THOMSON REUTERS

7MTT – NFS 7mode to CDOT Work Instructions

Authors: Ian Daniel

Contributors: David Ng, Sridhar Chevendra, Craig Goettig

Document Version: V1.5

Date: April 2016

Status: Draft

Contents

1 Introduction

1.1 Management Summary

This document details the process used to migrate an NFS share from 7mode to CDOT with 7MTT.

- Servers connected via NFS or SMB

- One Vserver per application
- Up to 128 Vservers per node
- One LIF per Vserver with NFS or SMB

- FAS80xx or FAS32xx cluster with 2 to 8 nodes
- Single node cluster for SnapVault backup systems

- DS2246 or DS4246 SAS attached disk shelves
- Multiple volumes per Vserver
- LIFs and Volumes on same node for direct path I/O

1.2 Change History

Ver	Date	Author	Key Changes
1	March 2016	Ian Daniel	Initial Version
1.1	April 2016	Ian Daniel	Added performance checks and script details.
1.2	April 2016	Ian Daniel	Added export policy script
1.3	April 2016	Ian Daniel	Modified LIF naming to adhere to standards. Updated QoS check and fixed formatting issues.
1.4	April 2016	Ian Daniel	Updated with changes requested by Support
1.5	April 2016	Ian Daniel	Updated with changes to dedicated QoS settings

1.3 Distribution List

Name	Role
Storage Support	
Storage Delivery	

1.4 References

No.	Title	Location
1	cDOT Naming Standards	Link
2	cDOT AV	Link
3	cDOT Vserver Creation	Link
4	cDOT Consolidated Build Standards	Link
5	cDOT Replication Deployment Guidelines	Link
6		

7		

1.5 Glossary

Term	Definition
cDOT	clustered Data ONTAP
Vserver	A logical storage virtual server, also known as a Storage Virtual Machine (SVM), which contains LIFs, Volumes, and configuration information such as access control details.
LIF	Logical Interface – a cDOT logical network interface with an IP address, assigned to a single Vserver.
7MTT	7mode Transition Tool – A tool used to migrate from 7mode to CDOT
WFA	OnCommand Workflow Automater – An automation framework application from NetApp, used for storage provisioning.

2 7-Mode to CDOT NFS Migration

2.1 High Level Migration Steps

Task	Owner
Create SVM and volume	Storage
Initialize snapmirror replication	Storage
Stop application during maintenance window	Application
Unmount source volumes during maintenance window	Platform
Final snapmirror sync/break during maintenance window	Storage
Update server mount info, e.g., '/etc/fstab'	Platform
Mount new volume during maintenance window	Platform
Start application	Application
Verify	Application/Platform

2.2 Pre-requisites

- **Ensure version of 7MTT is 2.3 or higher**
- **Ensure vfiler being migrated is not currently exceeding the 6K IOPS threshold unless it is moving to a dedicated controller.**
- **Ensure snapshot autodelete is disabled during the migration process for both source and destination.**
- Add the CDOT vserver name (not IP) to the source 7-Mode systems /etc/snapmirror.allow file
- Add the cDOT cluster ICL IP Addresses to the snapmirror.access option or the snapmirror.allow file on the source 7-Mode system prior to migration.
- Confirm the target aggregate has sufficient capacity
- Confirm the source volume is 64 bit
- Register the target vserver hostname in DNS
- Make note of service accounts on the source so they can be created in destination
- Confirm there is network connectivity between the source and target
- Make note of qtree quotas sizes so they can be applied in destination
- Make note of the source volume language
- Make note of the backup volume size

2.3 Vserver and LIF creation

2.3.1 Create Vserver (replace hyphen with underscore in vserver rootvolume name)

```
vserver create -vserver <vsname> -rootvolume <vsname>_root -aggregate <aggrname> -ns-switch file -
nm-switch file -rootvolume-security-style unix -language <language>

vserver show
```

2.3.2 Create LIF with default route and failover group

```
network interface create -vserver <vsname> -lif <vsname>-lif<lif#> -role data -data-protocol
<protocol> -home-node <node> -home-port <port> -address <ip> -netmask <netmask> -status-admin up -
firewall-policy mgmt -failover-group <group>

network routing-groups route create -vserver <vsname> -routing-group d<network>/<mask> -destination
0.0.0.0/0 -gateway <gateway>

vserver show
network interface show
network interface show -failover
network routing-groups route show –vserver <vsname>
```

2.4 DNS configuration

2.4.1 Setup DNS on a Vserver

```
vserver services dns create -vserver <vsname> -domains <domainname> -name-servers
<comma_separate_name_server_list>

vserver services dns show
```

2.4.2 Setup DNS on Cluster Admin Vserver

For ONTAP 8.2.x you need to make sure that whatever DNS domains are on your data vservers also get added to the admin vserver as DNS lookups can also be done via that vserver.

```
vserver services dns modify -vserver <admin_vsname> -domains <domainnames>

vserver services dns show
```

2.5 Create service account if required

```
security login role create -role <ROLE_NAME> -cmddirname "<COMMAND>" -access <ACCESS_TYPE> -vserver
<vserver>   #Repeat this command for each required cDOT command

security login create -username <USER_NAME> -application ontapi -authmethod <AUTHENTICATION_METHOD>
-role <ROLE_NAME> -vserver <vserver>
```

2.5.1 Oracle Account and Role

```
security login role create -role oracle -cmddirname "volume snapshot" -access all -
vserver
security login role create -role oracle -cmddirname "set" -access all -vserver
security login role create -role oracle -cmddirname "version" -access all -vserver
security login role create -role oracle -cmddirname "job show" -access readonly -vserver
security login role create -role oracle -cmddirname "df" -access readonly -vserver
security login role create -role oracle -cmddirname "snapmirror list-destinations" -
access readonly –vserver
security login create -username oracle -application ssh -authmethod publickey -role
oracle -vserver
security login publickey create -username oracle -vserver-publickey "<ssh-dss
pub_key_string>"
```

2.6 Enable NFS Protocol

2.6.1 Enable NFS

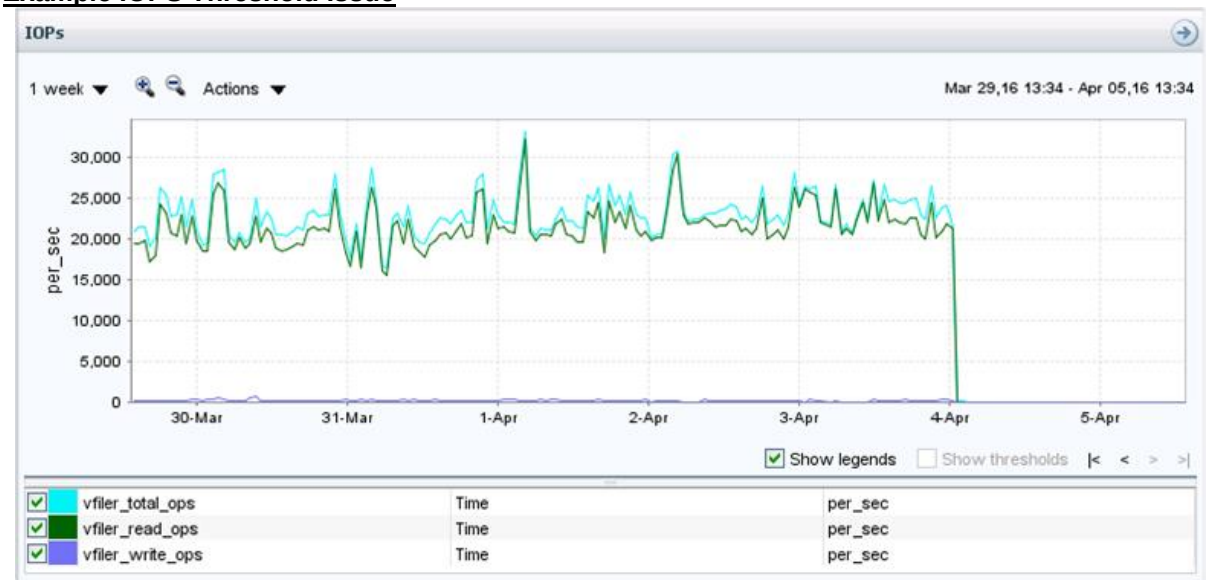
```
vserver nfs create -vserver <vsname> -access true -v3 enabled

vserver nfs show
```

2.7 Check Current IOPS In Use Prior To Migration – Shared Migrations ONLY

If the volume is seeing large periods of time over 6000 IOPs/sec please raise the flag that it needs to be looked at as 6000IOPs/sec will be the most they can use on cDOT due to QOS.So there is a high probability that the customer will feel the effect from this.

