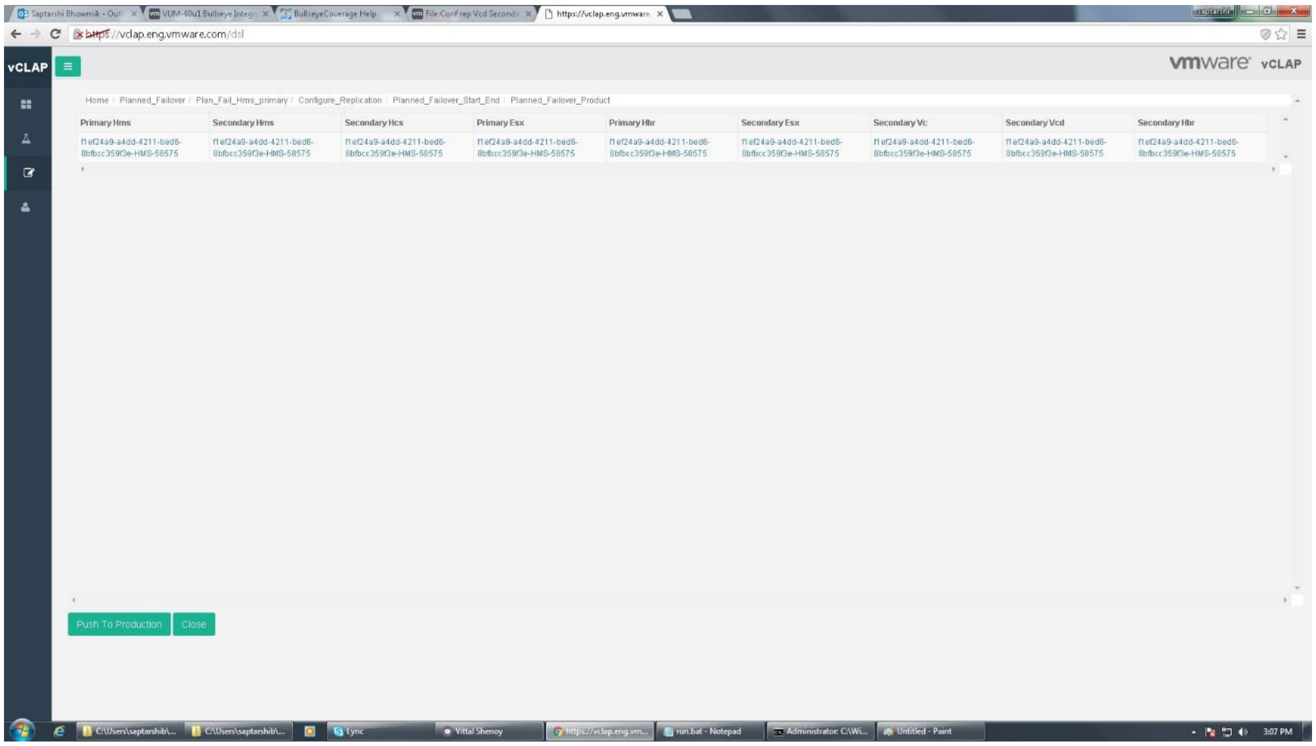
Figure 26 The Planned Failover page showing the names of the vm that performed that operation



Home / Planned_Failover / Planned_Failover_Product / Plan_Fail_Hms_primary / Configure_Replication / Planned_Failover_Start_End

Starttime	Endtime	Status	Trace Stack
2015-07-21 09:36:26.082	2015-07-21 09:37:49.366	success	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575

Figure 27 Showing the start time and end time of Planned Failover operation in a particular vm


Home / Planned_Failover / Plan_Fail_Hms_primary / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product

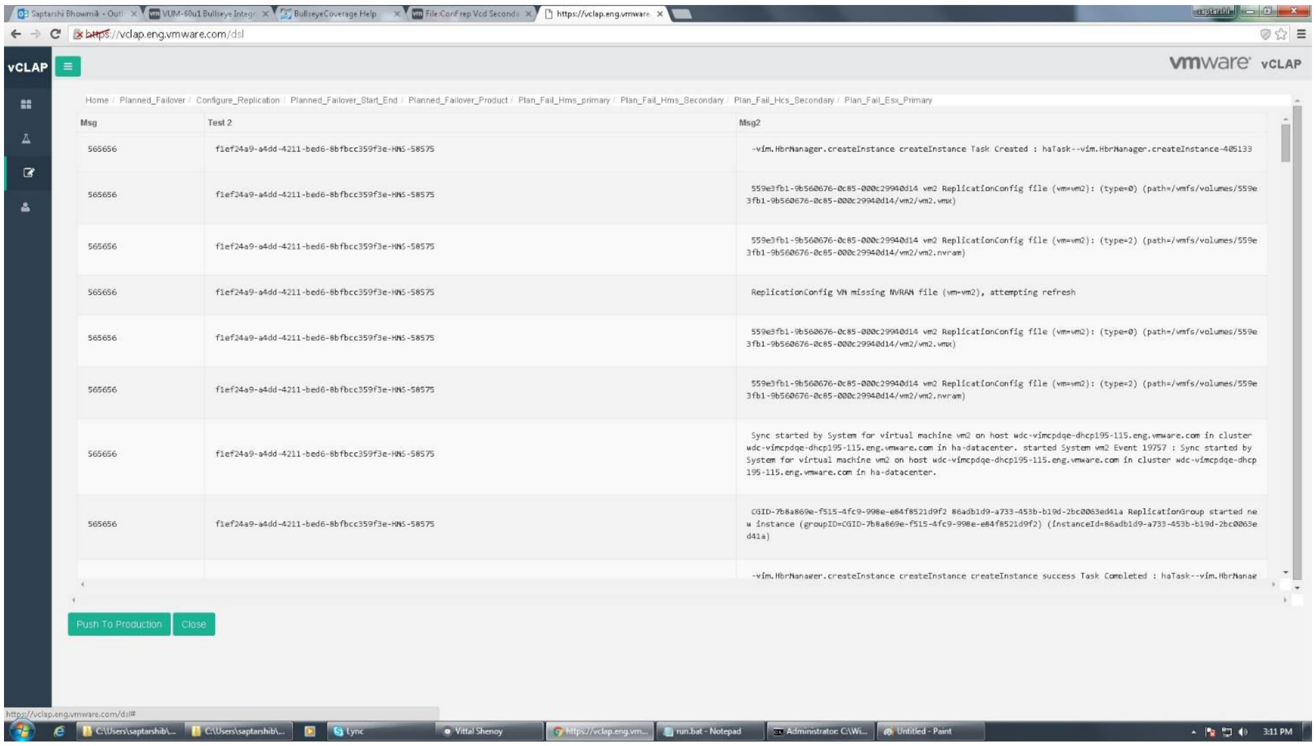
Primary Hms	Secondary Hms	Secondary Hcs	Primary Esx	Primary Hbr	Secondary Esx	Secondary Vc	Secondary Vcd	Secondary Hbr
f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575

Figure 28 The products that were involved in Planned Failover operation

In, Figure 26, The is the vm names along with the information that are unique to that vm, this information are a combination of operation ID, start time of that operation for that vm, end time of the operation for that vm etc. for Planned Failover operation is given. These information will help in further analysis of the logs in the database and extracting information specific to the vm.

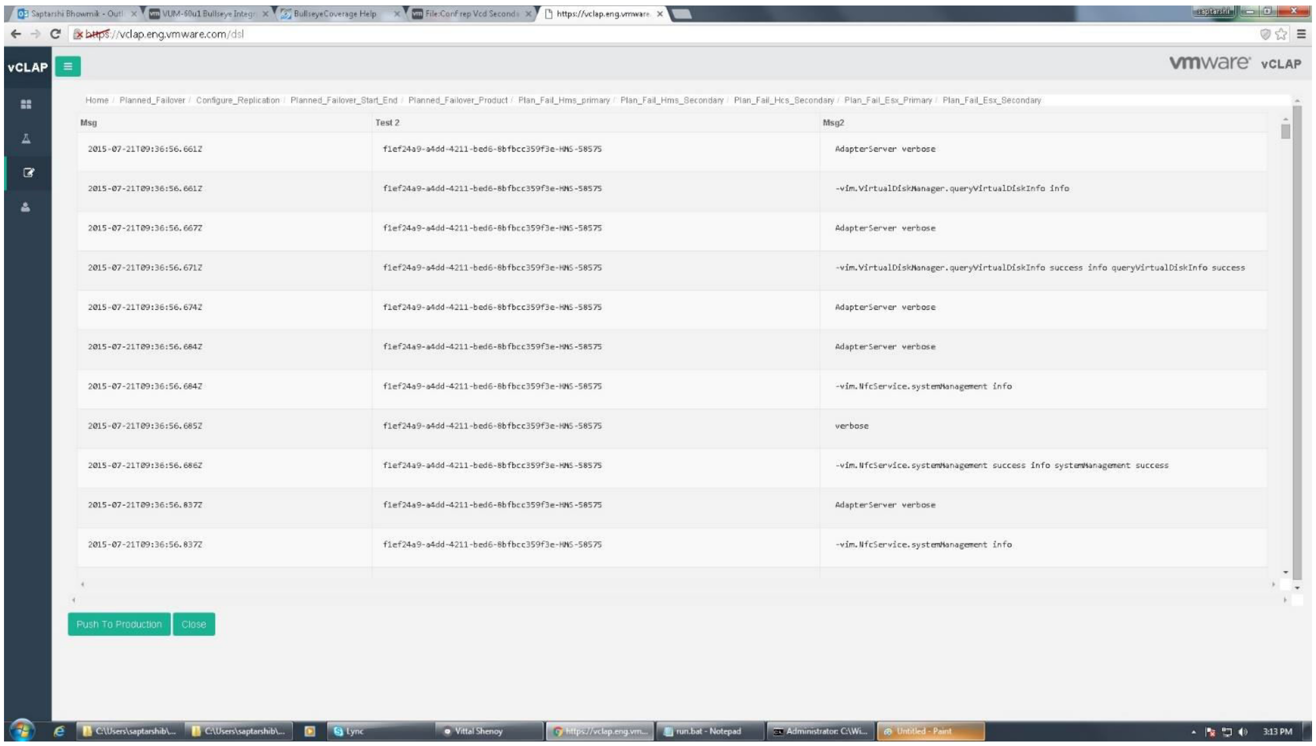
On clicking a particular vm in the above web page what we get is another web page with the start time end time, status and some information which are printed in the column of trace stack. Figure 27 shows how it looks like.

From there, clicking the trace stack column leads to another web page, which has the list of products, that are involved in the process of disaster recovery. Figure 28 clearly shows the result.


vCLAP
vmware vCLAP
Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary / Plan_Fail_Esx_Primary
Push To Production
Close

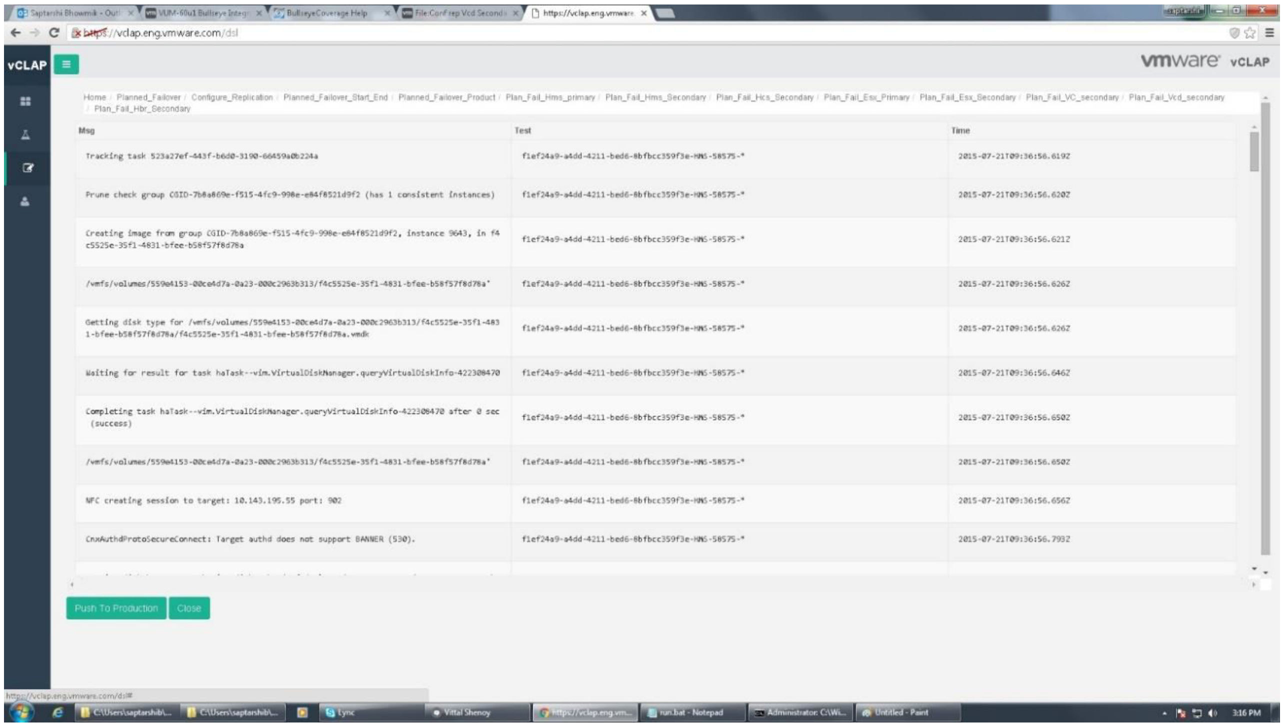
Msg	Test 2	Msg2
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.HbrManager.createInstance createInstance Task Created : haTask--vim.HbrManager.createInstance-405133
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	559e3fb1-9b560676-0c85-000c29940d14 vm2 ReplicationConfig file (vm=vm2): (type=0) (path=/vmfs/volumes/559e3fb1-9b560676-0c85-000c29940d14/vm2/vm2.vmx)
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	559e3fb1-9b560676-0c85-000c29940d14 vm2 ReplicationConfig file (vm=vm2): (type=2) (path=/vmfs/volumes/559e3fb1-9b560676-0c85-000c29940d14/vm2/vm2.nvram)
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	ReplicationConfig VM missing NVRAM file (vm=vm2), attempting refresh
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	559e3fb1-9b560676-0c85-000c29940d14 vm2 ReplicationConfig file (vm=vm2): (type=0) (path=/vmfs/volumes/559e3fb1-9b560676-0c85-000c29940d14/vm2/vm2.vmx)
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	559e3fb1-9b560676-0c85-000c29940d14 vm2 ReplicationConfig file (vm=vm2): (type=2) (path=/vmfs/volumes/559e3fb1-9b560676-0c85-000c29940d14/vm2/vm2.nvram)
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Sync started by System for virtual machine vm2 on host wdc-vimcpdqe-dhcp195-115.eng.vmware.com in cluster wdc-vimcpdqe-dhcp195-115.eng.vmware.com in ha-datacenter. started System vm2 Event 19757 : Sync started by System for virtual machine vm2 on host wdc-vimcpdqe-dhcp195-115.eng.vmware.com in cluster wdc-vimcpdqe-dhcp195-115.eng.vmware.com in ha-datacenter.
565656	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2 86adb1d9-a733-453b-b19d-2bc0063ed41a ReplicationGroup started new instance (groupID=CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2) (instanceId=86adb1d9-a733-453b-b19d-2bc0063ed41a)
		-vim.HbrManager.createInstance createInstance createInstance success Task Completed : haTask--vim.HbrManag

Figure 29 The output produced by the tool while tracing Planned Failover in ESXi Primary


vCLAP
vmware vCLAP
Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary / Plan_Fail_Esx_Primary / Plan_Fail_Esx_Secondary
Push To Production
Close

Msg	Test 2	Msg2
2015-07-21T09:36:56.661Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	AdapterServer verbose
2015-07-21T09:36:56.661Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.VirtualDiskManager.queryVirtualDiskInfo info
2015-07-21T09:36:56.667Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	AdapterServer verbose
2015-07-21T09:36:56.671Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.VirtualDiskManager.queryVirtualDiskInfo success info queryVirtualDiskInfo success
2015-07-21T09:36:56.674Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	AdapterServer verbose
2015-07-21T09:36:56.684Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	AdapterServer verbose
2015-07-21T09:36:56.684Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.NfcService.systemManagement info
2015-07-21T09:36:56.685Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	verbose
2015-07-21T09:36:56.686Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.NfcService.systemManagement success info systemManagement success
2015-07-21T09:36:56.837Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	AdapterServer verbose
2015-07-21T09:36:56.837Z	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	-vim.NfcService.systemManagement info

Figure 30 The output produced by the tool while tracing Planned Failover in ESXi Secondary