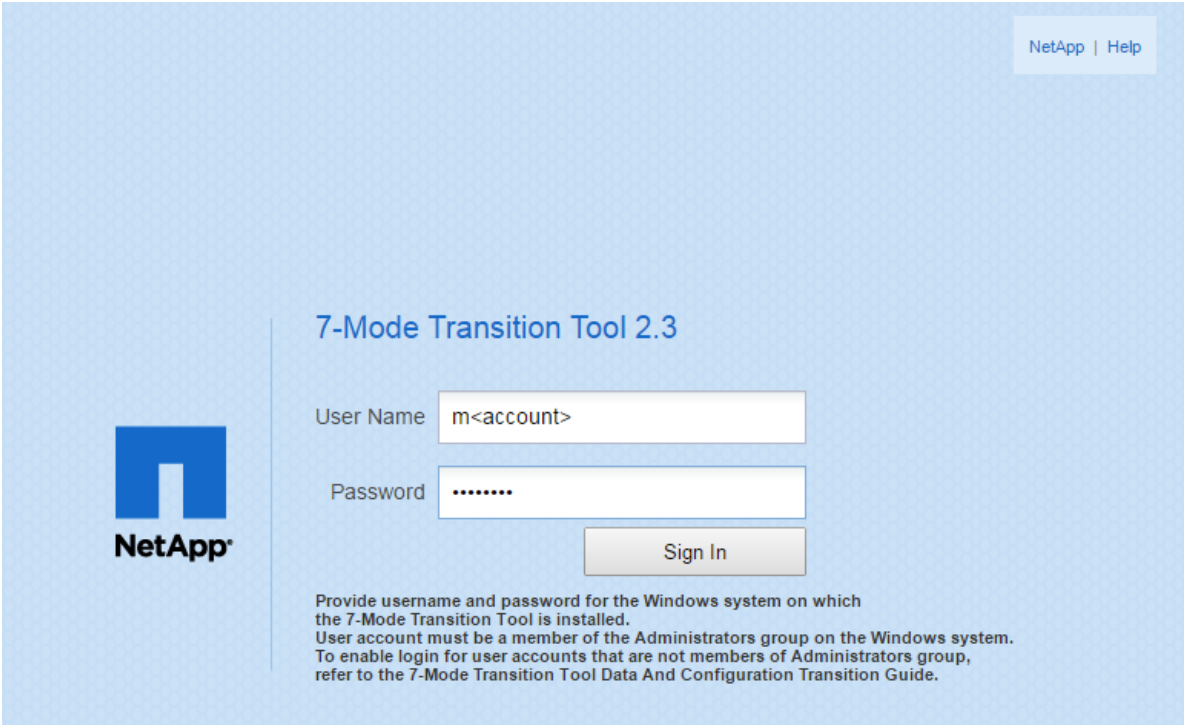
Example IOPS Threshold Issue



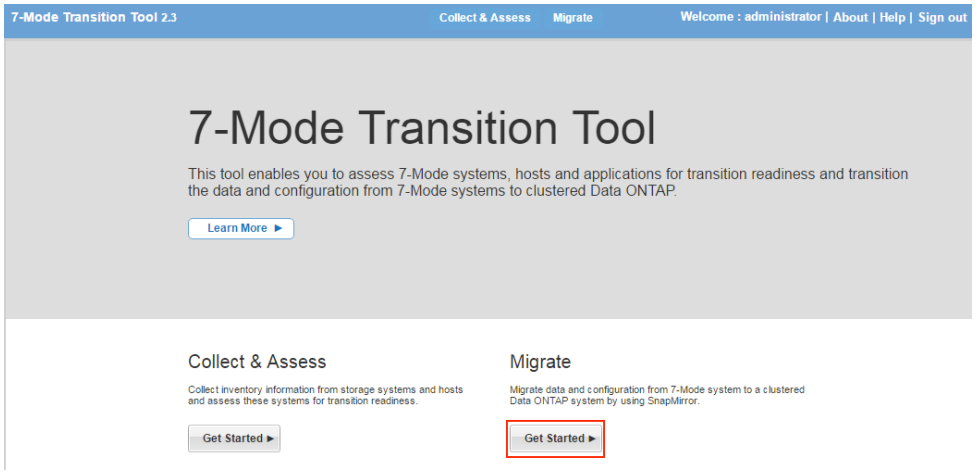
2.8 Start the Migration

2.8.1 Login to 7MTT

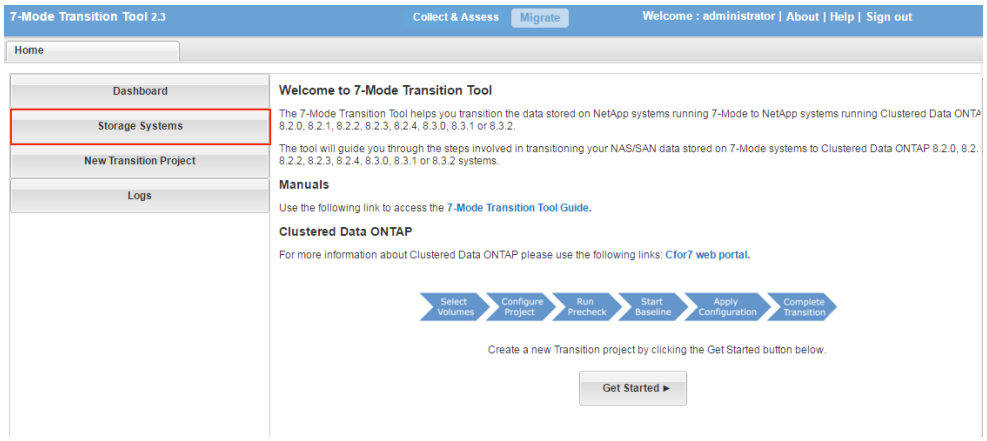
Connect the 7MTT (https://<7MTT_Server>:8443/transition) tool via web browser



Once logged in CLICK Get Started under Migrate’



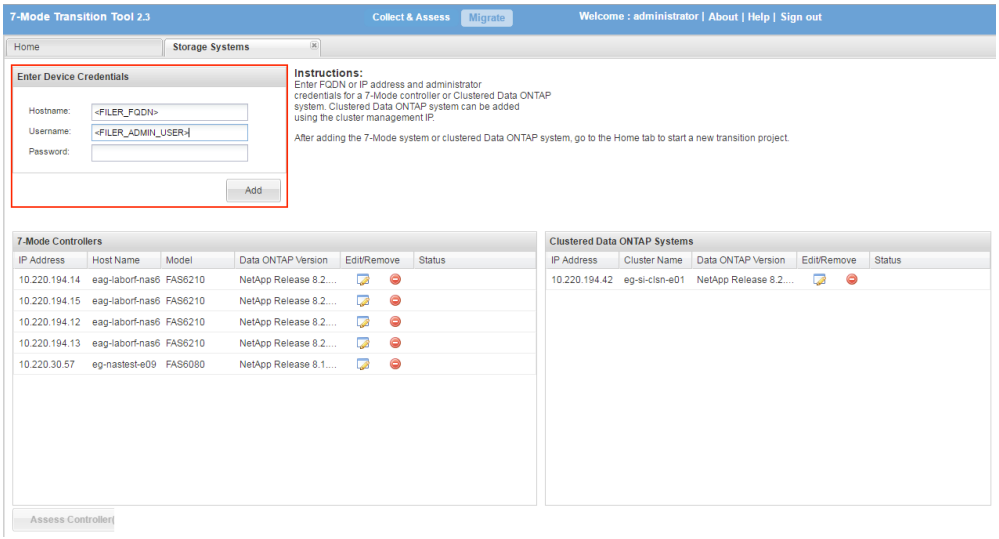
CLICK on the ‘Storage System’ button in the left pane:



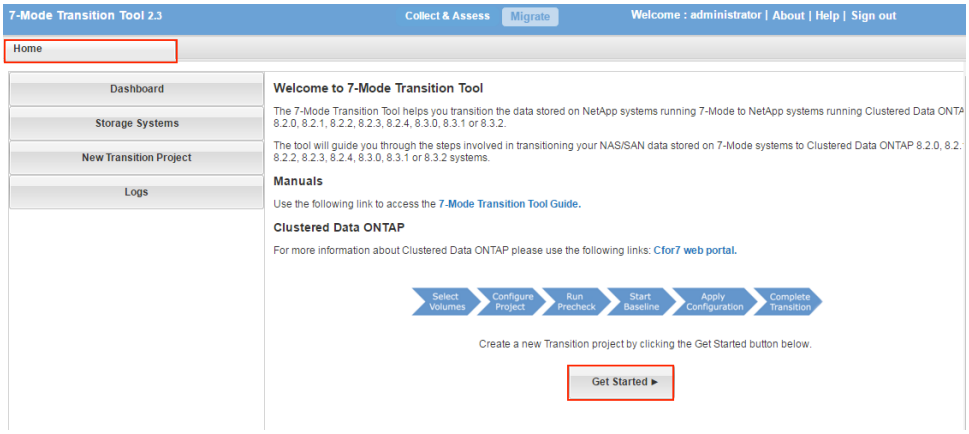
ADD storage systems to 7MTT (execute this step for the source and target storage systems):

ENTER the FQDN of the filer

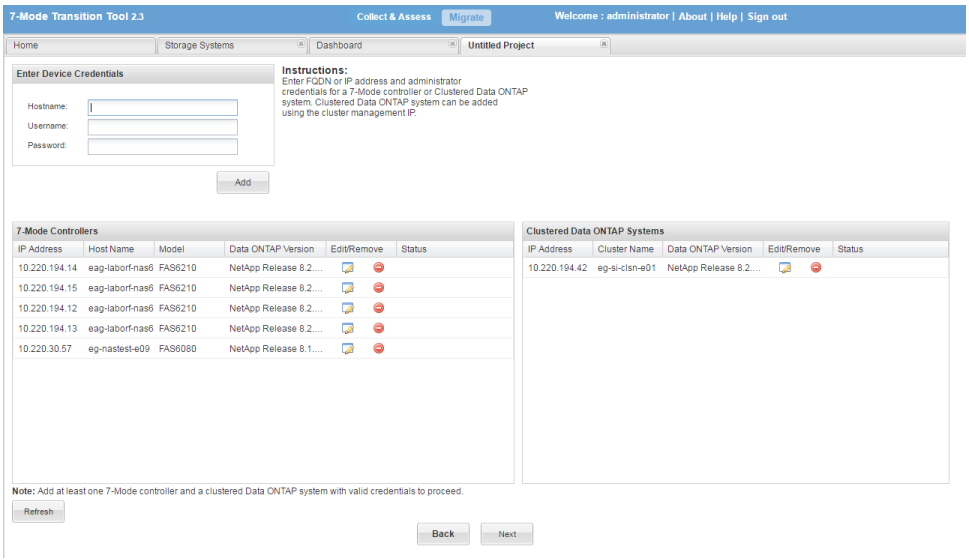
CLICK ADD



CLICK the ‘HOME’ tab -> CLICK ‘Get Started’



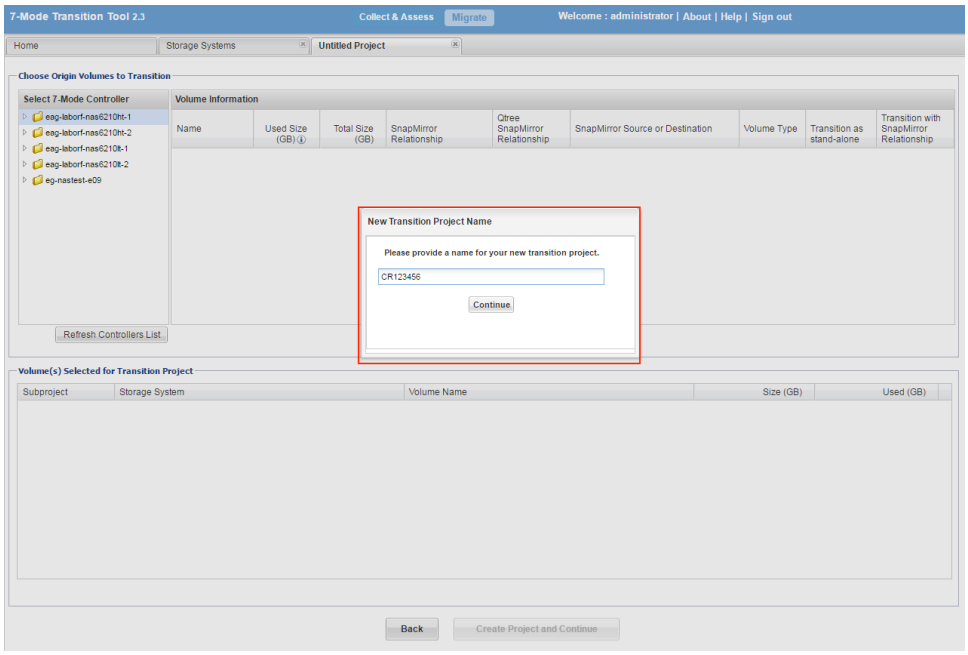
CONFIRM that the source and destination storage systems are listed



CLICK ‘Next’

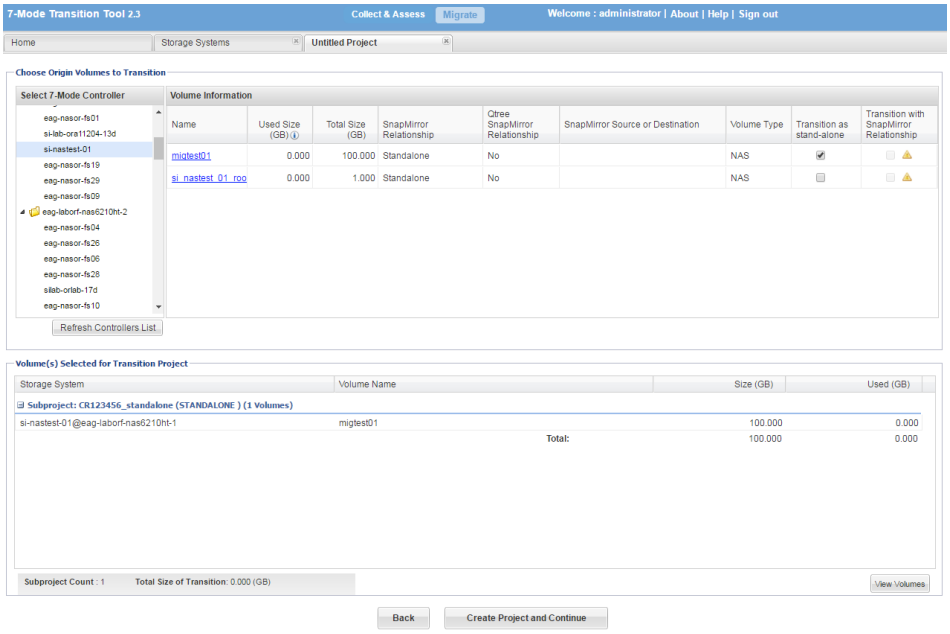
ENTER a name for the project

CLICK ‘Continue’



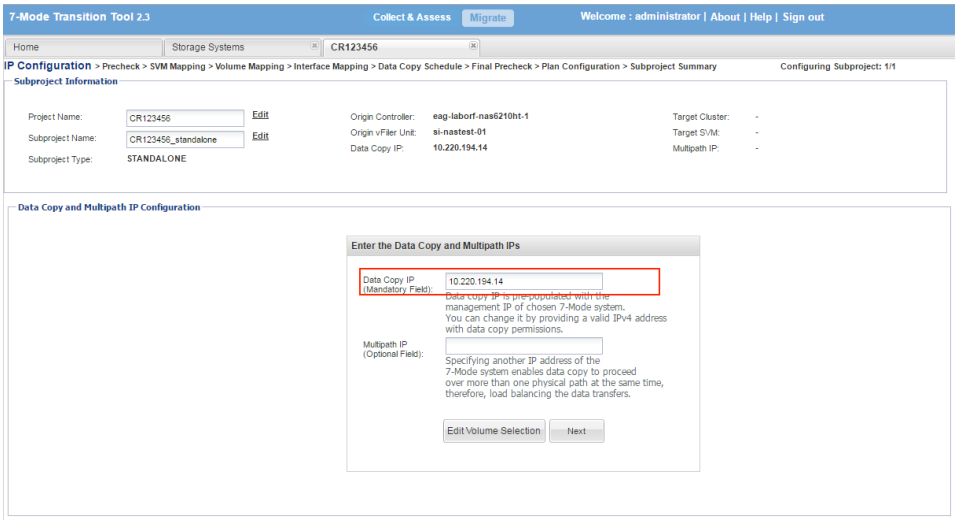
SELECT the source vfiler and volume(s):