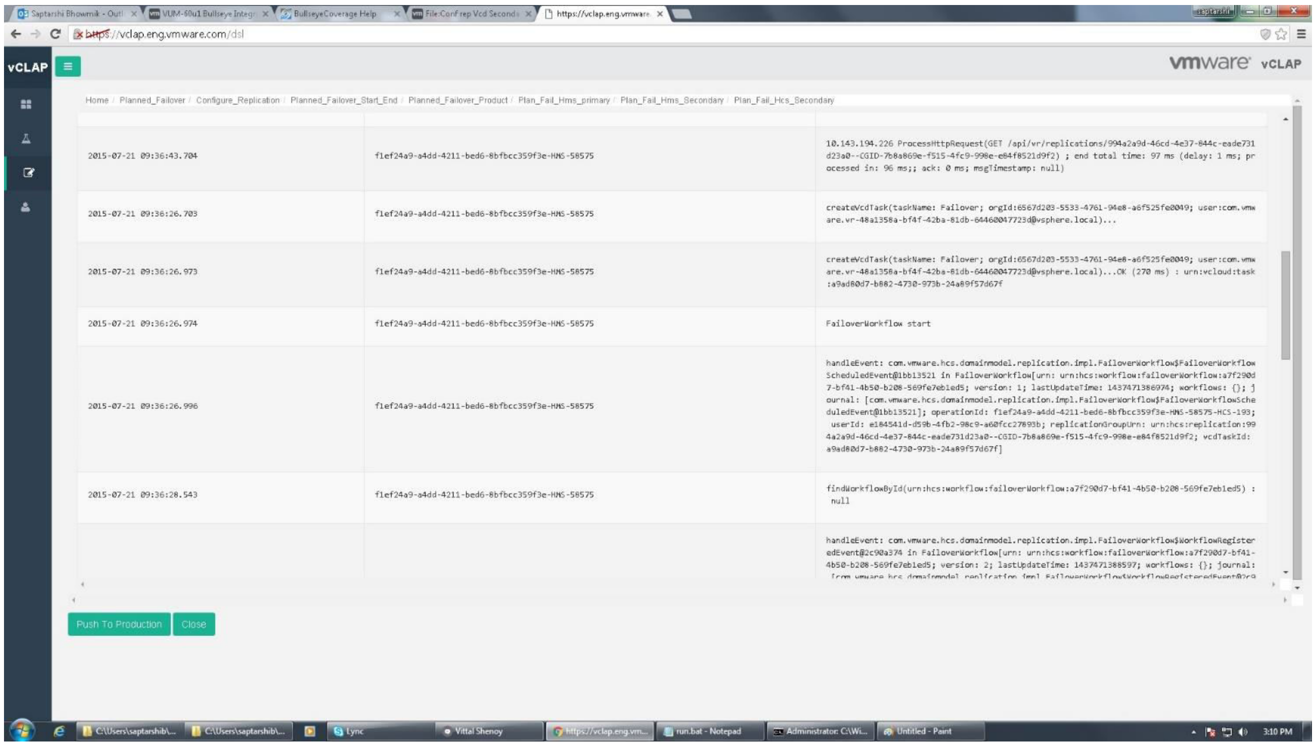


Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary / Plan_Fail_Esx_Primary / Plan_Fail_Esx_Secondary / Plan_Fail_VC_secondary / Plan_Fail_Vcd_secondary / Plan_Fail_Hbr_Secondary

Msg	Test	Time
Tracking task 523a27ef-443f-b6d0-3190-66459a0b224a	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.619Z
Prune check group CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2 (has 1 consistent instances)	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.620Z
Creating image from group CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2, instance 9643, in f4c5525e-35f1-4831-bfee-b58f57f8d78a	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.621Z
/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a'	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.626Z
Getting disk type for /vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a/f4c5525e-35f1-4831-bfee-b58f57f8d78a.vmdk	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.626Z
Waiting for result for task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422308470	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.646Z
Completing task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422308470 after 0 sec (success)	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.650Z
/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a'	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.650Z
NFC creating session to target: 10.143.195.55 port: 902	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.656Z
CnxAuthdProtoSecureConnect: Target authd does not support BANNER (530).	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-*	2015-07-21T09:36:56.793Z

Figure 31 The output produced by the tool while tracing Planned Failover in HBR Secondary

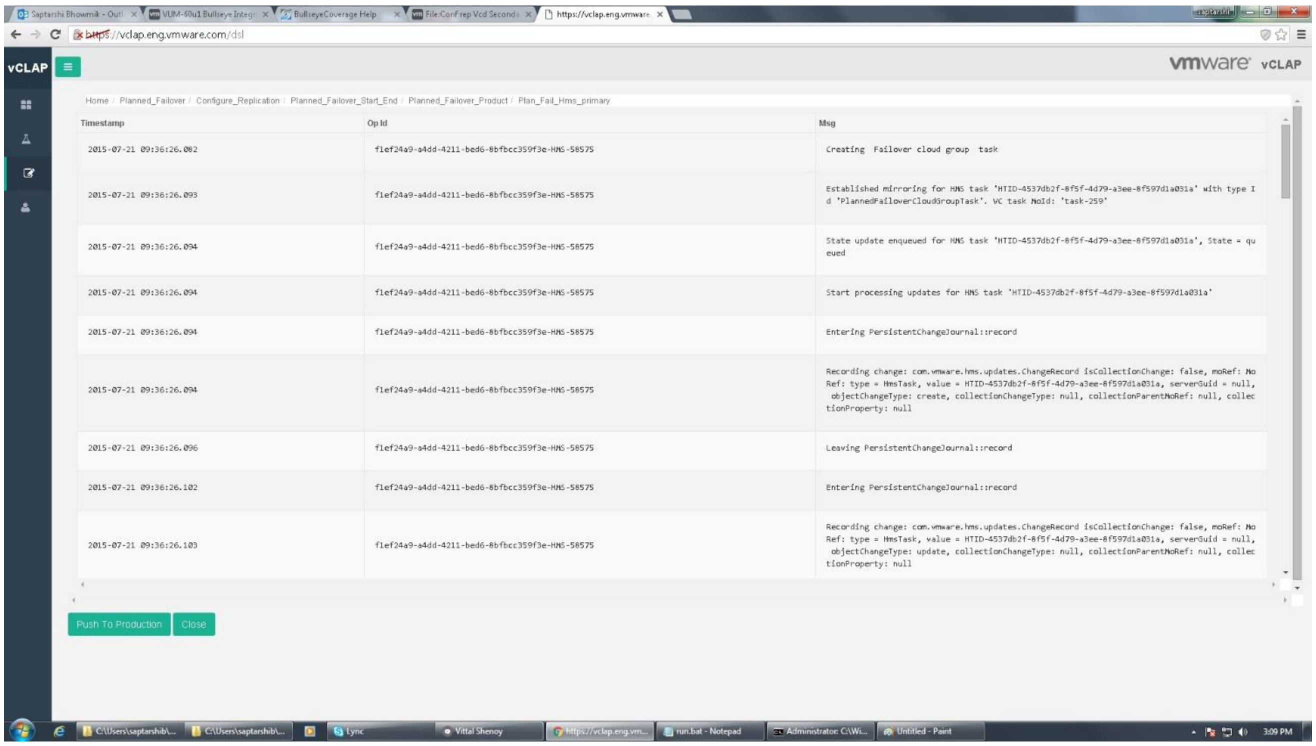
In, Figure 29 the result shows what record the ESXi Primary, which is the hypervisor on the first site or he protected site leaves when Planned Failover operation is done. Figure 30 the result shows what record the ESXi Secondary, which is the hypervisor on the recovery site leaves when Planned Failover operation is done. In, Figure 31 the result shows what record the HBR Secondary, leaves when Planned Failover operation is done. The tool gives the above output when the corresponding product is selected in the web page which displays the list of the product. Such as when the ESXi Primary is selected among the list of the products, the tool displays the ESXi Primary message that were present in the log files.



Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary

2015-07-21 09:36:43.704	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	10.143.194.226 ProcessHttpRequest(GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2) ; end total time: 97 ms (delay: 1 ms; processed in: 96 ms;; ack: 0 ms; msgTimestamp: null)
2015-07-21 09:36:26.703	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	createVcdTask(taskName: Failover; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...
2015-07-21 09:36:26.973	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	createVcdTask(taskName: Failover; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...OK (270 ms) : urn:vcloud:task:a9ad80d7-b882-4730-973b-24a89f57d67f
2015-07-21 09:36:26.974	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	FailoverWorkflow start
2015-07-21 09:36:26.996	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	handleEvent: com.vmware.hcs.domainmodel.replication.impl.FailoverWorkflow$FailoverWorkflowScheduledEvent@1bb13521 in FailoverWorkflow[urn: urn:hcs:workflow:failoverWorkflow:a7f290d7-bf41-4b50-b208-569fe7eb1ed5; version: 1; lastUpdateTime: 1437471386974; workflows: {}; journal: [com.vmware.hcs.domainmodel.replication.impl.FailoverWorkflow$FailoverWorkflowScheduledEvent@1bb13521]; operationId: f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193; userId: e184541d-d59b-4fb2-98c9-a60fcc27893b; replicationGroupUrn: urn:hcs:replication:994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2; vcdTaskId: a9ad80d7-b882-4730-973b-24a89f57d67f]
2015-07-21 09:36:28.543	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	findWorkflowById(urn:hcs:workflow:failoverWorkflow:a7f290d7-bf41-4b50-b208-569fe7eb1ed5) : null
		handleEvent: com.vmware.hcs.domainmodel.replication.impl.FailoverWorkflow$WorkflowRegisteredEvent@2c90a374 in FailoverWorkflow[urn: urn:hcs:workflow:failoverWorkflow:a7f290d7-bf41-4b50-b208-569fe7eb1ed5; version: 2; lastUpdateTime: 1437471388597; workflows: {}; journal:

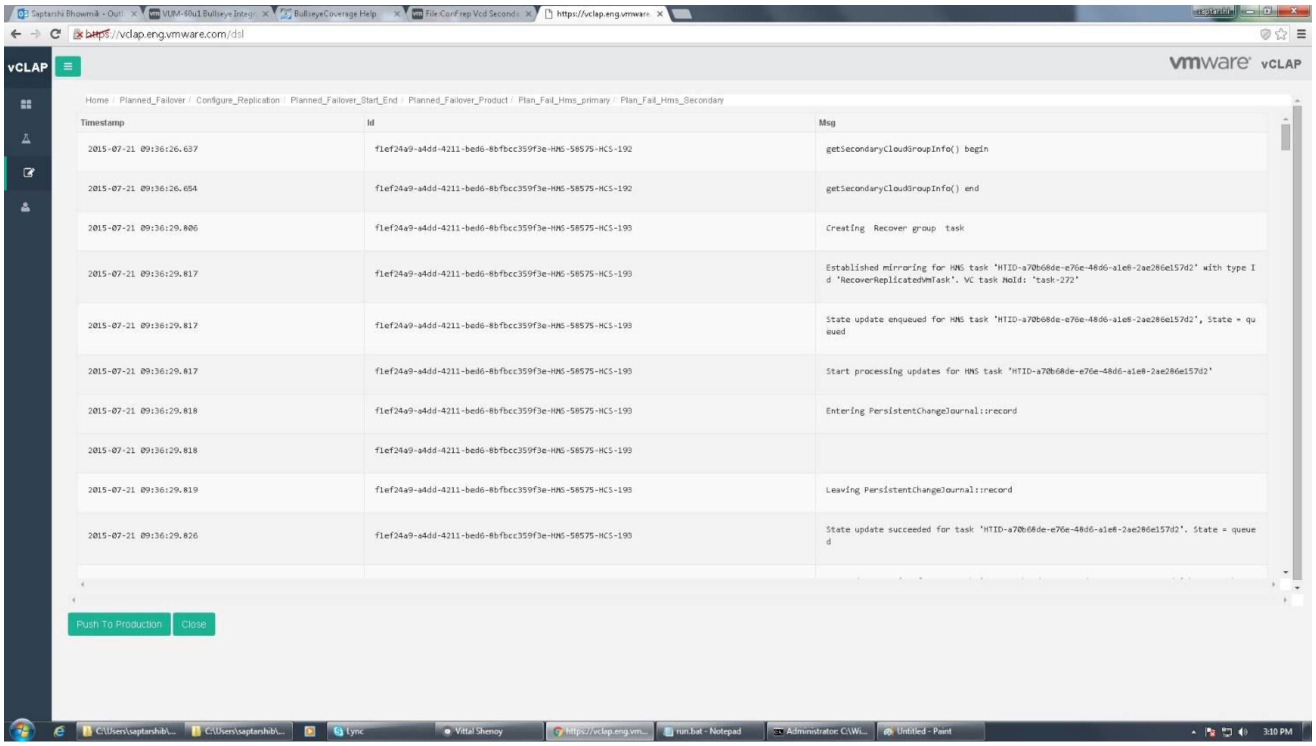
Figure 32 The output produced by the tool while tracing Planned Failover in HCS Secondary



Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary

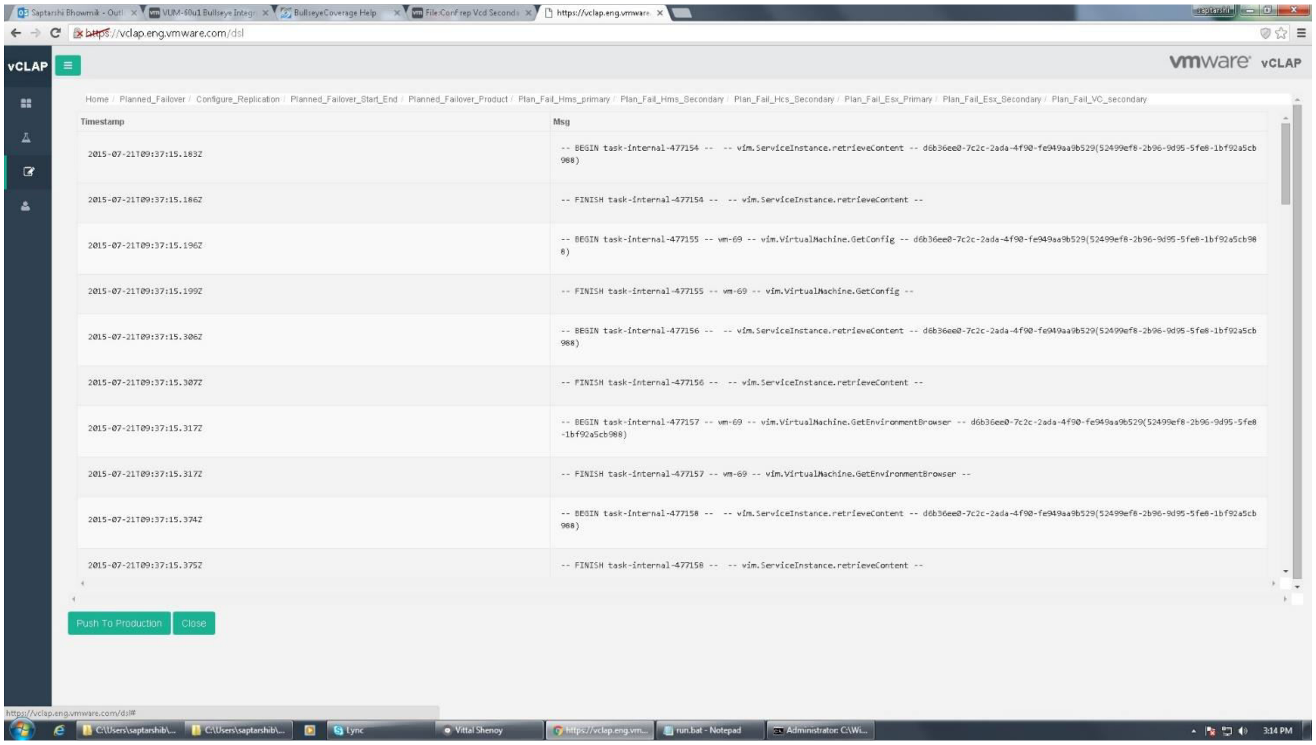
Timestamp	Op Id	Msg
2015-07-21 09:36:26.082	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Creating Failover cloud group task
2015-07-21 09:36:26.093	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Established mirroring for HMS task 'HTID-4537db2f-8f5f-4d79-a3ee-8f597d1a031a' with type Id 'PlannedFailoverCloudGroupTask'. VC task MoId: 'task-259'
2015-07-21 09:36:26.094	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	State update enqueued for HMS task 'HTID-4537db2f-8f5f-4d79-a3ee-8f597d1a031a', State = queued
2015-07-21 09:36:26.094	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Start processing updates for HMS task 'HTID-4537db2f-8f5f-4d79-a3ee-8f597d1a031a'
2015-07-21 09:36:26.094	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Entering PersistentChangeJournal::record
2015-07-21 09:36:26.094	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-4537db2f-8f5f-4d79-a3ee-8f597d1a031a, serverGuid = null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 09:36:26.096	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Leaving PersistentChangeJournal::record
2015-07-21 09:36:26.102	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Entering PersistentChangeJournal::record
2015-07-21 09:36:26.103	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-4537db2f-8f5f-4d79-a3ee-8f597d1a031a, serverGuid = null, objectChangeType: update, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null

Figure 33 The output produced by the tool while tracing Planned Failover in HMS Primary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary
Push To Production
Close