CLICK ‘Create Project and Continue’



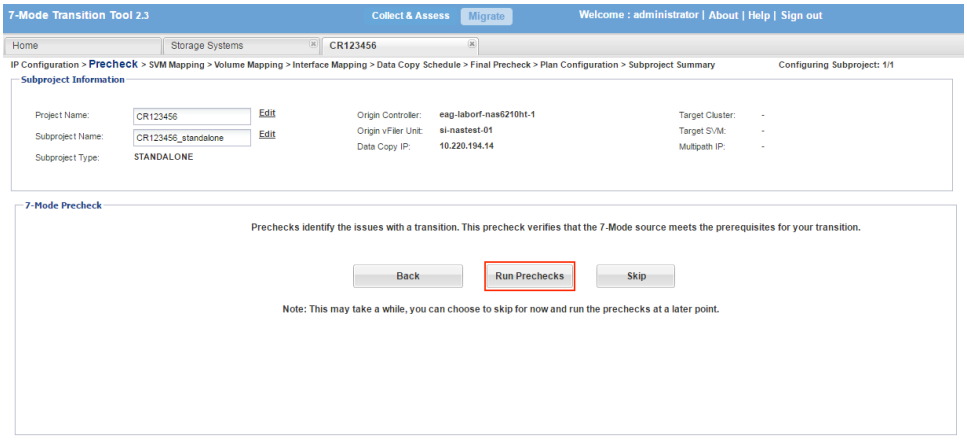
ENTER an IP to be used for replication traffic on the source filer

CLICK ‘Next’



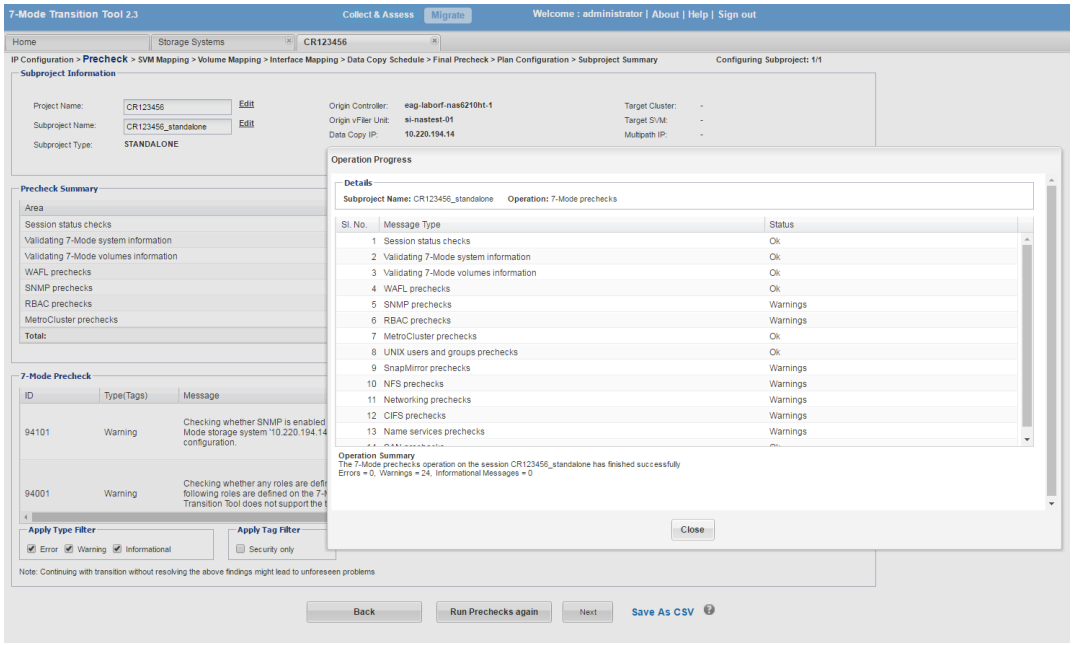
The next window will ask if you want to run pre-check.

CLICK ‘Run Prechecks’



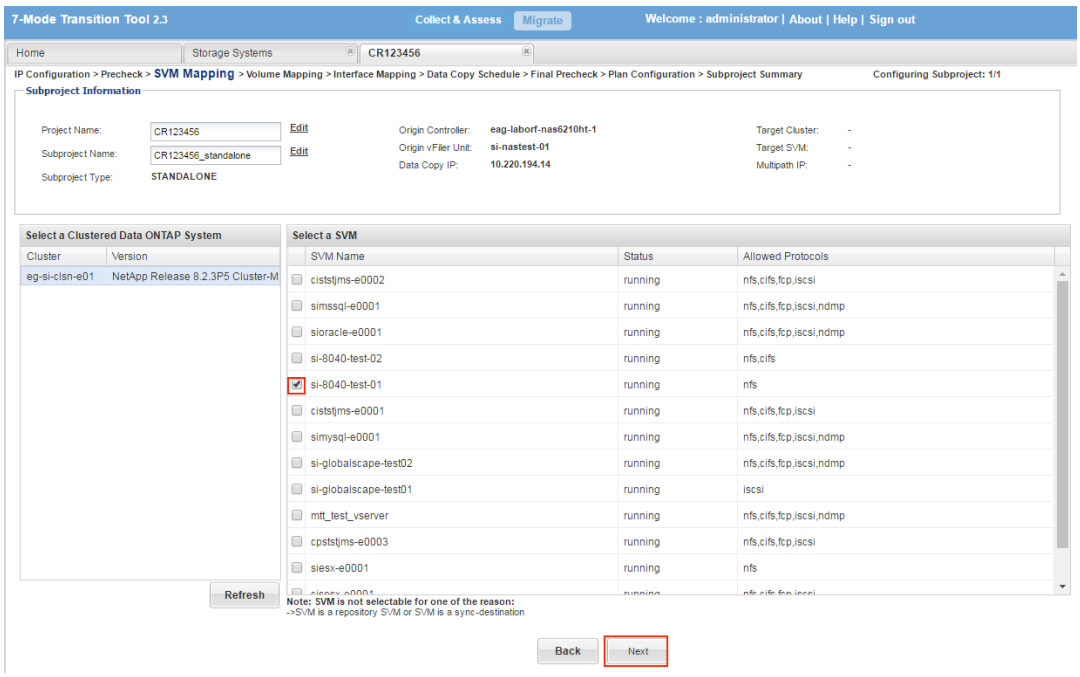
Review the Report. You can ignore the warnings listed below. Resolve errors if they appear.

CLICK ‘Close’ and then click Next’



SELECT a target vserver

CLICK ‘Next’



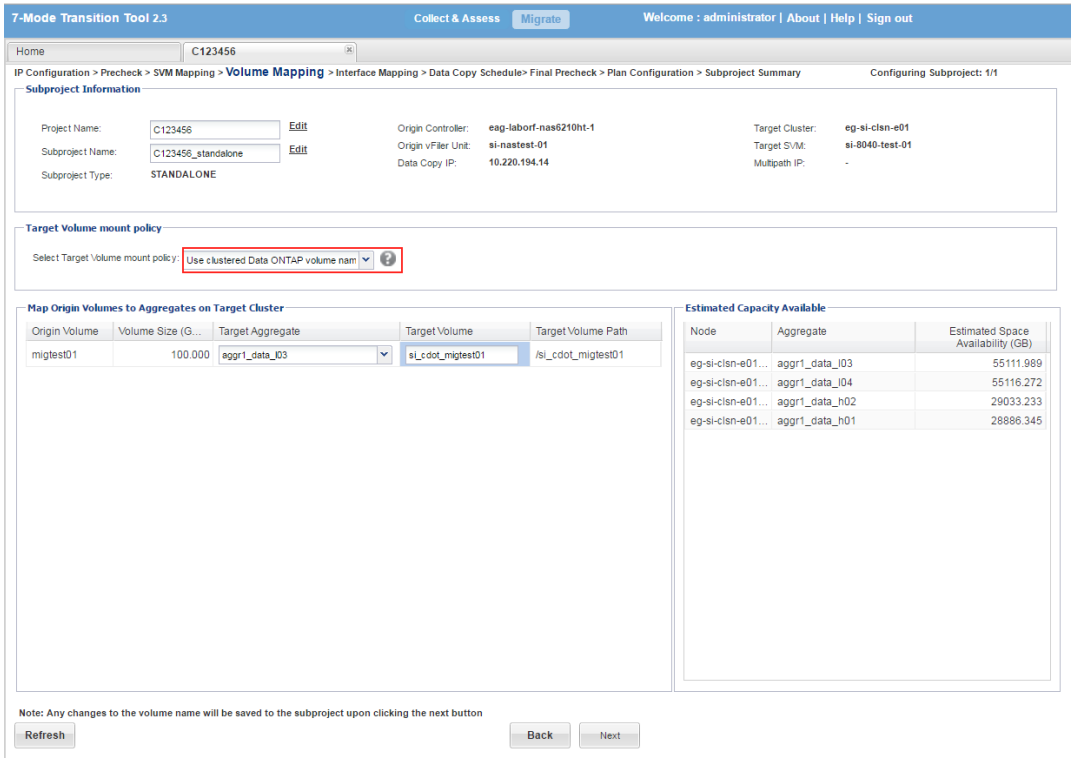
IMPORTANT: Make sure you enter the correct information in the next section! This version of 7MTT allows you to set junction path = volume name, this is the DCO standard. But, you need to modify the junction-path after the cut-over manually in older versions of 7MTT.

ENTER the destination aggregate

ENTER the destination volume name

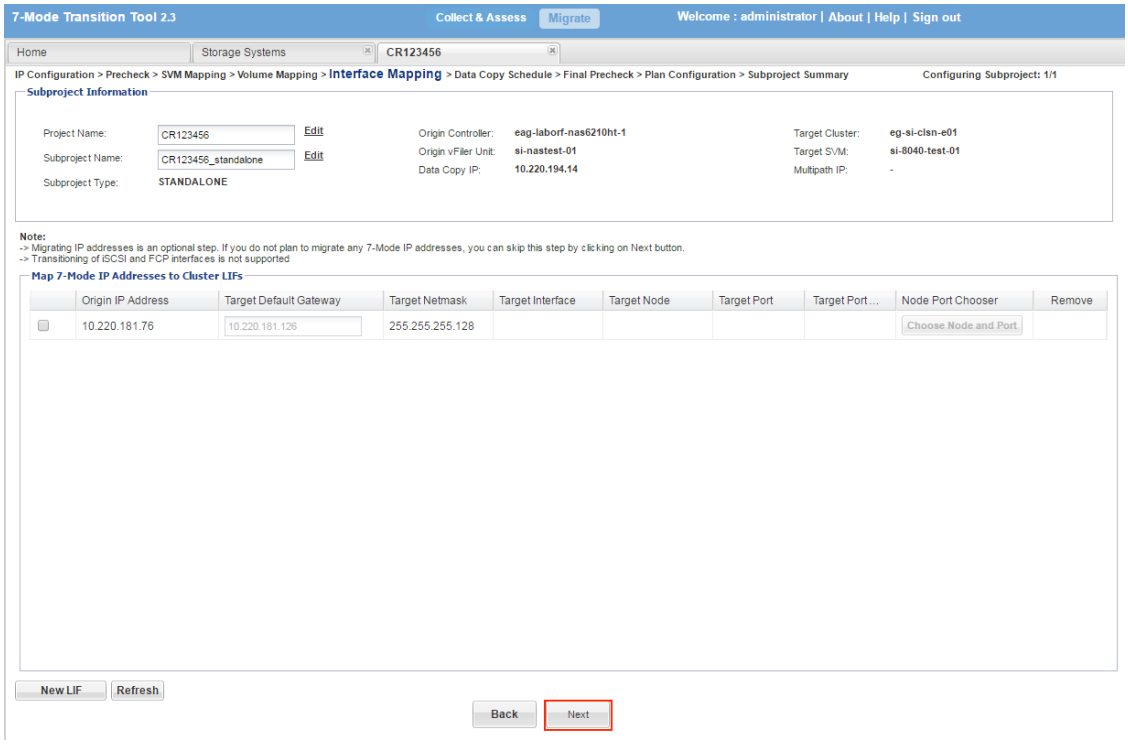
SELECT 'Use Clustered OTAP volume name' from 'Target Volume Mount Policy'

CLICK 'Next'



NOTE: We are not migrating IP addresses

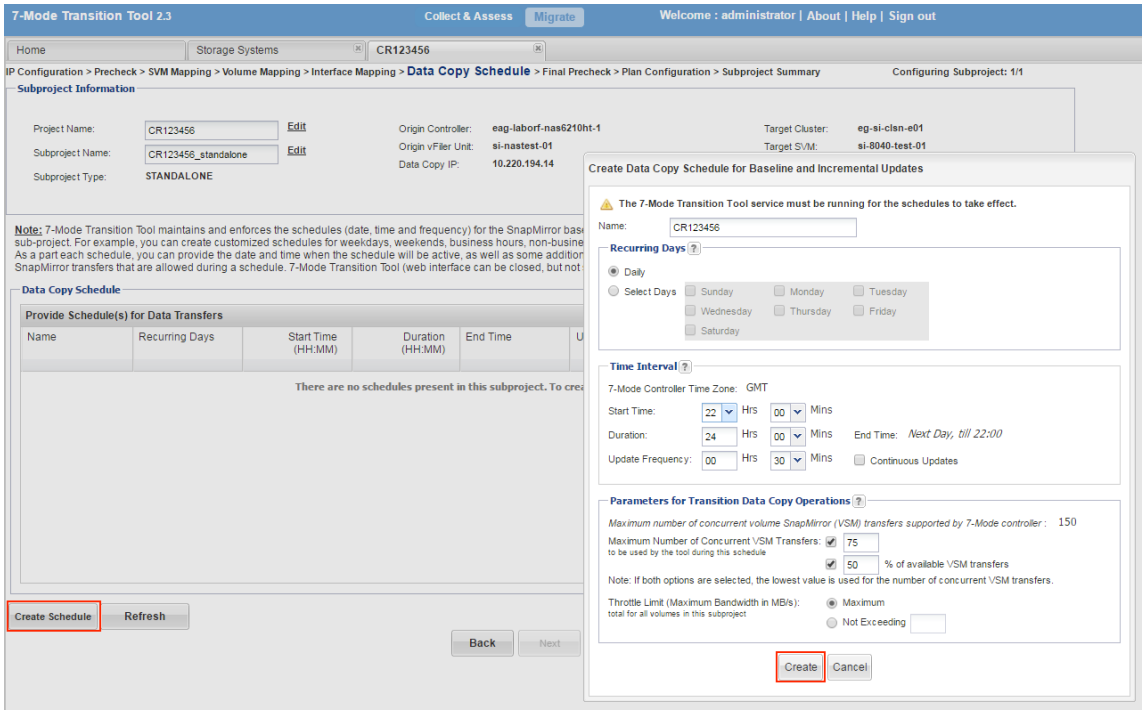
CLICK 'Next'



CLICK ‘Create Schedule’, a popup box will appear

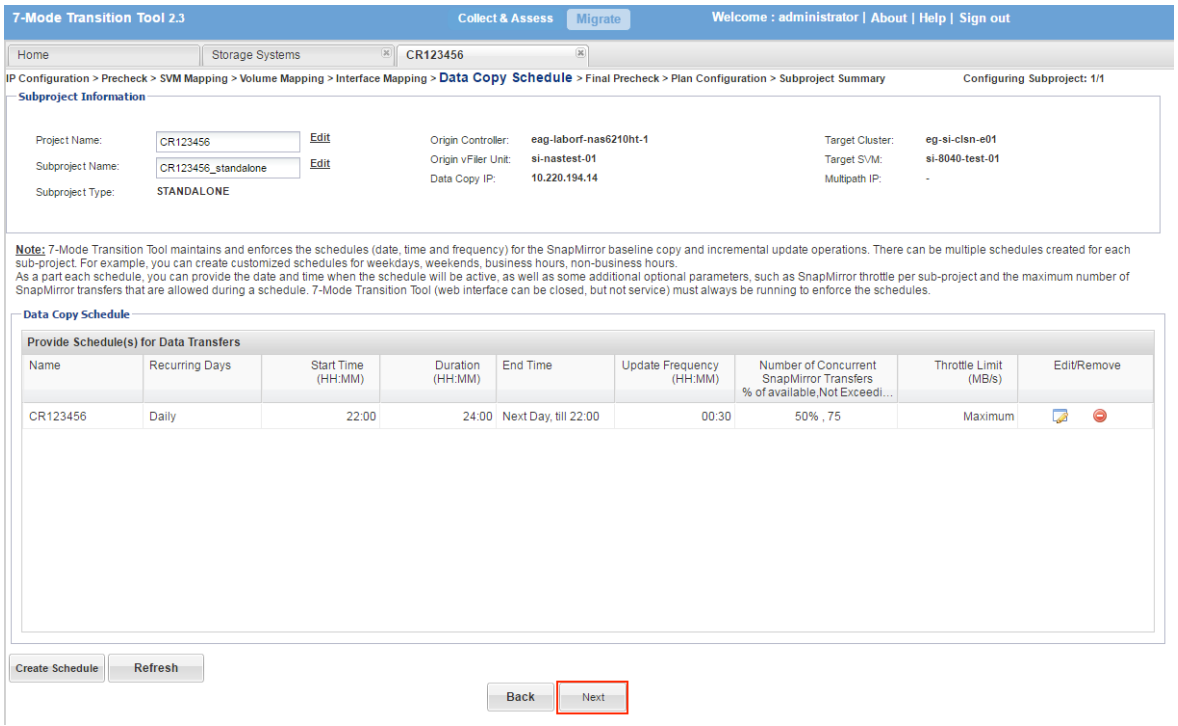
ENTER a replication schedule (run off hours)