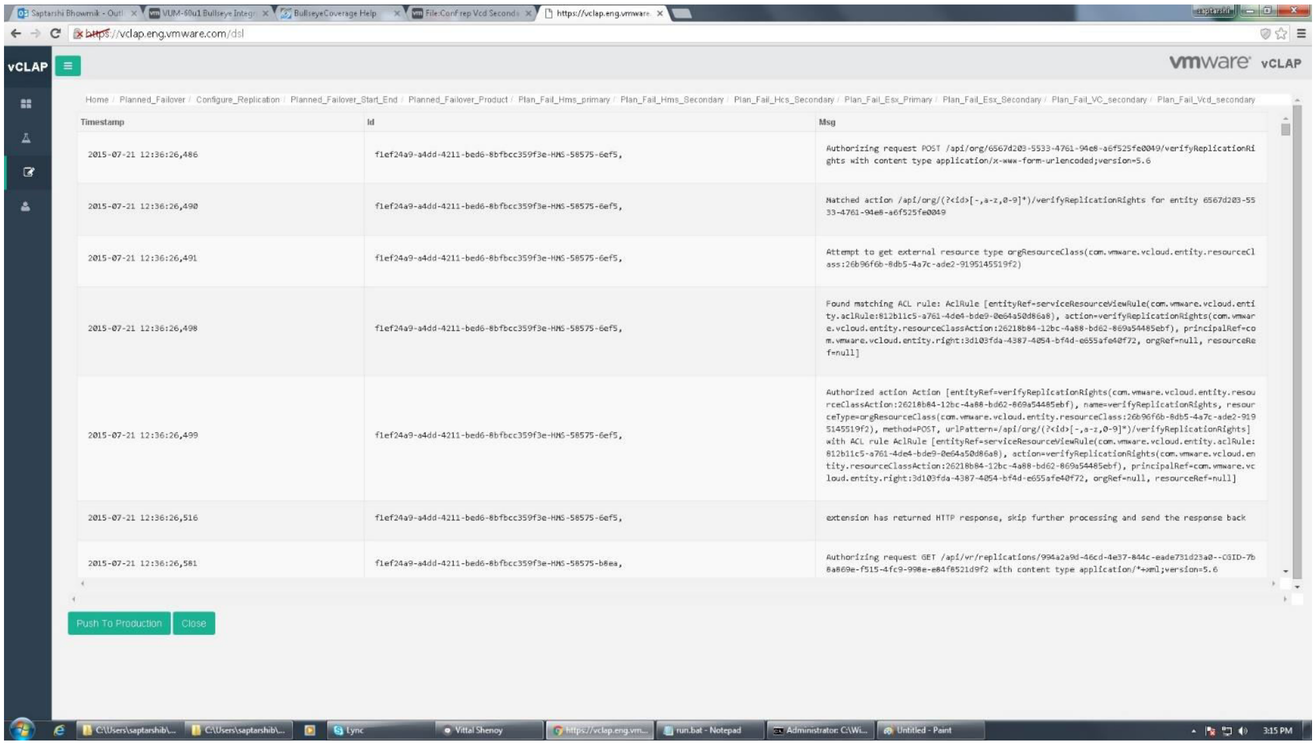
Timestamp	Id	Msg
2015-07-21 09:36:26.637	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-192	getSecondaryCloudGroupInfo() begin
2015-07-21 09:36:26.654	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-192	getSecondaryCloudGroupInfo() end
2015-07-21 09:36:29.806	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	Creating Recover group task
2015-07-21 09:36:29.817	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	Established mirroring for HMS task 'HTID-a70b68de-e76e-48d6-a1e8-2ae286e157d2' with type Id 'RecoverReplicatedVmTask'. VC task MoId: 'task-272'
2015-07-21 09:36:29.817	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	State update enqueued for HMS task 'HTID-a70b68de-e76e-48d6-a1e8-2ae286e157d2', State = queued
2015-07-21 09:36:29.817	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	Start processing updates for HMS task 'HTID-a70b68de-e76e-48d6-a1e8-2ae286e157d2'
2015-07-21 09:36:29.818	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	Entering PersistentChangeJournal::record
2015-07-21 09:36:29.818	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	
2015-07-21 09:36:29.819	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	Leaving PersistentChangeJournal::record
2015-07-21 09:36:29.826	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-HCS-193	State update succeeded for task 'HTID-a70b68de-e76e-48d6-a1e8-2ae286e157d2'. State = queued

Figure 34 The output produced by the tool while tracing Planned Failover in HMS Secondary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary / Plan_Fail_Esx_Primary / Plan_Fail_Esx_Secondary / Plan_Fail_VC_secondary
Push To Production
Close

Timestamp	Msg
2015-07-21T09:37:15.183Z	-- BEGIN task-internal-477154 -- -- vim.ServiceInstance.retrieveContent -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:37:15.186Z	-- FINISH task-internal-477154 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T09:37:15.196Z	-- BEGIN task-internal-477155 -- vm-69 -- vim.VirtualMachine.GetConfig -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:37:15.199Z	-- FINISH task-internal-477155 -- vm-69 -- vim.VirtualMachine.GetConfig --
2015-07-21T09:37:15.306Z	-- BEGIN task-internal-477156 -- -- vim.ServiceInstance.retrieveContent -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:37:15.307Z	-- FINISH task-internal-477156 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T09:37:15.317Z	-- BEGIN task-internal-477157 -- vm-69 -- vim.VirtualMachine.GetEnvironmentBrowser -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:37:15.317Z	-- FINISH task-internal-477157 -- vm-69 -- vim.VirtualMachine.GetEnvironmentBrowser --
2015-07-21T09:37:15.374Z	-- BEGIN task-internal-477158 -- -- vim.ServiceInstance.retrieveContent -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:37:15.375Z	-- FINISH task-internal-477158 -- -- vim.ServiceInstance.retrieveContent --

Figure 35 The output produced by the tool while tracing Planned Failover in VC Secondary



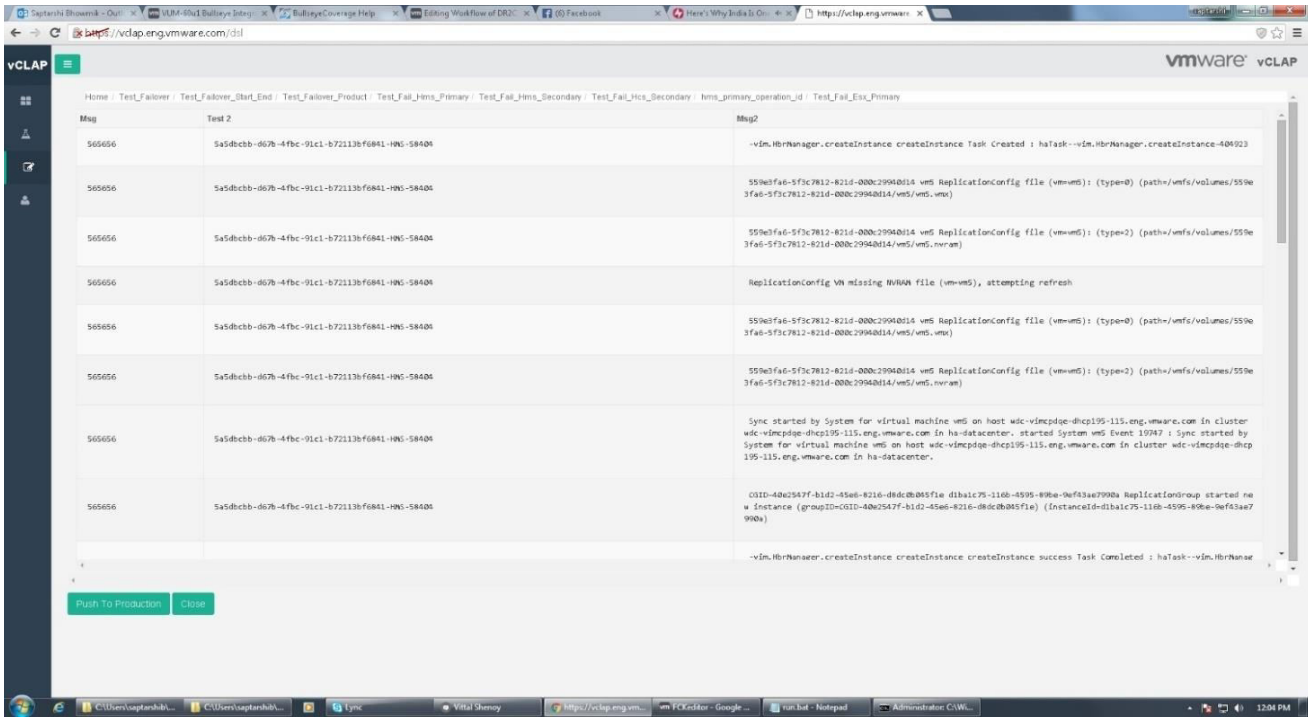
vCLAP
vmware vCLAP
Home / Planned_Failover / Configure_Replication / Planned_Failover_Start_End / Planned_Failover_Product / Plan_Fail_Hms_primary / Plan_Fail_Hms_Secondary / Plan_Fail_Hcs_Secondary / Plan_Fail_Esx_Primary / Plan_Fail_Esx_Secondary / Plan_Fail_VC_secondary / Plan_Fail_Vcd_secondary
Push To Production
Close

Timestamp	Id	Msg
2015-07-21 12:36:26,486	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	Authorizing request POST /api/org/6567d203-5533-4761-94e8-a6f525fe0049/verifyReplicationRights with content type application/x-www-form-urlencoded;version=5.6
2015-07-21 12:36:26,490	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	Matched action /api/org/(?<id>[-,a-z,0-9]*)/verifyReplicationRights for entity 6567d203-5533-4761-94e8-a6f525fe0049
2015-07-21 12:36:26,491	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	Attempt to get external resource type orgResourceClass(com.vmware.vcloud.entity.resourceClass:26b96f6b-8db5-4a7c-ade2-9195145519f2)
2015-07-21 12:36:26,498	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	Found matching ACL rule: AclRule [entityRef=serviceResourceViewRule(com.vmware.vcloud.entity.aclRule:812b11c5-a761-4de4-bde9-0e64a50d86a8), action=verifyReplicationRights(com.vmware.vcloud.entity.resourceClassAction:26218b84-12bc-4a88-bd62-869a54485ebf), principalRef=com.vmware.vcloud.entity.right:3d103fda-4387-4054-bf4d-e655afe40f72, orgRef=null, resourceRef=null]
2015-07-21 12:36:26,499	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	Authorized action Action [entityRef=verifyReplicationRights(com.vmware.vcloud.entity.resourceClassAction:26218b84-12bc-4a88-bd62-869a54485ebf), name=verifyReplicationRights, resourceType=orgResourceClass(com.vmware.vcloud.entity.resourceClass:26b96f6b-8db5-4a7c-ade2-9195145519f2), method=POST, urlPattern=/api/org/(?<id>[-,a-z,0-9]*)/verifyReplicationRights] with ACL rule AclRule [entityRef=serviceResourceViewRule(com.vmware.vcloud.entity.aclRule:812b11c5-a761-4de4-bde9-0e64a50d86a8), action=verifyReplicationRights(com.vmware.vcloud.entity.resourceClassAction:26218b84-12bc-4a88-bd62-869a54485ebf), principalRef=com.vmware.vcloud.entity.right:3d103fda-4387-4054-bf4d-e655afe40f72, orgRef=null, resourceRef=null]
2015-07-21 12:36:26,516	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-6ef5,	extension has returned HTTP response, skip further processing and send the response back
2015-07-21 12:36:26,581	f1ef24a9-a4dd-4211-bed6-8bfbcc359f3e-HMS-58575-b8ea,	Authorizing request GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2 with content type application/*+xml;version=5.6

Figure 36 The output produced by the tool while tracing Planned Failover in VCD Secondary

Figure 32 shows what record the HCS leaves, when Planned Failover operation is done. Figure 35 shows the record left by vCenter on the recovery site and Figure 36 shows the log records of Planned Failover on vCloud Director. In planned Recovery also HMS Primary and HMS Secondary leaves traces in the log files. The tool extracts those information from the log files in the database and properly shows them in a tabular format in a web page. Operation ID is displayed along with the log messages because operation ID are unique not only to a product but they are unique to the operation as well.

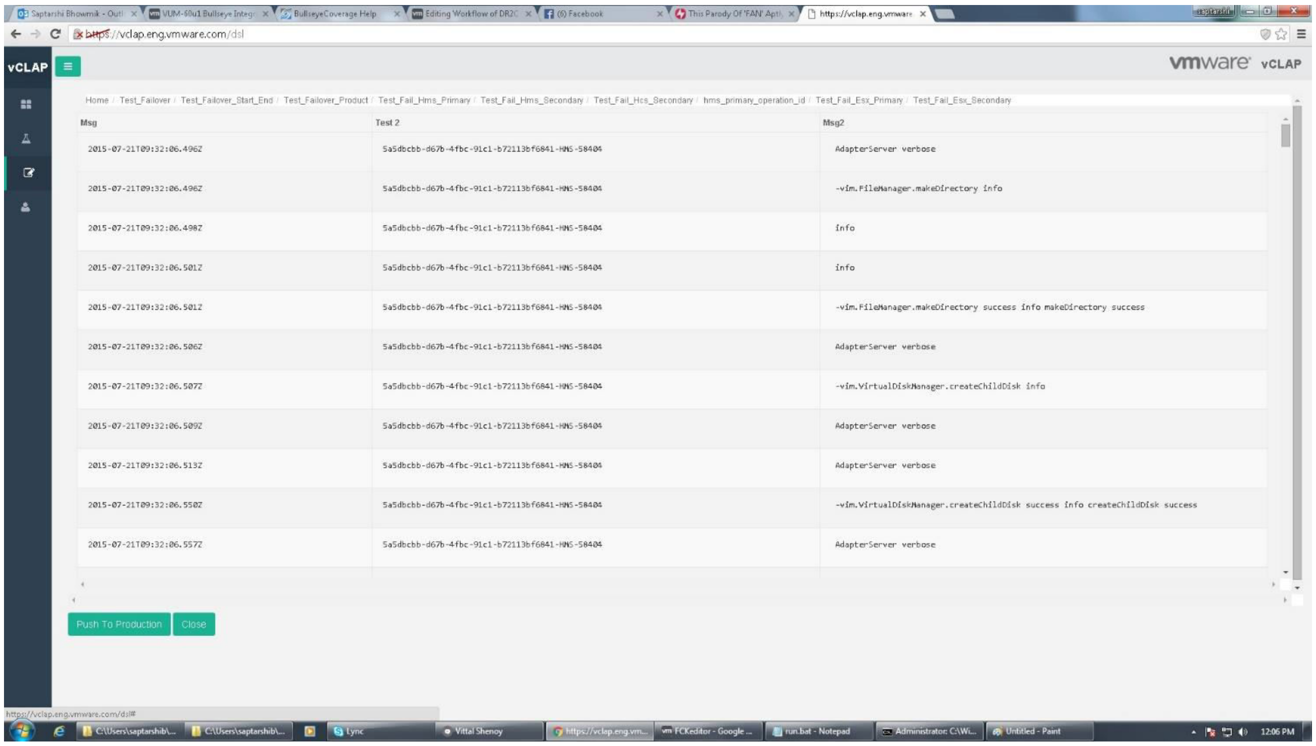
5.3.3 TEST FAILOVER OPERATION



Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary / hms_primary_operation_id / Test_Fail_Esx_Primary

Msg	Test 2	Msg2
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-vim.HbrManager.createInstance createInstance Task Created : haTask--vim.HbrManager.createInstance-404923
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	559e3fa6-5f3c7812-821d-000c29940d14 vm5 ReplicationConfig file (vm=vm5): (type=0) (path=/vmfs/volumes/559e3fa6-5f3c7812-821d-000c29940d14/vm5/vm5.vmx)
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	559e3fa6-5f3c7812-821d-000c29940d14 vm5 ReplicationConfig file (vm=vm5): (type=2) (path=/vmfs/volumes/559e3fa6-5f3c7812-821d-000c29940d14/vm5/vm5.nvram)
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	ReplicationConfig VM missing NVRAM file (vm=vm5), attempting refresh
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	559e3fa6-5f3c7812-821d-000c29940d14 vm5 ReplicationConfig file (vm=vm5): (type=0) (path=/vmfs/volumes/559e3fa6-5f3c7812-821d-000c29940d14/vm5/vm5.vmx)
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	559e3fa6-5f3c7812-821d-000c29940d14 vm5 ReplicationConfig file (vm=vm5): (type=2) (path=/vmfs/volumes/559e3fa6-5f3c7812-821d-000c29940d14/vm5/vm5.nvram)
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Sync started by System for virtual machine vm5 on host wdc-vimcpdqe-dhcp195-115.eng.vmware.com in cluster wdc-vimcpdqe-dhcp195-115.eng.vmware.com in ha-datacenter. started System vm5 Event 19747 : Sync started by System for virtual machine vm5 on host wdc-vimcpdqe-dhcp195-115.eng.vmware.com in cluster wdc-vimcpdqe-dhcp195-115.eng.vmware.com in ha-datacenter.
565656	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e d1ba1c75-116b-4595-89be-9ef43ae7990a ReplicationGroup started new instance (groupID=CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e) (instanceId=d1ba1c75-116b-4595-89be-9ef43ae7990a)
		-vim.HbrManager.createInstance createInstance createInstance success Task Completed : haTask--vim.HbrManag

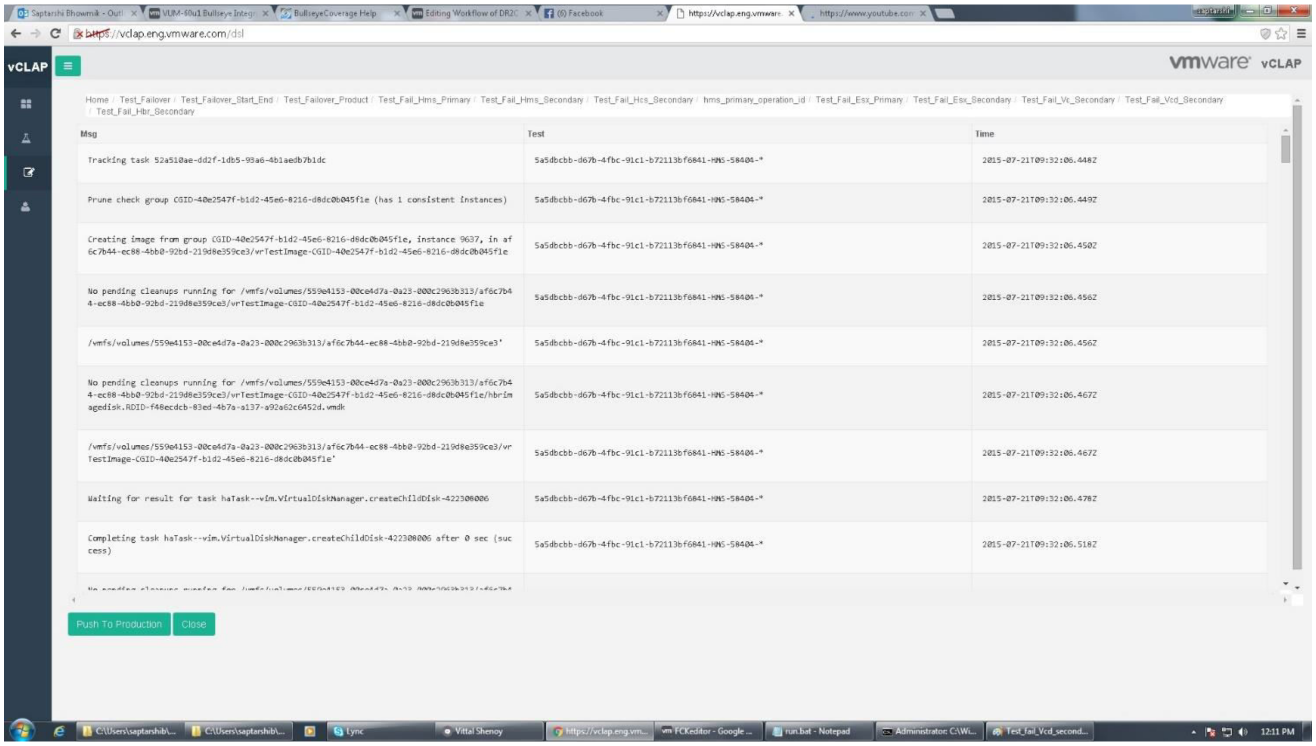
Figure 37 The output produced by the tool while tracing Test Failover in ESXi Primary



Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary / hms_primary_operation_id / Test_Fail_Esx_Primary / Test_Fail_Esx_Secondary

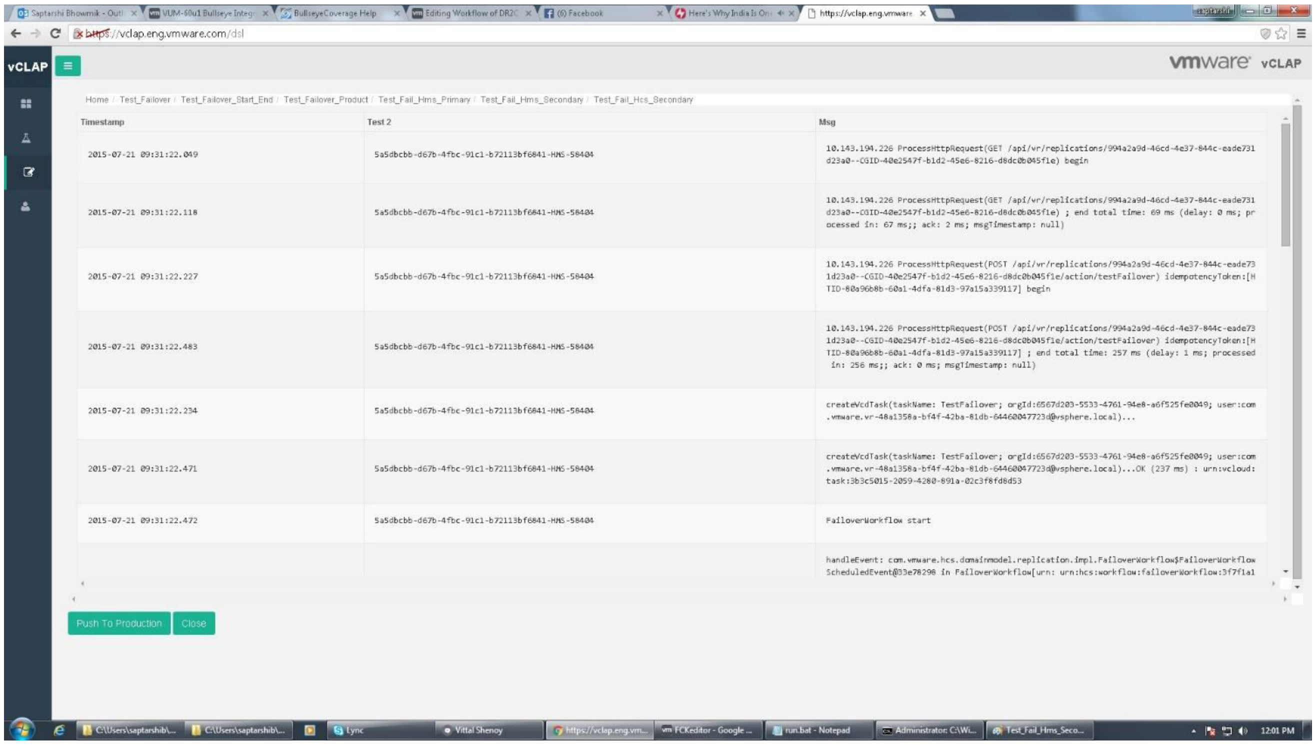
Msg	Test 2	Msg2
2015-07-21T09:32:06.496Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	AdapterServer verbose
2015-07-21T09:32:06.496Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-vim.FileManager.makeDirectory info
2015-07-21T09:32:06.498Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	info
2015-07-21T09:32:06.501Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	info
2015-07-21T09:32:06.501Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-vim.FileManager.makeDirectory success info makeDirectory success
2015-07-21T09:32:06.506Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	AdapterServer verbose
2015-07-21T09:32:06.507Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-vim.VirtualDiskManager.createChildDisk info
2015-07-21T09:32:06.509Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	AdapterServer verbose
2015-07-21T09:32:06.513Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	AdapterServer verbose
2015-07-21T09:32:06.550Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-vim.VirtualDiskManager.createChildDisk success info createChildDisk success
2015-07-21T09:32:06.557Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	AdapterServer verbose

Figure 38 The output produced by the tool while tracing Test Failover in ESXi Secondary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary / hms_primary_operation_id / Test_Fail_Esx_Primary / Test_Fail_Esx_Secondary / Test_Fail_Vc_Secondary / Test_Fail_Vcd_Secondary / Test_Fail_Hbr_Secondary
Push To Production
Close

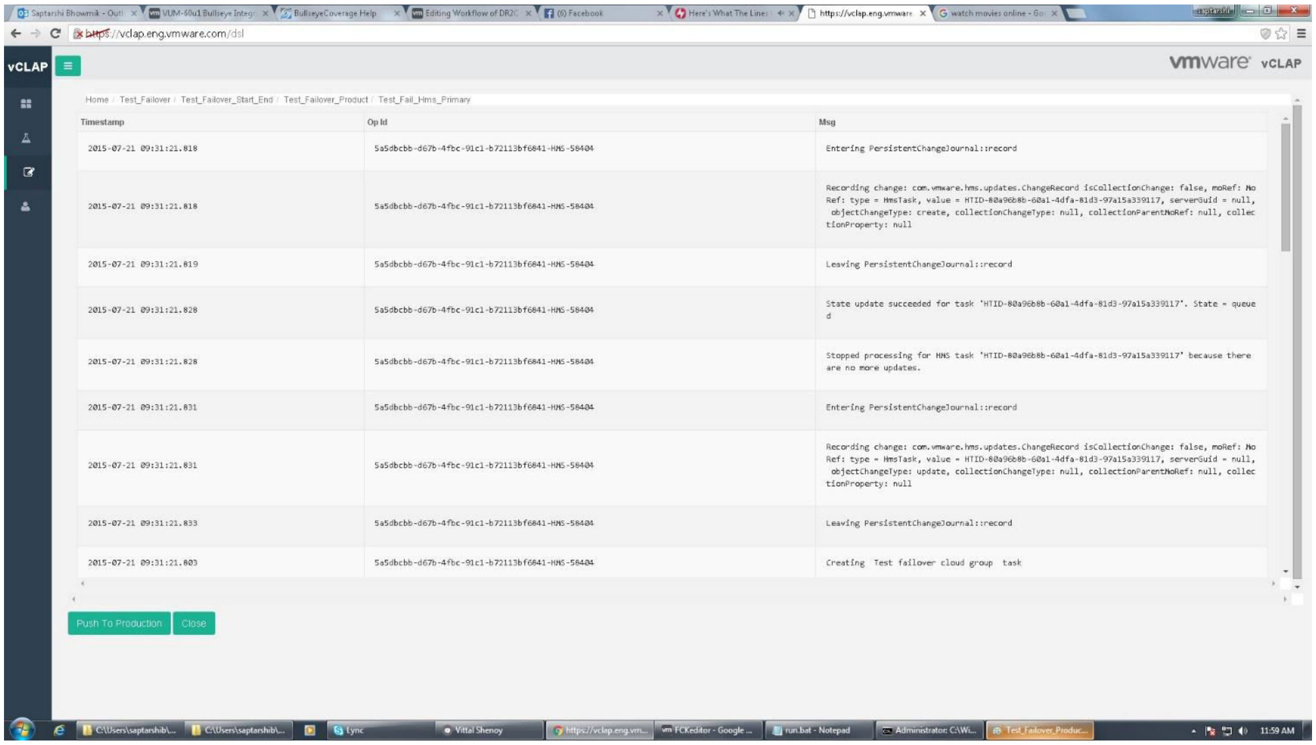
Msg	Test	Time
Tracking task 52a510ae-dd2f-1db5-93a6-4b1aedb7b1dc	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.448Z
Prune check group CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e (has 1 consistent instances)	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.449Z
Creating image from group CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e, instance 9637, in af6c7b44-ec88-4bb0-92bd-219d8e359ce3/vrTestImage-CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.450Z
No pending cleanups running for /vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/af6c7b44-ec88-4bb0-92bd-219d8e359ce3/vrTestImage-CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.456Z
/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/af6c7b44-ec88-4bb0-92bd-219d8e359ce3'	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.456Z
No pending cleanups running for /vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/af6c7b44-ec88-4bb0-92bd-219d8e359ce3/vrTestImage-CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e/hbrimagedisk.RDID-f48ecdcb-83ed-4b7a-a137-a92a62c6452d.vmdk	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.467Z
/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/af6c7b44-ec88-4bb0-92bd-219d8e359ce3/vrTestImage-CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e'	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.467Z
Waiting for result for task haTask--vim.VirtualDiskManager.createChildDisk-422308006	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.478Z
Completing task haTask--vim.VirtualDiskManager.createChildDisk-422308006 after 0 sec (success)	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-*	2015-07-21T09:32:06.518Z
No pending cleanups running for /vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/af6c7b4		

Figure 39 The output produced by the tool while tracing Test Failover in HBR Secondary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary
Push To Production
Close

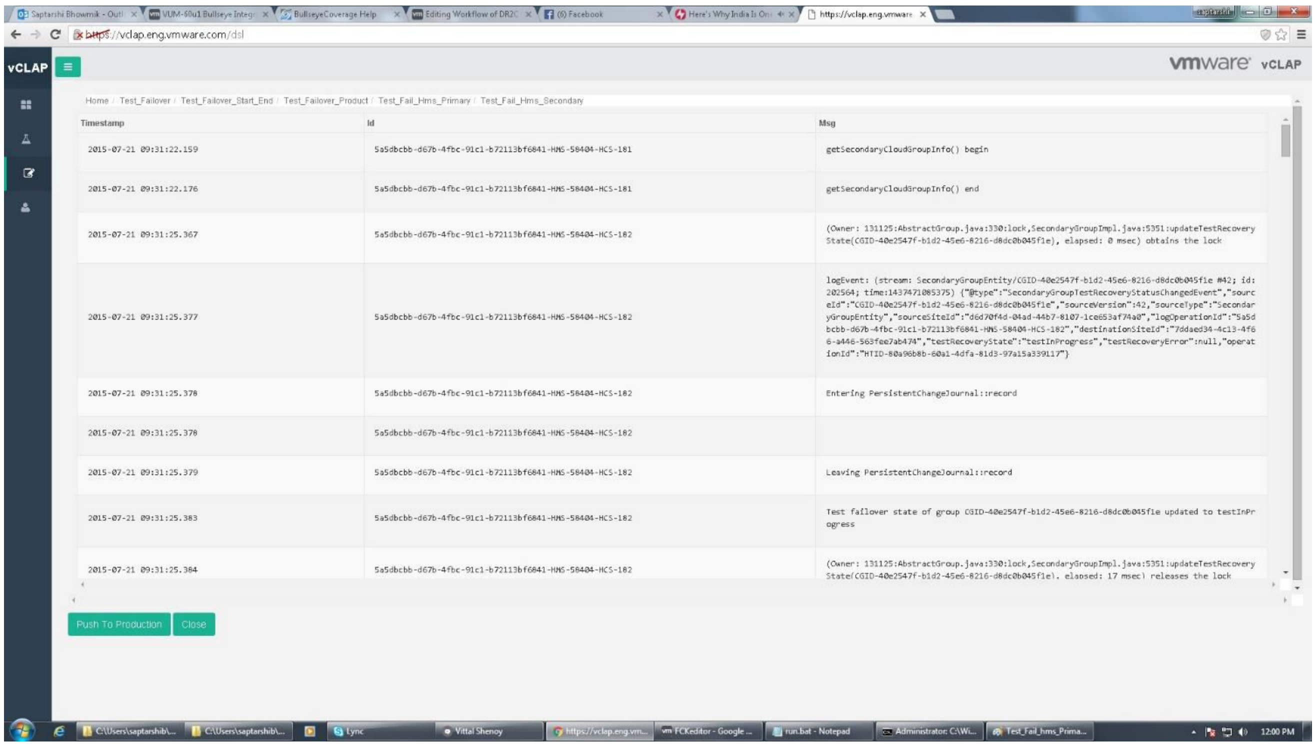
Timestamp	Test 2	Msg
2015-07-21 09:31:22.049	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	10.143.194.226 ProcessHttpRequest(GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e) begin
2015-07-21 09:31:22.118	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	10.143.194.226 ProcessHttpRequest(GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e) ; end total time: 69 ms (delay: 0 ms; processed in: 67 ms;; ack: 2 ms; msgTimestamp: null)
2015-07-21 09:31:22.227	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	10.143.194.226 ProcessHttpRequest(POST /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e/action/testFailover) idempotencyToken:[HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117] begin
2015-07-21 09:31:22.483	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	10.143.194.226 ProcessHttpRequest(POST /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e/action/testFailover) idempotencyToken:[HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117] ; end total time: 257 ms (delay: 1 ms; processed in: 256 ms;; ack: 0 ms; msgTimestamp: null)
2015-07-21 09:31:22.234	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	createVcdTask(taskName: TestFailover; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...
2015-07-21 09:31:22.471	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	createVcdTask(taskName: TestFailover; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...OK (237 ms) : urn:vcloud:task:3b3c5015-2059-4280-891a-02c3f8fd8d53
2015-07-21 09:31:22.472	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	FailoverWorkflow start
		handleEvent: com.vmware.hcs.domainmodel.replication.impl.FailoverWorkflow$FailoverWorkflowScheduledEvent@33e78298 in FailoverWorkflow[urn: urn:hcs:workflow:failoverWorkflow:3f7f1a1

Figure 40 The output produced by the tool while tracing Test Failover in HCS Secondary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary
Push To Production
Close

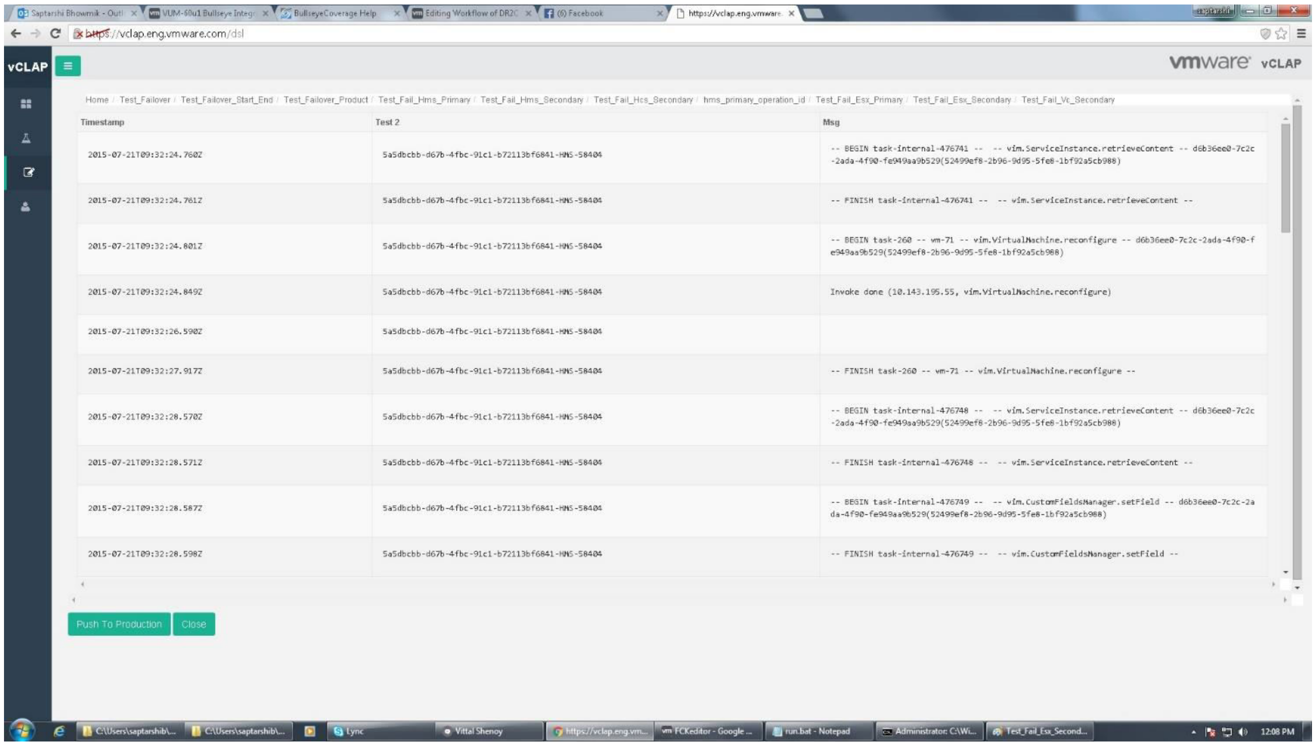
Timestamp	Op Id	Msg
2015-07-21 09:31:21.818	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Entering PersistentChangeJournal::record
2015-07-21 09:31:21.818	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117, serverGuid = null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 09:31:21.819	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Leaving PersistentChangeJournal::record
2015-07-21 09:31:21.828	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	State update succeeded for task 'HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117'. State = queued
2015-07-21 09:31:21.828	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Stopped processing for HMS task 'HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117' because there are no more updates.
2015-07-21 09:31:21.831	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Entering PersistentChangeJournal::record
2015-07-21 09:31:21.831	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117, serverGuid = null, objectChangeType: update, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 09:31:21.833	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Leaving PersistentChangeJournal::record
2015-07-21 09:31:21.803	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Creating Test failover cloud group task

Figure 41 The output produced by the tool while tracing Test Failover in HMS Primary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary
Push To Production
Close

Timestamp	Id	Msg
2015-07-21 09:31:22.159	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-181	getSecondaryCloudGroupInfo() begin
2015-07-21 09:31:22.176	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-181	getSecondaryCloudGroupInfo() end
2015-07-21 09:31:25.367	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	(Owner: 131125:AbstractGroup.java:330:lock,SecondaryGroupImpl.java:5351:updateTestRecoveryState(CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e), elapsed: 0 msec) obtains the lock
2015-07-21 09:31:25.377	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	logEvent: (stream: SecondaryGroupEntity/CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e #42; id: 202564; time:1437471085375) {"@type":"SecondaryGroupTestRecoveryStatusChangedEvent","sourceId":"CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e","sourceVersion":42,"sourceType":"SecondaryGroupEntity","sourceSiteId":"d6d70f4d-04ad-44b7-8107-1ce653af74a0","logOperationId":"5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182","destinationSiteId":"7ddaed34-4c13-4f66-a446-563fee7ab474","testRecoveryState":"testInProgress","testRecoveryError":null,"operationId":"HTID-80a96b8b-60a1-4dfa-81d3-97a15a339117"}
2015-07-21 09:31:25.378	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	Entering PersistentChangeJournal::record
2015-07-21 09:31:25.378	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	
2015-07-21 09:31:25.379	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	Leaving PersistentChangeJournal::record
2015-07-21 09:31:25.383	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	Test failover state of group CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e updated to testInProgress
2015-07-21 09:31:25.384	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-HCS-182	(Owner: 131125:AbstractGroup.java:330:lock,SecondaryGroupImpl.java:5351:updateTestRecoveryState(CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e), elapsed: 17 msec) releases the lock

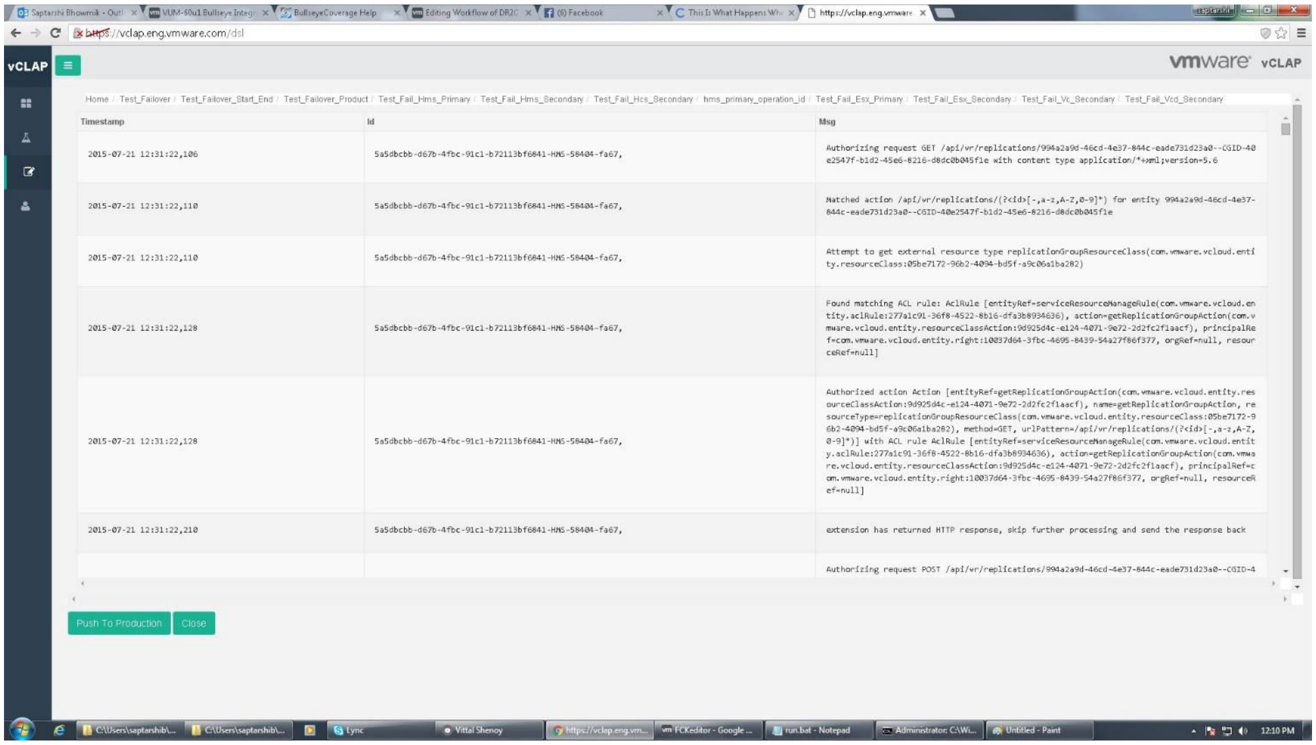
Figure 42 The output produced by the tool while tracing Test Failover in HMS Secondary



Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary / hms_primary_operation_id / Test_Fail_Esx_Primary / Test_Fail_Esx_Secondary / Test_Fail_Vc_Secondary

Timestamp	Test 2	Msg
2015-07-21T09:32:24.760Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- BEGIN task-internal-476741 -- -- vim.ServiceInstance.retrieveContent -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:32:24.761Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- FINISH task-internal-476741 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T09:32:24.801Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- BEGIN task-260 -- vm-71 -- vim.VirtualMachine.reconfigure -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:32:24.849Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	Invoke done (10.143.195.55, vim.VirtualMachine.reconfigure)
2015-07-21T09:32:26.590Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	
2015-07-21T09:32:27.917Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- FINISH task-260 -- vm-71 -- vim.VirtualMachine.reconfigure --
2015-07-21T09:32:28.570Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- BEGIN task-internal-476748 -- -- vim.ServiceInstance.retrieveContent -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:32:28.571Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- FINISH task-internal-476748 -- -- vim.ServiceInstance.retrieveContent --
2015-07-21T09:32:28.587Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- BEGIN task-internal-476749 -- -- vim.CustomFieldsManager.setField -- d6b36ee0-7c2c-2ada-4f90-fe949aa9b529(52499ef8-2b96-9d95-5fe8-1bf92a5cb988)
2015-07-21T09:32:28.598Z	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404	-- FINISH task-internal-476749 -- -- vim.CustomFieldsManager.setField --

Figure 43 The output produced by the tool while tracing Test Failover in VC Secondary



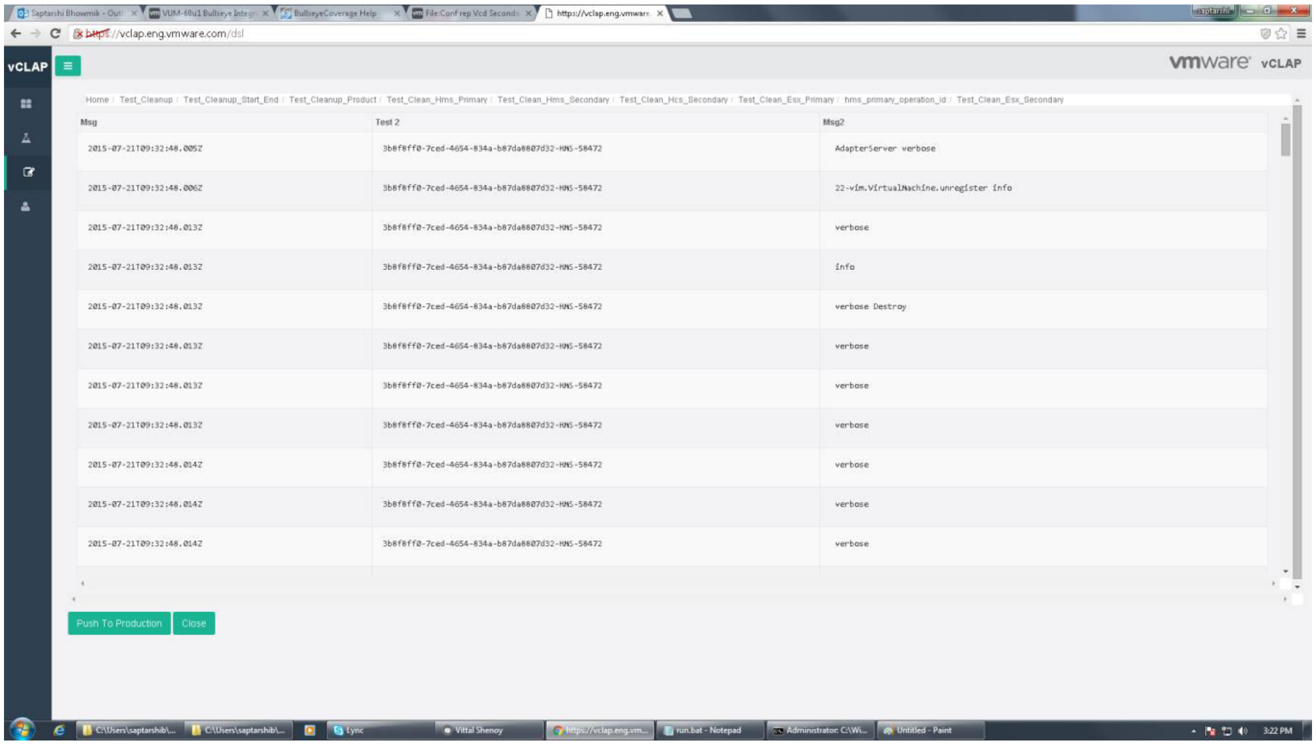
Home / Test_Failover / Test_Failover_Start_End / Test_Failover_Product / Test_Fail_Hms_Primary / Test_Fail_Hms_Secondary / Test_Fail_Hcs_Secondary / hms_primary_operation_id / Test_Fail_Esx_Primary / Test_Fail_Esx_Secondary / Test_Fail_Vc_Secondary / Test_Fail_Vcd_Secondary

Timestamp	Id	Msg
2015-07-21 12:31:22,106	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	Authorizing request GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e with content type application/*+xml;version=5.6
2015-07-21 12:31:22,110	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	Matched action /api/vr/replications/(?<id>[-,a-z,A-Z,0-9]*) for entity 994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-40e2547f-b1d2-45e6-8216-d8dc0b045f1e
2015-07-21 12:31:22,110	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	Attempt to get external resource type replicationGroupResourceClass(com.vmware.vcloud.entity.resourceClass:05be7172-96b2-4094-bd5f-a9c06a1ba282)
2015-07-21 12:31:22,128	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	Found matching ACL rule: AclRule [entityRef=serviceResourceManageRule(com.vmware.vcloud.entity.aclRule:277a1c91-36f8-4522-8b16-dfa3b8934636), action=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), principalRef=com.vmware.vcloud.entity.right:10037d64-3fbc-4695-8439-54a27f86f377, orgRef=null, resourceRef=null]
2015-07-21 12:31:22,128	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	Authorized action Action [entityRef=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), name=getReplicationGroupAction, resourceType=replicationGroupResourceClass(com.vmware.vcloud.entity.resourceClass:05be7172-96b2-4094-bd5f-a9c06a1ba282), method=GET, urlPattern=/api/vr/replications/(?<id>[-,a-z,A-Z,0-9]*)] with ACL rule AclRule [entityRef=serviceResourceManageRule(com.vmware.vcloud.entity.aclRule:277a1c91-36f8-4522-8b16-dfa3b8934636), action=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), principalRef=com.vmware.vcloud.entity.right:10037d64-3fbc-4695-8439-54a27f86f377, orgRef=null, resourceRef=null]
2015-07-21 12:31:22,210	5a5dbcbb-d67b-4fbc-91c1-b72113bf6841-HMS-58404-fa67,	extension has returned HTTP response, skip further processing and send the response back
		Authorizing request POST /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-4

Figure 44 The output produced by the tool while tracing Test Failover in VCD Secondary

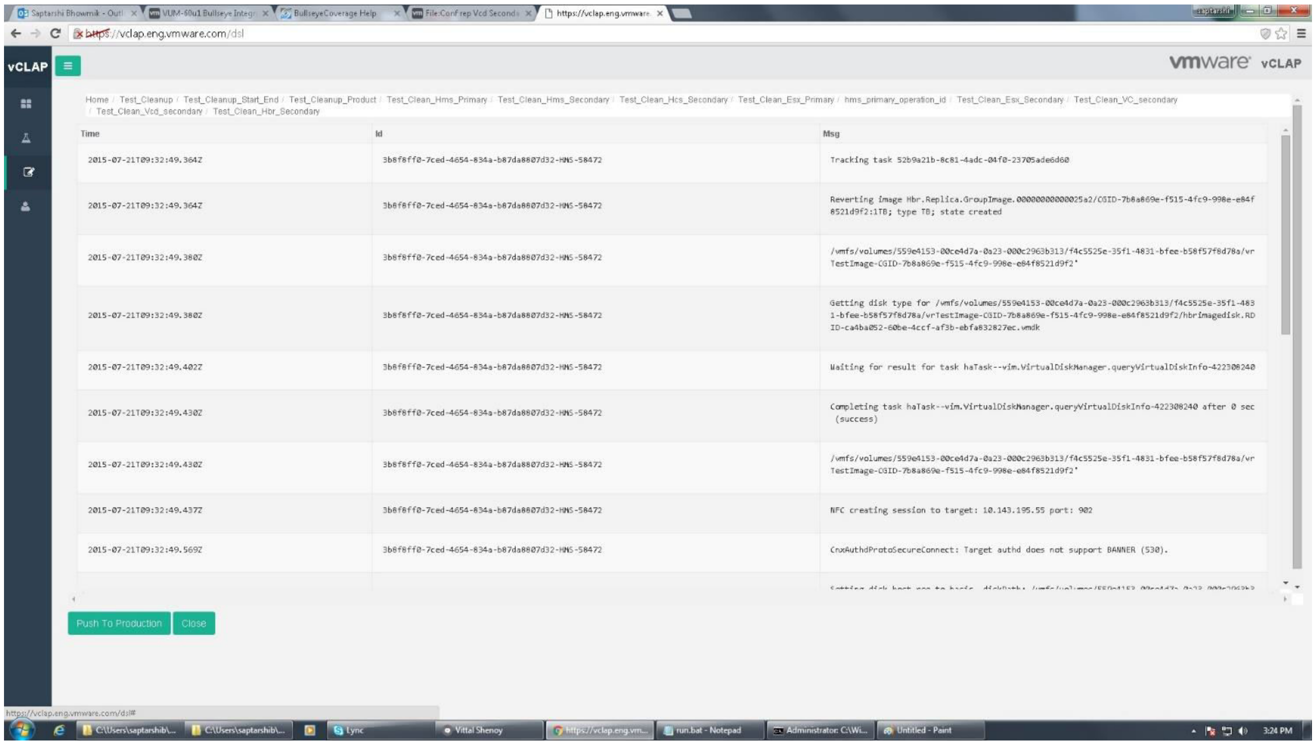
In Figure 37 the tool shows the traces left by hypervisor residing on the primary site when Test Failover operation is performed. It shows the log message, timestamp and the operation ID in a tabular format. In Figure 38 the tool shows the traces left by hypervisor residing on the secondary site when Test Failover operation is performed. It shows the log message, timestamp and the operation ID in a tabular format.
In Figure 39 the tool shows the traces left by vSphere Replication Server residing on the secondary site when Test Failover operation is performed. It shows the log message, timestamp and the operation ID in a tabular format.Figure 40 the result shows what record the HCS, leaves when Planned Failover operation is done. Figure 43 shows the record left by vCenter on the recovery site.Figure 41 shows the log records of Test Failover on HMS Primary. Figure 42 shows the log records of Test Failover on HMS Secondary. In Figure 44 the tool shows the traces left by the vCloud Director residing on the secondary site when Test Failover operation is performed. It shows the log message, timestamp and the operation ID in a tabular format.