CLICK CREATE



The newly created schedule will appear

CLICK ‘Next’

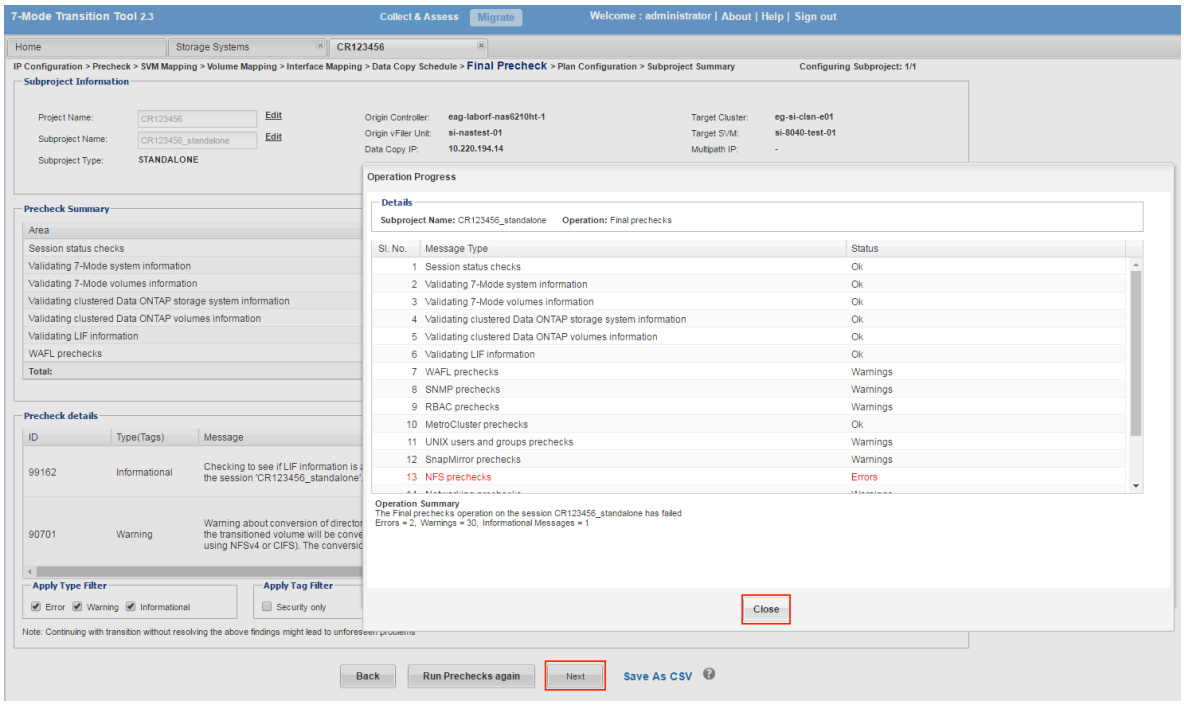


RUN PreCheck and make sure there are no errors

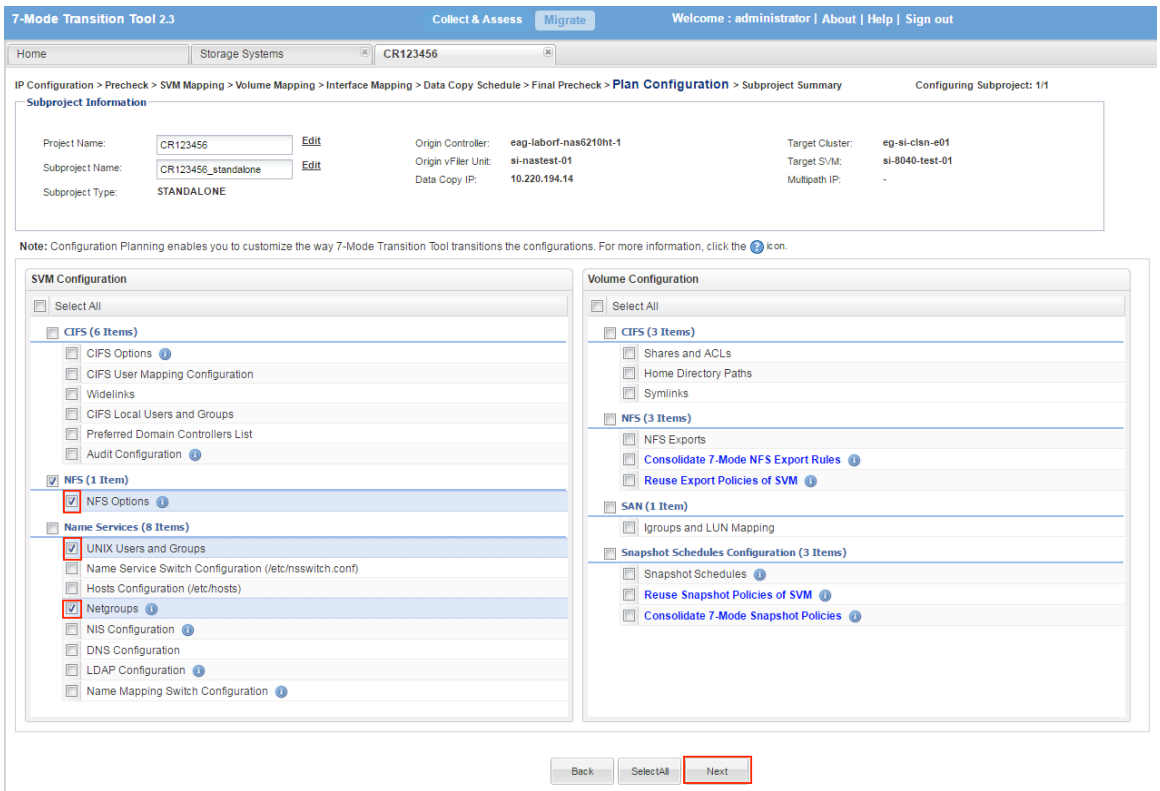
CLICK ‘Run Precheck’; REVIEW the output

CLICK ‘Close’

CLICK ‘Next’



Modify the options to remove exports creation on destination vserver.