5.3.4 TEST CLEANUP OPERATION


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Test_Cleanup / Test_Cleanup_Start_End / Test_Cleanup_Product / Test_Clean_Hms_Primary / Test_Clean_Hms_Secondary / Test_Clean_Hcs_Secondary / Test_Clean_Esx_Primary / hms_primary_operation_id / Test_Clean_Esx_Secondary
Push To Production
Close

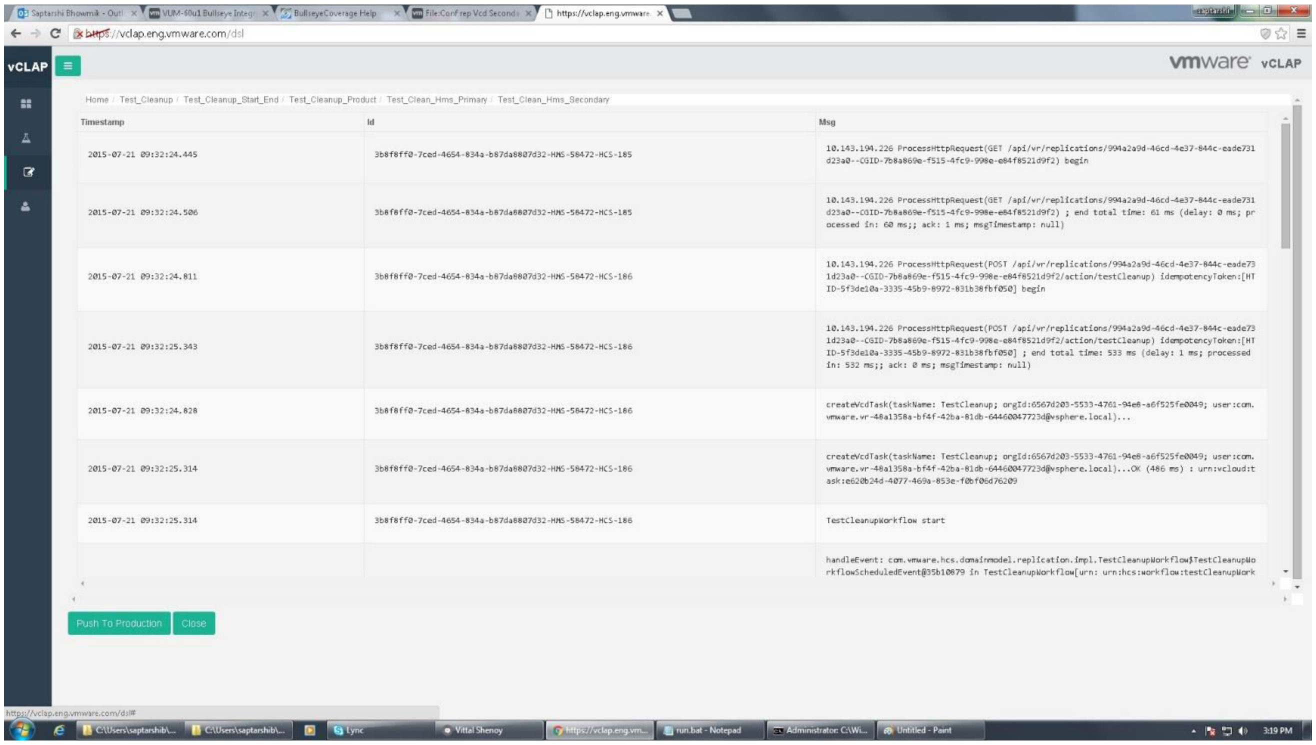
Msg	Test 2	Msg2
2015-07-21T09:32:48.005Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	AdapterServer verbose
2015-07-21T09:32:48.006Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	22-vim.VirtualMachine.unregister info
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	info
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose Destroy
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.013Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.014Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.014Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose
2015-07-21T09:32:48.014Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	verbose

Figure 45 The output produced by the tool while tracing Test Cleanup in ESXi Secondary


Home / Test_Cleanup / Test_Cleanup_Start_End / Test_Cleanup_Product / Test_Clean_Hms_Primary / Test_Clean_Hms_Secondary / Test_Clean_Hcs_Secondary / Test_Clean_Esx_Primary / hms_primary_operation_id / Test_Clean_Esx_Secondary / Test_Clean_VC_secondary / Test_Clean_Vcd_secondary / Test_Clean_Hbr_Secondary

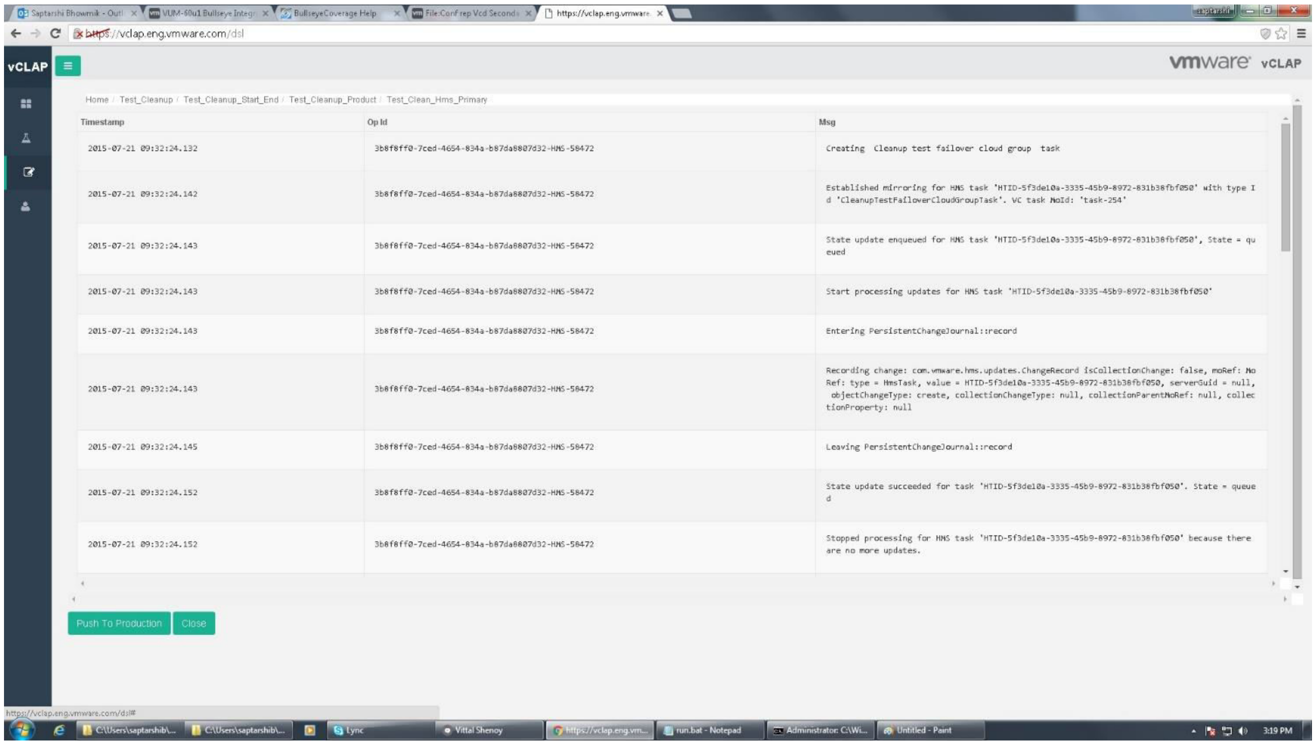
Time	Id	Msg
2015-07-21T09:32:49.364Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Tracking task 52b9a21b-8c81-4adc-04f0-23705ade6d60
2015-07-21T09:32:49.364Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Reverting image Hbr.Replica.GroupImage.00000000000025a2/CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2:1TB; type TB; state created
2015-07-21T09:32:49.380Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a/vrTestImage-CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2'
2015-07-21T09:32:49.380Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Getting disk type for /vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a/vrTestImage-CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2/hbrimagedisk.RDID-ca4ba052-60be-4ccf-af3b-ebfa832827ec.vmdk
2015-07-21T09:32:49.402Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Waiting for result for task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422308240
2015-07-21T09:32:49.430Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Completing task haTask--vim.VirtualDiskManager.queryVirtualDiskInfo-422308240 after 0 sec (success)
2015-07-21T09:32:49.430Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	/vmfs/volumes/559e4153-00ce4d7a-0a23-000c2963b313/f4c5525e-35f1-4831-bfee-b58f57f8d78a/vrTestImage-CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2'
2015-07-21T09:32:49.437Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	NFC creating session to target: 10.143.195.55 port: 902
2015-07-21T09:32:49.569Z	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	CnxAuthdProtoSecureConnect: Target authd does not support BANNER (530).

Figure 46 The output produced by the tool while tracing Test Cleanup in HBR Secondary


Home / Test_Cleanup / Test_Cleanup_Start_End / Test_Cleanup_Product / Test_Clean_Hms_Primary / Test_Clean_Hms_Secondary

Timestamp	Id	Msg
2015-07-21 09:32:24.445	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-185	10.143.194.226 ProcessHttpRequest(GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2) begin
2015-07-21 09:32:24.506	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-185	10.143.194.226 ProcessHttpRequest(GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2) ; end total time: 61 ms (delay: 0 ms; processed in: 60 ms;; ack: 1 ms; msgTimestamp: null)
2015-07-21 09:32:24.811	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-186	10.143.194.226 ProcessHttpRequest(POST /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2/action/testCleanup) idempotencyToken:[HTID-5f3de10a-3335-45b9-8972-831b38fbf050] begin
2015-07-21 09:32:25.343	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-186	10.143.194.226 ProcessHttpRequest(POST /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2/action/testCleanup) idempotencyToken:[HTID-5f3de10a-3335-45b9-8972-831b38fbf050] ; end total time: 533 ms (delay: 1 ms; processed in: 532 ms;; ack: 0 ms; msgTimestamp: null)
2015-07-21 09:32:24.828	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-186	createVcdTask(taskName: TestCleanup; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...
2015-07-21 09:32:25.314	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-186	createVcdTask(taskName: TestCleanup; orgId:6567d203-5533-4761-94e8-a6f525fe0049; user:com.vmware.vr-48a1358a-bf4f-42ba-81db-64460047723d@vsphere.local)...OK (486 ms) : urn:vcloud:task:e620b24d-4077-469a-853e-f0bf06d76209
2015-07-21 09:32:25.314	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472-HCS-186	TestCleanupWorkflow start
		handleEvent: com.vmware.hcs.domainmodel.replication.impl.TestCleanupWorkflow$TestCleanupWorkflowScheduledEvent@35b10879 in TestCleanupWorkflow[urn: urn:hcs:workflow:testCleanupWork

Figure 47 The output produced by the tool while tracing Test Cleanup in HMS Secondary


vCLAP
vmware vCLAP
Home / Test_Cleanup / Test_Cleanup_Start_End / Test_Cleanup_Product / Test_Clean_Hms_Primary
Push To Production
Close

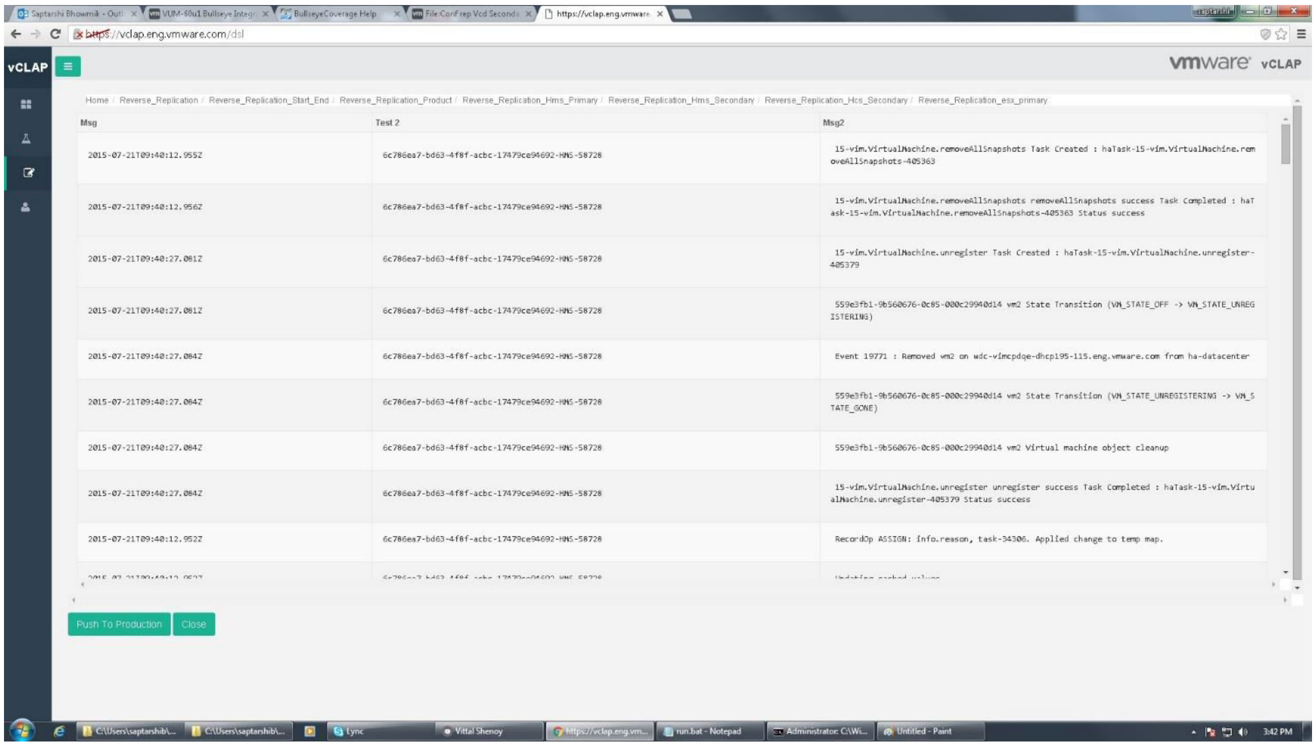
Timestamp	Op Id	Msg
2015-07-21 09:32:24.132	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Creating Cleanup test failover cloud group task
2015-07-21 09:32:24.142	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Established mirroring for HMS task 'HTID-5f3de10a-3335-45b9-8972-831b38fbf050' with type Id 'CleanupTestFailoverCloudGroupTask'. VC task MoId: 'task-254'
2015-07-21 09:32:24.143	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	State update enqueued for HMS task 'HTID-5f3de10a-3335-45b9-8972-831b38fbf050', State = queued
2015-07-21 09:32:24.143	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Start processing updates for HMS task 'HTID-5f3de10a-3335-45b9-8972-831b38fbf050'
2015-07-21 09:32:24.143	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Entering PersistentChangeJournal::record
2015-07-21 09:32:24.143	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-5f3de10a-3335-45b9-8972-831b38fbf050, serverGuid = null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 09:32:24.145	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Leaving PersistentChangeJournal::record
2015-07-21 09:32:24.152	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	State update succeeded for task 'HTID-5f3de10a-3335-45b9-8972-831b38fbf050'. State = queued
2015-07-21 09:32:24.152	3b8f8ff0-7ced-4654-834a-b87da8807d32-HMS-58472	Stopped processing for HMS task 'HTID-5f3de10a-3335-45b9-8972-831b38fbf050' because there are no more updates.

Figure 48 The output produced by the tool while tracing Test Cleanup in HMS Primary

Test Cleanup Results are shown above. The products which undergo a cleanup operation to remove the Virtual machines from the recovery site, leaves the messages in the log files, the tool extract those messages and displays them in a tabular format.

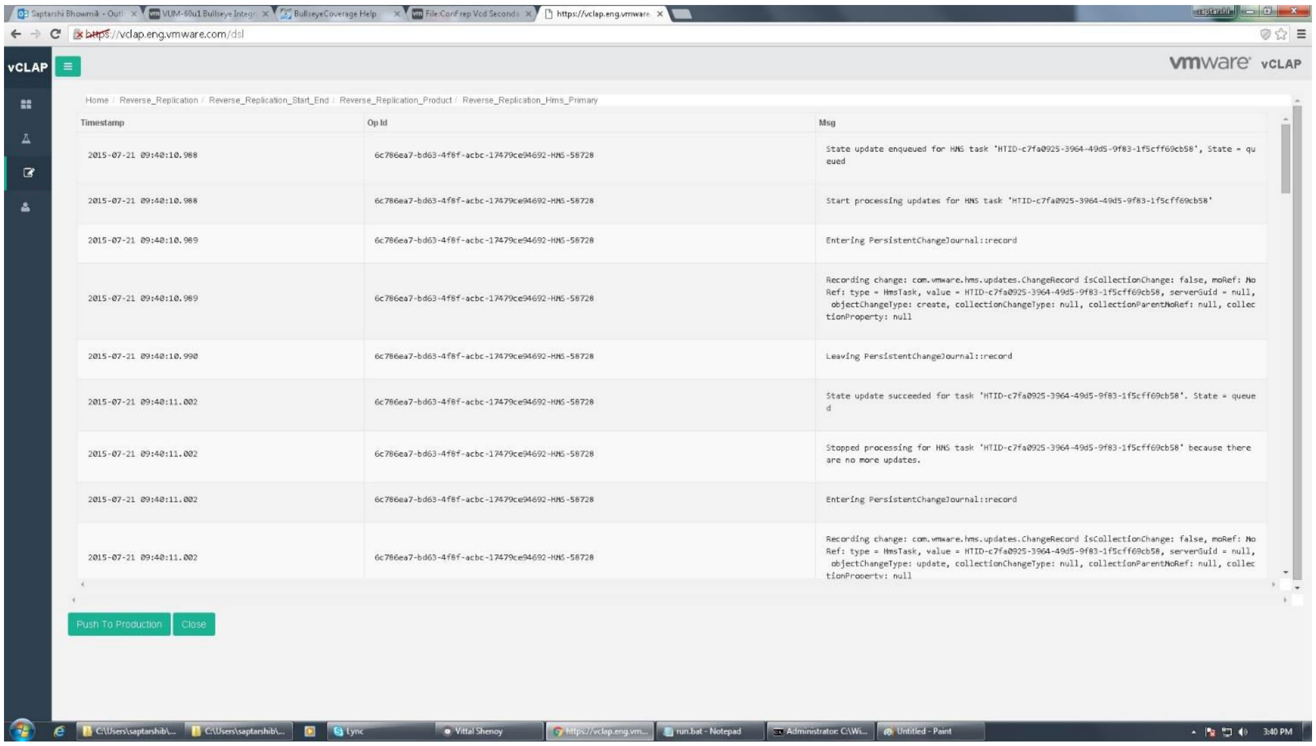
The result for four products that is ESXi Secondary, HBR Secondary, HMS Primary and HMS Secondary are shown above. In Figure 45 the tools displays the result of the Test Cleanup operation, it searches the logs in the databases and finds out the traces left by the hypervisor on the secondary site. In figure 46, the tools displays the result of the Test Cleanup operation, it searches the logs in the databases and finds out the traces left by the vSphere Replication Server on the secondary site. It clearly displays the operation ID and the log messages. In Figure 47 and Figure 48 tool traces and shows the results of Test Cleanup operation done by vSphere Replication Management Server on both protected site and recovery site. In the next two parts results produced when the tool is used to trace Reverse Replication and the result produced when the tool is used to trace Sync operation is displayed properly.