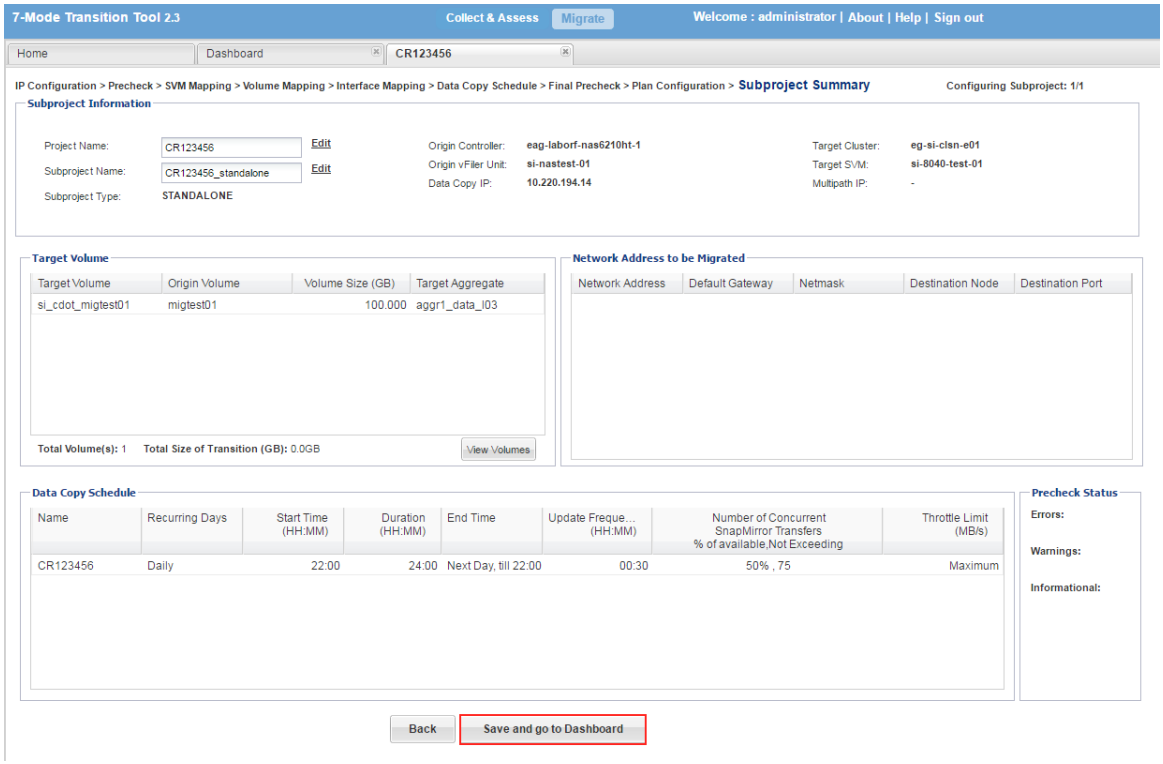


Relevant Options Description

Option	Description	Set
NFS Options	If the NFS options are not selected for transition, the following 7-Mode NFS options are not transitioned to the clustered Data ONTAP: nfs.udp.xfersize, nfs.v4.id.domain, nfs.v4.acl.max.aces, nfs.tcp.xfersize, nfs.rpcsec.ctx.high, nfs.rpcsec.ctx.idle, nfs.response.trigger, wafl.default_nt_user, nfs.mount_rootonly, nfs.tcp.enable, nfs.udp.enable, nfs.response.trace, nfs.v4.read_delegation, nfs.v4.write_delegation, nfs.v4.acl.enable, nfs.vstorage.enable, nfs.v3.enable, nfs.v4.enable.	Yes
UNIX Users and Groups	Copies local user/group information	Yes
Name Service Switch Configuration (/etc/nsswitch.conf)	Copies nsswitch.conf across	No
Host Configuration (/etc/hosts)	Copies host entries over	No
Netgroups	If Netgroups are selected for transition, all the existing Netgroups on the target SVM are replaced with the Netgroups transitioned from the 7-Mode. If there are no Netgroups configured in the '/etc/netgroups' file of the 7-Mode system, then existing Netgroups on the target SVM are retained	Yes
NIS Configuration	If NIS configuration is not selected for transition, NIS is not added to the name service switch on the target SVM.	No
DNS Configuration	Copies DNS configuration over	No
LDAP Configuration	If LDAP configuration is not selected for transition, LDAP is not added to the name service switch and Name mapping switch on the target SVM.	No
Name Mapping Switch Configuration	If LDAP configuration is not selected for transition, LDAP is not added to the name mapping switch on the target SVM even if 7-Mode has the option 'ldap.usermap.enable' set to on. You must manually add LDAP to the name mapping switch after LDAP is configured on the target SVM.	No
NFS Exports	Create export policies on destination	No
Consolidate Export Rules	If this option is selected, only one NFS export policy is created on the target SVM for all the volumes and qtrees with matching 7-Mode export rules. And the created NFS export policy is used to export all those volumes/qtrees on the target SVM. If the 'Reuse the export policies of SVM' option is selected, and if there is a matching NFS export policy existing on the target SVM, the pre-existing policy is reused rather than creating a new export policy.	No
Re-Use Export Policies	If this option is selected, and if there is a pre-exisitng NFS export policy that matches with 7-Mode export policy, the pre-exisitng export policy is reused rather than creating a new export policy.	No
Snapshot Schedules	If the Snapshot schedules are not selected for transition, Snapshot schedules of 7-Mode volumes are not transitioned to clustered Data ONTAP and the 'default' Snapshot policy of the target SVM is assigned to the transitioned volumes.	No
Reuse Snapshot Policies of SVM	If this option is selected, and if there is a pre-exisitng Snapshot policy that matches with 7-Mode Snapshot policy, the pre-exisitng policy is reused rather than creating a new policy.	No
Consolidate 7-Mode Snapshot Policies	If this option is selected, only one Snapshot policy (with all the required schedules) is created on the target SVM for all the 7-Mode volumes with matching 7-Mode Snapshot schedules. And the created Snapshot policy is used for all those volumes on the target SVM. If the 'Reuse Snapshot Policies of SVM' option is selected, and if there is a matching Snapshot policy existing on the target SVM, the pre-existing policy is reused rather than creating a new policy.	No

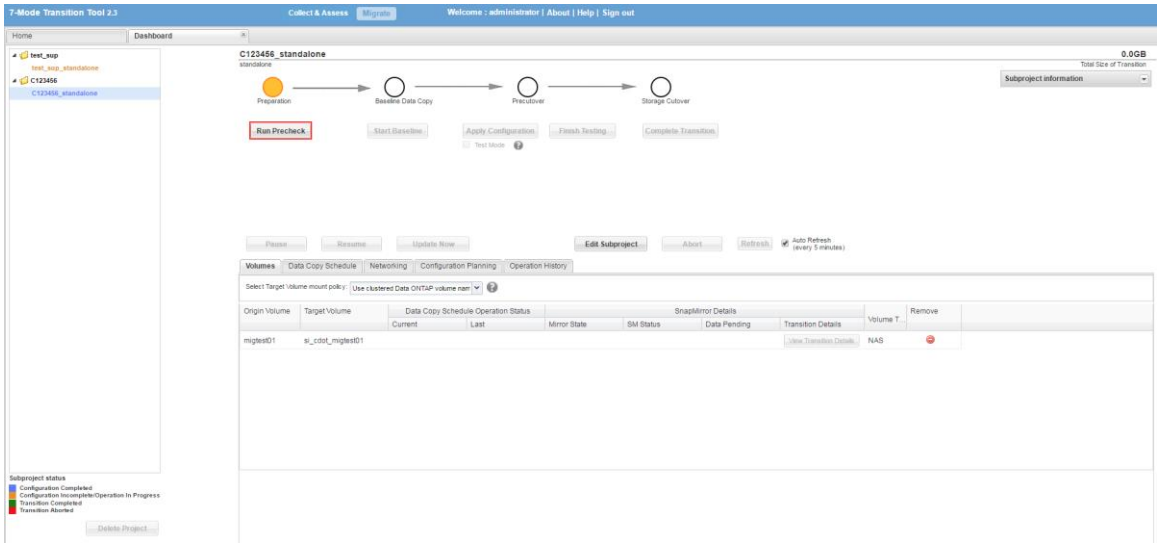
Click Next

CLICK ‘Save and go to Dashboard’