5.3.5 REVERSE REPLICATION OPERATION


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Hms_Primary / Reverse_Replication_Hms_Secondary / Reverse_Replication_Hcs_Secondary / Reverse_Replication_esx_primary
Push To Production
Close

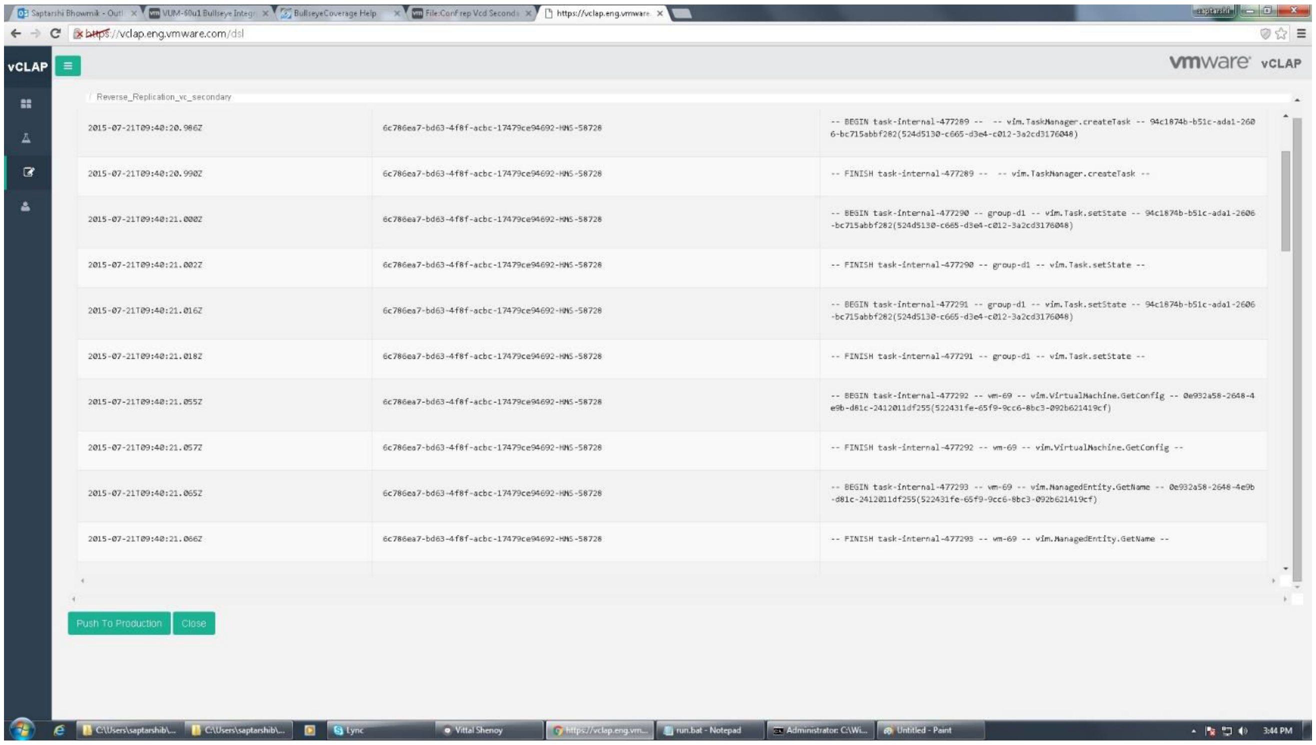
Msg	Test 2	Msg2
2015-07-21T09:40:12.955Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	15-vim.VirtualMachine.removeAllSnapshots Task Created : haTask-15-vim.VirtualMachine.removeAllSnapshots-405363
2015-07-21T09:40:12.956Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	15-vim.VirtualMachine.removeAllSnapshots removeAllSnapshots success Task Completed : haTask-15-vim.VirtualMachine.removeAllSnapshots-405363 Status success
2015-07-21T09:40:27.081Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	15-vim.VirtualMachine.unregister Task Created : haTask-15-vim.VirtualMachine.unregister-405379
2015-07-21T09:40:27.081Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	559e3fb1-9b560676-0c85-000c29940d14 vm2 State Transition (VM_STATE_OFF -> VM_STATE_UNREGISTERING)
2015-07-21T09:40:27.084Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Event 19771 : Removed vm2 on wdc-vimcpdqe-dhcp195-115.eng.vmware.com from ha-datacenter
2015-07-21T09:40:27.084Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	559e3fb1-9b560676-0c85-000c29940d14 vm2 State Transition (VM_STATE_UNREGISTERING -> VM_STATE_GONE)
2015-07-21T09:40:27.084Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	559e3fb1-9b560676-0c85-000c29940d14 vm2 Virtual machine object cleanup
2015-07-21T09:40:27.084Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	15-vim.VirtualMachine.unregister unregister success Task Completed : haTask-15-vim.VirtualMachine.unregister-405379 Status success
2015-07-21T09:40:12.952Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	RecordOp ASSIGN: info.reason, task-34306. Applied change to temp map.

Figure 49 The output produced by the tool while tracing Reverse Replication in ESXi Primary


https://vclap.eng.vmware.com/dsl
vCLAP
vmware vCLAP
Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Hms_Primary
Push To Production
Close

Timestamp	Op Id	Msg
2015-07-21 09:40:10.988	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	State update enqueued for HMS task 'HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58', State = queued
2015-07-21 09:40:10.988	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Start processing updates for HMS task 'HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58'
2015-07-21 09:40:10.989	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Entering PersistentChangeJournal::record
2015-07-21 09:40:10.989	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58, serverGuid = null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-07-21 09:40:10.990	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Leaving PersistentChangeJournal::record
2015-07-21 09:40:11.002	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	State update succeeded for task 'HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58'. State = queued
2015-07-21 09:40:11.002	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Stopped processing for HMS task 'HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58' because there are no more updates.
2015-07-21 09:40:11.002	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Entering PersistentChangeJournal::record
2015-07-21 09:40:11.002	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type = HmsTask, value = HTID-c7fa0925-3964-49d5-9f83-1f5cff69cb58, serverGuid = null, objectChangeType: update, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null

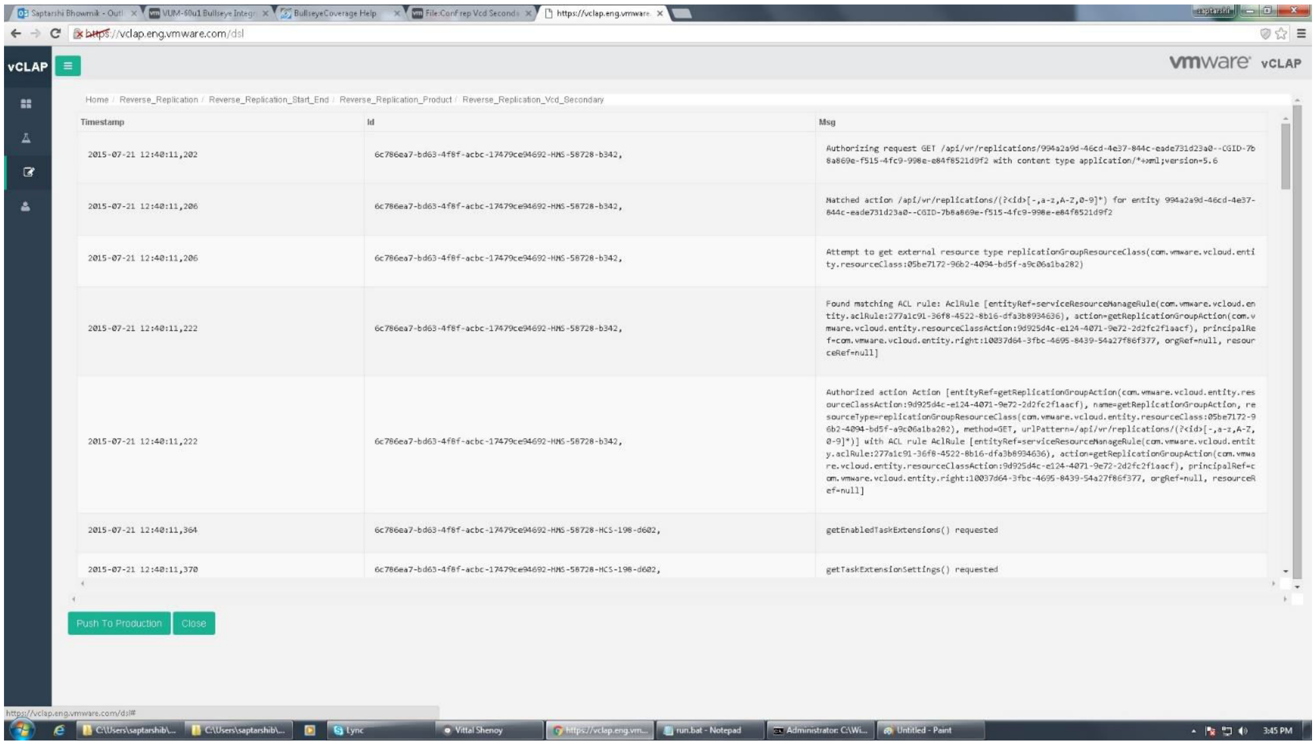
Figure 50 The output produced by the tool while tracing Reverse Replication in HMS Primary



Reverse_Replication_vc_secondary

2015-07-21T09:40:20.986Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- BEGIN task-internal-477289 -- -- vim.TaskManager.createTask -- 94c1874b-b51c-ada1-2606-bc715abbf282(524d5130-c665-d3e4-c012-3a2cd3176048)
2015-07-21T09:40:20.990Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- FINISH task-internal-477289 -- -- vim.TaskManager.createTask --
2015-07-21T09:40:21.000Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- BEGIN task-internal-477290 -- group-d1 -- vim.Task.setState -- 94c1874b-b51c-ada1-2606-bc715abbf282(524d5130-c665-d3e4-c012-3a2cd3176048)
2015-07-21T09:40:21.002Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- FINISH task-internal-477290 -- group-d1 -- vim.Task.setState --
2015-07-21T09:40:21.016Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- BEGIN task-internal-477291 -- group-d1 -- vim.Task.setState -- 94c1874b-b51c-ada1-2606-bc715abbf282(524d5130-c665-d3e4-c012-3a2cd3176048)
2015-07-21T09:40:21.018Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- FINISH task-internal-477291 -- group-d1 -- vim.Task.setState --
2015-07-21T09:40:21.055Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- BEGIN task-internal-477292 -- vm-69 -- vim.VirtualMachine.GetConfig -- 0e932a58-2648-4e9b-d81c-2412011df255(522431fe-65f9-9cc6-8bc3-092b621419cf)
2015-07-21T09:40:21.057Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- FINISH task-internal-477292 -- vm-69 -- vim.VirtualMachine.GetConfig --
2015-07-21T09:40:21.065Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- BEGIN task-internal-477293 -- vm-69 -- vim.ManagedEntity.GetName -- 0e932a58-2648-4e9b-d81c-2412011df255(522431fe-65f9-9cc6-8bc3-092b621419cf)
2015-07-21T09:40:21.066Z	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728	-- FINISH task-internal-477293 -- vm-69 -- vim.ManagedEntity.GetName --

Figure 51 The output produced by the tool while tracing Reverse Replication in VC Secondary



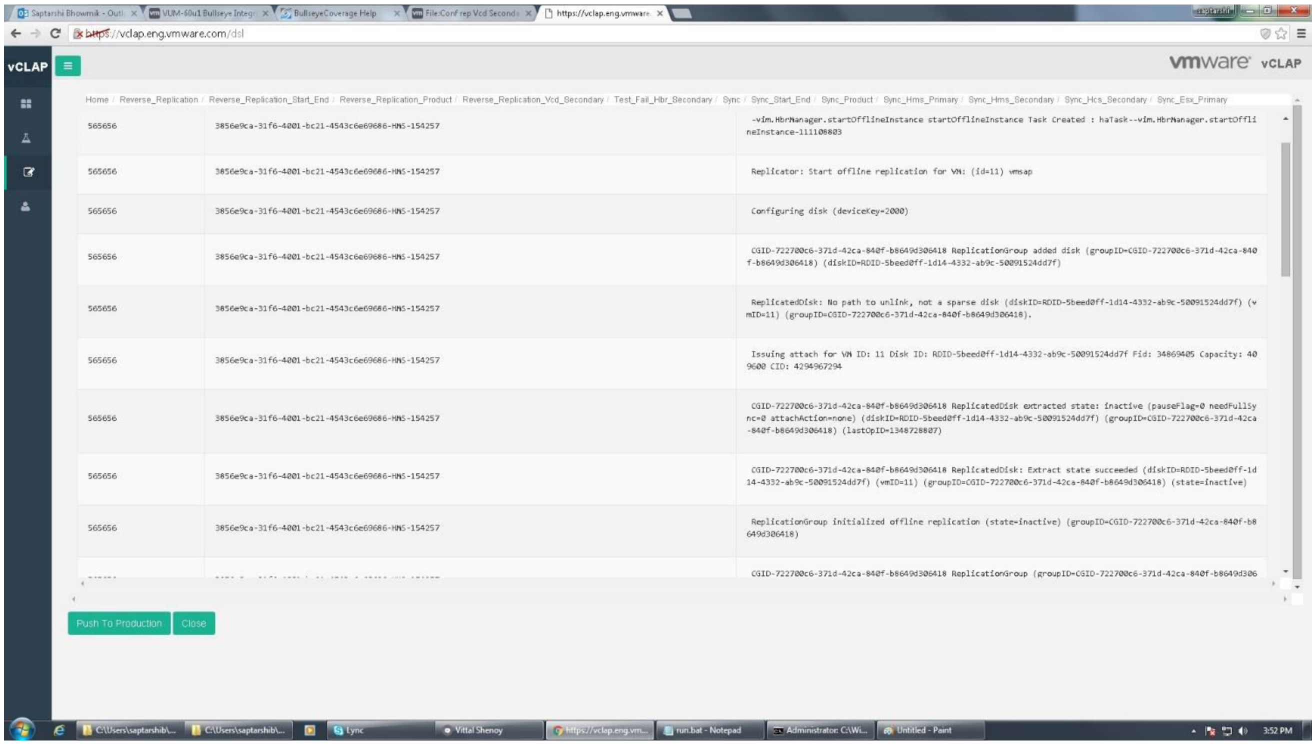
Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Vcd_Secondary

Timestamp	Id	Msg
2015-07-21 12:40:11,202	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-b342,	Authorizing request GET /api/vr/replications/994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2 with content type application/*+xml;version=5.6
2015-07-21 12:40:11,206	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-b342,	Matched action /api/vr/replications/(?<id>[-,a-z,A-Z,0-9]*) for entity 994a2a9d-46cd-4e37-844c-eade731d23a0--CGID-7b8a869e-f515-4fc9-998e-e84f8521d9f2
2015-07-21 12:40:11,206	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-b342,	Attempt to get external resource type replicationGroupResourceClass(com.vmware.vcloud.entity.resourceClass:05be7172-96b2-4094-bd5f-a9c06a1ba282)
2015-07-21 12:40:11,222	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-b342,	Found matching ACL rule: AclRule [entityRef=serviceResourceManageRule(com.vmware.vcloud.entity.aclRule:277a1c91-36f8-4522-8b16-dfa3b8934636), action=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), principalRef=com.vmware.vcloud.entity.right:10037d64-3fbc-4695-8439-54a27f86f377, orgRef=null, resourceRef=null]
2015-07-21 12:40:11,222	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-b342,	Authorized action Action [entityRef=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), name=getReplicationGroupAction, resourceType=replicationGroupResourceClass(com.vmware.vcloud.entity.resourceClass:05be7172-96b2-4094-bd5f-a9c06a1ba282), method=GET, urlPattern=/api/vr/replications/(?<id>[-,a-z,A-Z,0-9]*)] with ACL rule AclRule [entityRef=serviceResourceManageRule(com.vmware.vcloud.entity.aclRule:277a1c91-36f8-4522-8b16-dfa3b8934636), action=getReplicationGroupAction(com.vmware.vcloud.entity.resourceClassAction:9d925d4c-e124-4071-9e72-2d2fc2f1aacf), principalRef=com.vmware.vcloud.entity.right:10037d64-3fbc-4695-8439-54a27f86f377, orgRef=null, resourceRef=null]
2015-07-21 12:40:11,364	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-HCS-198-d602,	getEnabledTaskExtensions() requested
2015-07-21 12:40:11,370	6c786ea7-bd63-4f8f-acbc-17479ce94692-HMS-58728-HCS-198-d602,	getTaskExtensionSettings() requested

Figure 52 The output produced by the tool while tracing Reverse Replication in VCD Secondary

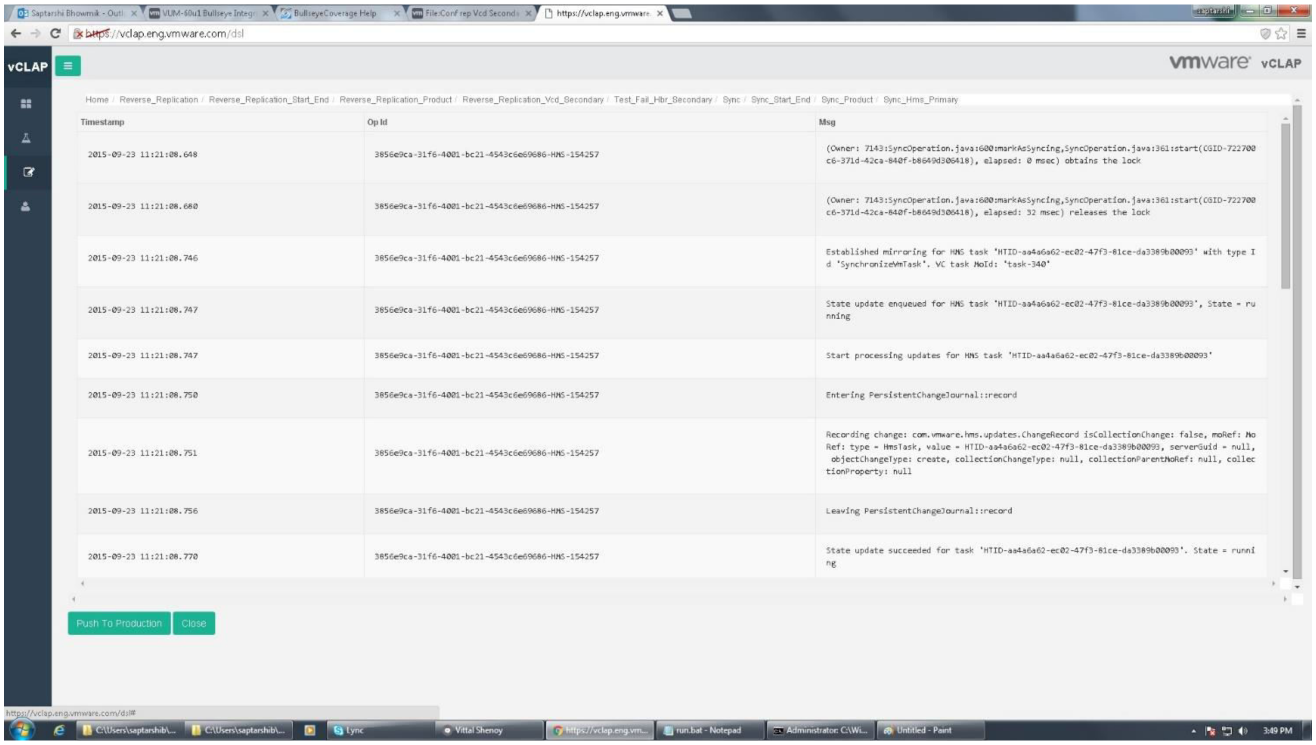
The Reverse Replication operation is done to replicate the virtual machines from recovery site that is the secondary site to the primary site which is the protected site. In Figure 49 the tool shows the traces left by the hypervisor on the primary site when Reverse Replication operation is performed. It shows the log message, timestamp and the operation ID in a tabular format. In Figure 50 the tool shows the traces left by the vSphere Replication Management Server on the primary site when Reverse Replication operation is performed. In Figure 51 the tool shows the traces left by the vCenter on the secondary site when Reverse Replication operation is performed. It shows the log message, timestamp and the operation ID in a tabular format and In Figure 49 the tool shows the traces left by the vCloud Director on the secondary site when Reverse Replication operation is performed and again it shows the log message, timestamp and the operation ID in a tabular format.

5.3.6 SYNC OPERATION


vCLAP
https://vclap.eng.vmware.com/dsl
vmware vCLAP
Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Vcd_Secondary / Test_Fail_Hbr_Secondary / Sync / Sync_Start_End / Sync_Product / Sync_Hms_Primary / Sync_Hms_Secondary / Sync_Hcs_Secondary / Sync_Esx_Primary
Push To Production
Close

```
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   -vim.HbrManager.startOfflineInstance startOfflineInstance Task Created : haTask--vim.HbrManager.startOffli
                                                          neInstance-111108803
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   Replicator: Start offline replication for VM: (id=11) vmsap
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   Configuring disk (deviceKey=2000)
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   CGID-722700c6-371d-42ca-840f-b8649d306418 ReplicationGroup added disk (groupID=CGID-722700c6-371d-42ca-840
                                                          f-b8649d306418) (diskID=RDID-5beed0ff-1d14-4332-ab9c-50091524dd7f)
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   ReplicatedDisk: No path to unlink, not a sparse disk (diskID=RDID-5beed0ff-1d14-4332-ab9c-50091524dd7f) (v
                                                          mID=11) (groupID=CGID-722700c6-371d-42ca-840f-b8649d306418).
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   Issuing attach for VM ID: 11 Disk ID: RDID-5beed0ff-1d14-4332-ab9c-50091524dd7f Fid: 34869405 Capacity: 40
                                                          9600 CID: 4294967294
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   CGID-722700c6-371d-42ca-840f-b8649d306418 ReplicatedDisk extracted state: inactive (pauseFlag=0 needFullSy
                                                          nc=0 attachAction=none) (diskID=RDID-5beed0ff-1d14-4332-ab9c-50091524dd7f) (groupID=CGID-722700c6-371d-42ca
                                                          -840f-b8649d306418) (lastOpID=1348728807)
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   CGID-722700c6-371d-42ca-840f-b8649d306418 ReplicatedDisk: Extract state succeeded (diskID=RDID-5beed0ff-1d
                                                          14-4332-ab9c-50091524dd7f) (vmID=11) (groupID=CGID-722700c6-371d-42ca-840f-b8649d306418) (state=inactive)
565656   3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257   ReplicationGroup initialized offline replication (state=inactive) (groupID=CGID-722700c6-371d-42ca-840f-b8
                                                          649d306418)
                                                          CGID-722700c6-371d-42ca-840f-b8649d306418 ReplicationGroup (groupID=CGID-722700c6-371d-42ca-840f-b8649d306
```

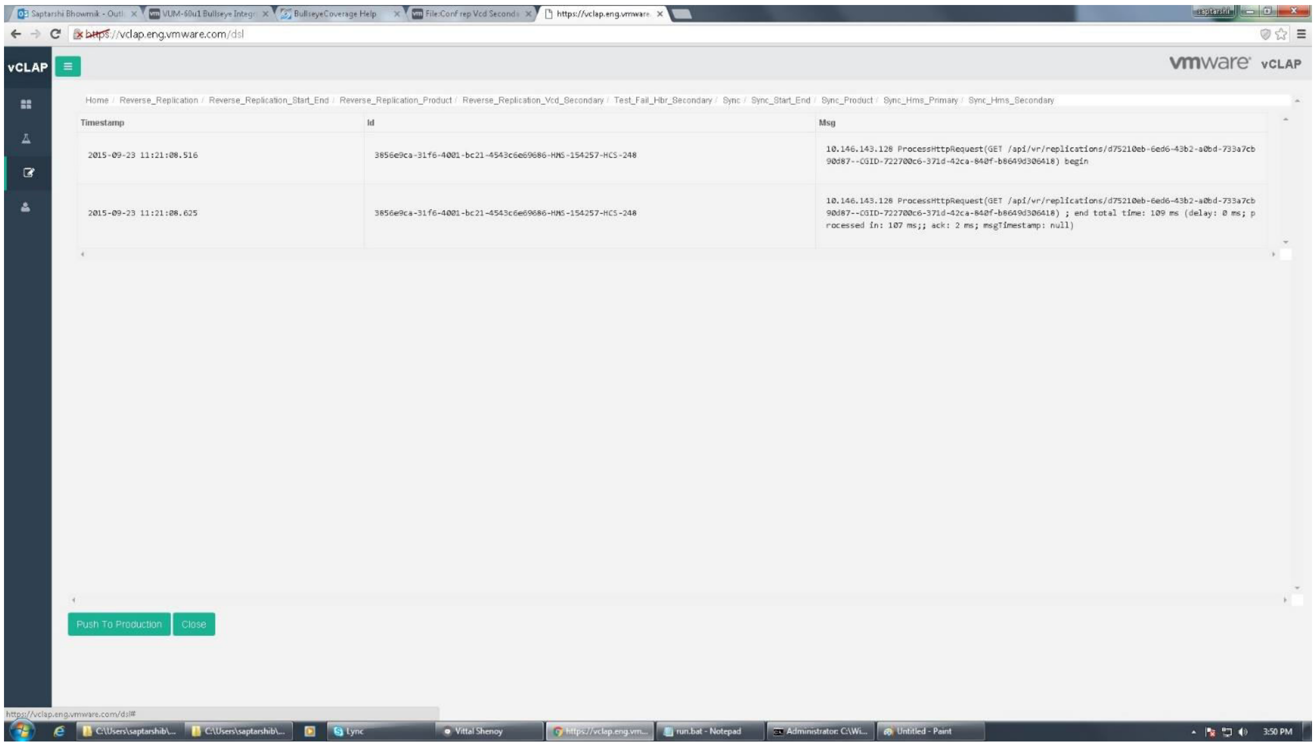
Figure 53 The output produced by the tool while tracing Sync in ESXi Secondary



Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Vcd_Secondary / Test_Fail_Hbr_Secondary / Sync / Sync_Start_End / Sync_Product / Sync_Hms_Primary

Timestamp	Op Id	Msg
2015-09-23 11:21:08.648	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	(Owner: 7143:SyncOperation.java:600:markAsSyncing,SyncOperation.java:361:start(CGID-722700c6-371d-42ca-840f-b8649d306418), elapsed: 0 msec) obtains the lock
2015-09-23 11:21:08.680	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	(Owner: 7143:SyncOperation.java:600:markAsSyncing,SyncOperation.java:361:start(CGID-722700c6-371d-42ca-840f-b8649d306418), elapsed: 32 msec) releases the lock
2015-09-23 11:21:08.746	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	Established mirroring for HMS task 'HTID-aa4a6a62-ec02-47f3-81ce-da3389b00093' with type Id 'SynchronizeVmTask'. VC task MoId: 'task-340'
2015-09-23 11:21:08.747	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	State update enqueued for HMS task 'HTID-aa4a6a62-ec02-47f3-81ce-da3389b00093', State - running
2015-09-23 11:21:08.747	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	Start processing updates for HMS task 'HTID-aa4a6a62-ec02-47f3-81ce-da3389b00093'
2015-09-23 11:21:08.750	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	Entering PersistentChangeJournal::record
2015-09-23 11:21:08.751	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	Recording change: com.vmware.hms.updates.ChangeRecord isCollectionChange: false, moRef: MoRef: type - HmsTask, value - HTID-aa4a6a62-ec02-47f3-81ce-da3389b00093, serverGuid - null, objectChangeType: create, collectionChangeType: null, collectionParentMoRef: null, collectionProperty: null
2015-09-23 11:21:08.756	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	Leaving PersistentChangeJournal::record
2015-09-23 11:21:08.770	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257	State update succeeded for task 'HTID-aa4a6a62-ec02-47f3-81ce-da3389b00093'. State = running

Figure 54 The output produced by the tool while tracing Sync in HMS Primary



Home / Reverse_Replication / Reverse_Replication_Start_End / Reverse_Replication_Product / Reverse_Replication_Vcd_Secondary / Test_Fail_Hbr_Secondary / Sync / Sync_Start_End / Sync_Product / Sync_Hms_Primary / Sync_Hms_Secondary

Timestamp	Id	Msg
2015-09-23 11:21:08.516	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257-HCS-248	10.146.143.128 ProcessHttpRequest(GET /api/vr/replications/d75210eb-6ed6-43b2-a0bd-733a7cb90d87--CGID-722700c6-371d-42ca-840f-b8649d306418) begin
2015-09-23 11:21:08.625	3856e9ca-31f6-4001-bc21-4543c6e69686-HMS-154257-HCS-248	10.146.143.128 ProcessHttpRequest(GET /api/vr/replications/d75210eb-6ed6-43b2-a0bd-733a7cb90d87--CGID-722700c6-371d-42ca-840f-b8649d306418) ; end total time: 109 ms (delay: 0 ms; processed in: 107 ms;; ack: 2 ms; msgTimestamp: null)

Figure 55 The output produced by the tool while tracing Sync in HMS Secondary

In Figure 53, the tools displays the result of the sync operation, it searches the logs in the databases and finds out the traces left by the hypervisor on the secondary site, the hypervisor on the secondary site when undergoes the sync operation leaves the messages in various log files, the tool categorically find out all those traces, combines it into one and displays them in a table. The table shown in the output has the messages and the operation ID associated with the product unique to that particular operation.

Figure 54, similarly displays the traces left by the vSphere Replication Management Server in the primary site when sync operation is performed. It shows the timestamp of the message, that is the exact time when the vSphere Replication Management Server is leaving the trace in the log files. It also displays the operation ID and the log message.

In Figure 55, the tool displays the traces left by the vSphere Replication Management Server in the primary site when sync operation is performed. Here again it shows the timestamp of the message, operation ID and the log message. It displays all of it in a tabular format.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1MAJOR CONTRIBUTIONS

In this paper, Cloud Computing basics have been defined. Firstly, the service classifications are discussed in details, which forms the basis of cloud computing. Then after that the deployments models are discussed. After this the cloud operations such as cloning, migration and other operations which are fairly part of cloud computing paradigm are talked about. In the next part some light is thrown on the disasters in cloud. In the final part of the chapter the organization of the thesis is mentioned.

The next chapter talks about some related works that are done previously. Since the work is mostly concentrated on log analysis and requires the knowledge of Domain Specific Languages, therefore majorly these two areas have been studied. The related works is classified into two sections 1.log analysis 2.Domain Specific Languages.

In the third chapter, the disaster in cloud is mainly focused on, firstly the concept of cloud replication is discussed, and how it is helping in actual world, why disaster recovery is needed. Since the base of the stack is built on VMware applications, so the applications such as Site Recovery Manager and vSphere Replication are discussed in details. The final part of the chapter talks about the life cycle of the cloud disaster recovery, how it happens and how the products are integrated together to help the disaster recovery happen. When all the products combine then the disaster recovery successfully occurs, also how different operations play their part is also talked about. The entire process is discussed here.

In, the fourth chapter the log analysis is discussed first after which various challenges faced in log collection is talked about, also the logging architecture is briefly discussed. In the next part of the chapter three applications are talked about which are part of the log analysis, firstly groovy language, secondly Domain Specific Language and thirdly a query language which is used in the database.

In the fifth chapter, the first part talk about the experimental setup, how the stack is built and how products are used to collect the logs files, the second part talks about the algorithms used to efficiently search the log file and trace the life cycle of an operation. The algorithms are implemented in Domain Specific Language. Also the flow of the tool is shown in a diagram. Finally in the last part the results are shown, the screenshots of how the tool works to trace the operation in the logs is shown.

6.2 FUTURE WORK

A number of issues and topics around cloud based logging haven't been able to discuss in this paper. However, in future these particular fields will be looked into: security visualization, timeline analysis. Also the log files which are studied and analyzed here are static logs, which are fetched beforehand. First log is generated then it is collected and after that the analysis is done. But the real log analysis should be dynamic. That is the analysis should be done simultaneously with log generation. This particular problem needs to be solved.

In today's world, every single action involves computers, in every one second thousand terabytes of data are generated, starting from security to business operations, everything involves generation of data. Also due to the advent of Internet of Things data generation has risen exponentially, so pretty much this data is stored in a cloud. So cloud computing is slowly becoming synonymous with the modern world. If more dependence on cloud takes place, then it will become necessary for the people to maintain the infrastructure, as a result the log files need to be analyzed for issues. However with the increase in size more and more time is required for analysis of the ever increasing logs. So better algorithms need to be generated which can significantly reduce the time complexity and space complexity. Also the patterns used here can be used in big data to do some crucial mapping and reduce.

BIBLIOGRAPHY

1. G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. V¨olkel. Design Guidelines for Domain Specific Languages. In OOPSLA Workshop on Domain-Specific Modeling (DSM), 2009.

2. Kosar, T., Oliveira, N., Mernik, M., Varanda Pereira, M. J., Črepinšek, M., da Cruz, D., Henriques, P. R.: Comparing General-Purpose and Domain-Specific Languages: An Empirical Study. Computer Science and Information Systems, Vol. 7, No. 2, 247-264. (2010)

3. Kelly, S., Tolvanen, J-P.: Domain-Specific Modeling: Enabling Full-Code Generation, John Wiley and Sons. (2008)

4. Lédeczi, Á., Bakay, Á., Maróti, M., Völgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G.: Composing Domain-Specific Design Environments, Computer, Vol. 34, No. 11, 44-51. (2001)

5. Van Deursen, Arie, Paul Klint, and Joost Visser. "Domain-Specific Languages: An Annotated Bibliography." *Sigplan Notices* 35.6 (2000): 26-36.

6. http://www.mcrinc.com/Documents/Newsletters/201207_CloudComputing_E-Guide.pdf

7. NIST, NIST Definitoon of cloud computing v15, NIST, Editor. 2009, National Institute of Standards and Technology: Gaithersburg, MD (2009).

8. Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In The 10th International Conference on Algorithms and Architectures for Parallel Processing, Busan, Korea, 2010.

9. Kimberly Keeton, Dirk Beyer, Ernesto Brau, Arif Merchant, Cipriano Santos, and Alex Zhang. On the road to recovery:restoring data after disasters. European Conference on Computer Systems, 40(4), 2006.

10. Michael Armbrust et al. “PIQL: Success-Tolerant Query Processing in the Cloud”. Proc. of VLDB. 2011, pp. 181–192.

11. M. Armbrust et al. PIQL: A Performance Insightful Query Language. In SIGMOD, 2010.

12. M. Armbrust, N. Lanham, S. Tu, A. Fox, M. J. Franklin, and D. A. Patterson, “The case for piql: a performance insightful query language,” in SoCC, 2010, pp. 131–136

13. Gharat, M. A. A., & Mhamunkar, M. D. E. Disaster Recovery in Cloud Computing. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 4(5) 2015.

14. M. Pokharel, S. Lee and J. S. Park, "Disaster Recovery for System Architecture Using Cloud Computing," Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on, Seoul, 2010, pp. 304-307. doi: 10.1109/SAINT.2010.23

15. R. Marty. Cloud application logging for forensics. In proceedings of the 2011 ACM Symposium on Applied Computing, pages 178–184. ACM, 2011.