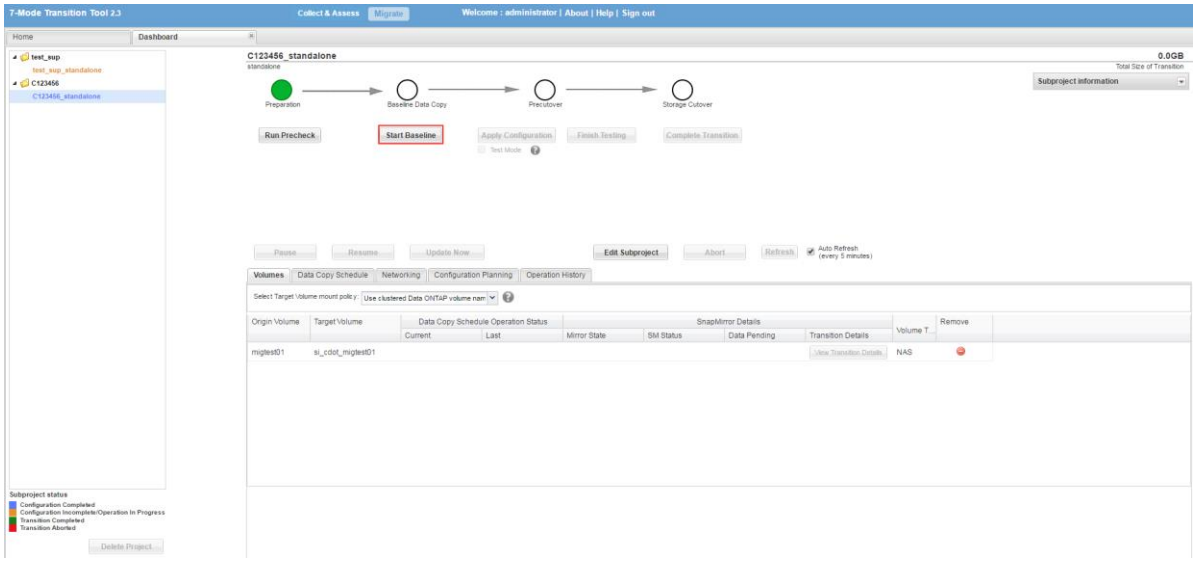


A data transfer workflow will be displayed in the top of the page. The circle above ‘Baseline Data Copy’ will have a white fill color before your first transfer. It will have an orange fill color while the first transfer is running. Then a green fill color after your first successful transfer.

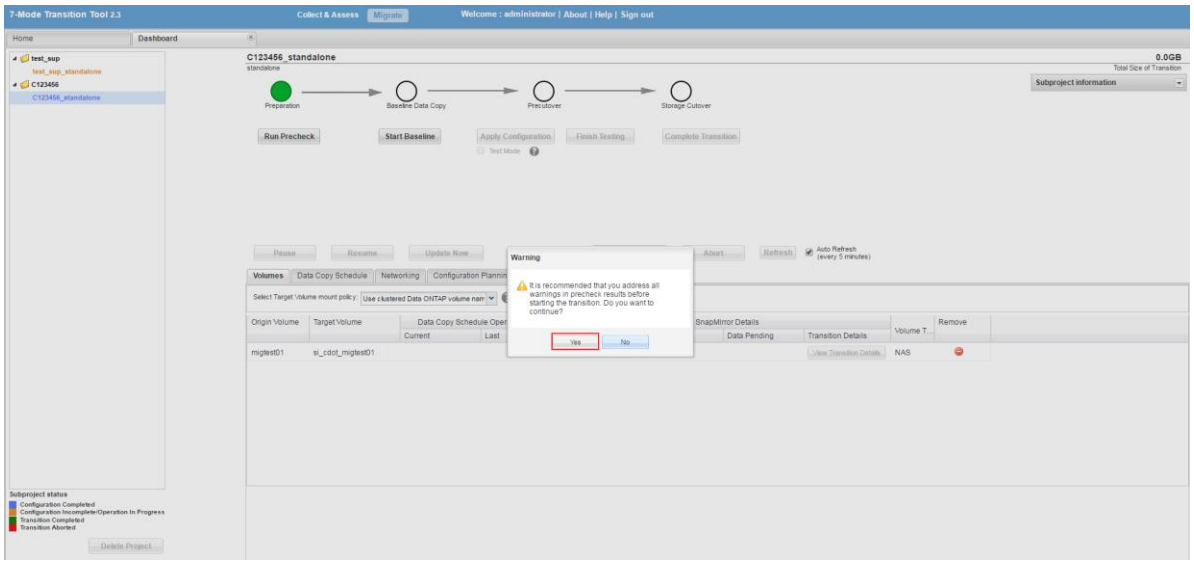
CLICK ‘Run Pre-Check’; REVIEW output



CLICK ‘Start Baseline’

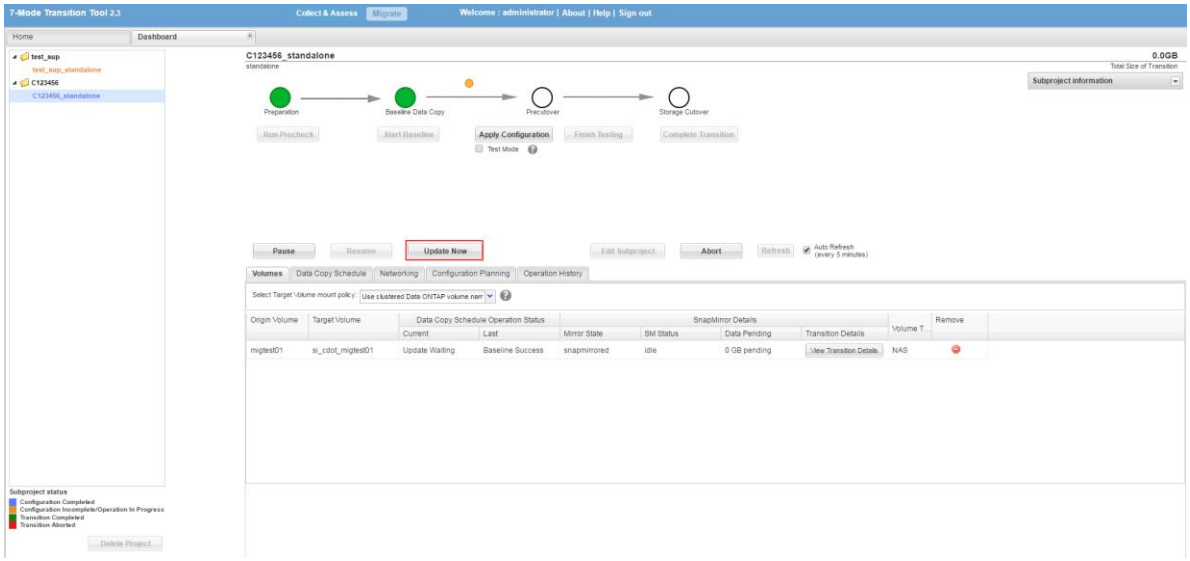


A popup window will appear, CLICK ‘yes’



Execute an update transfer before the cutover window so that you have minimal changes to transfer during the cutover window:

CLICK ‘Update Now’



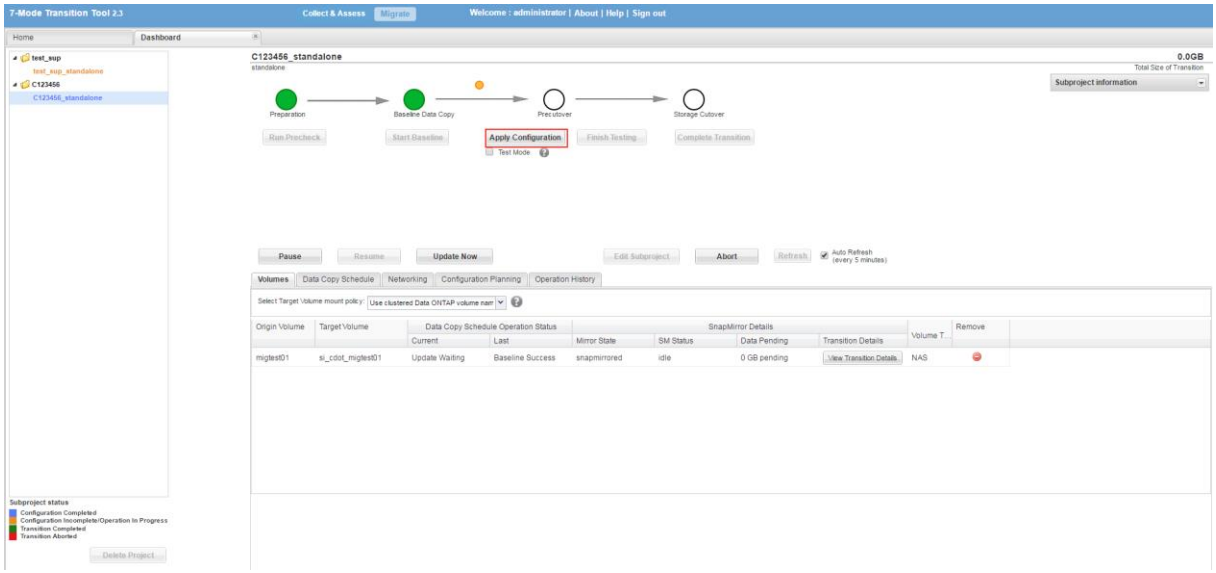
Ensure the steps below are completed during the downtime window:

The application/database has been shut down on all hosts

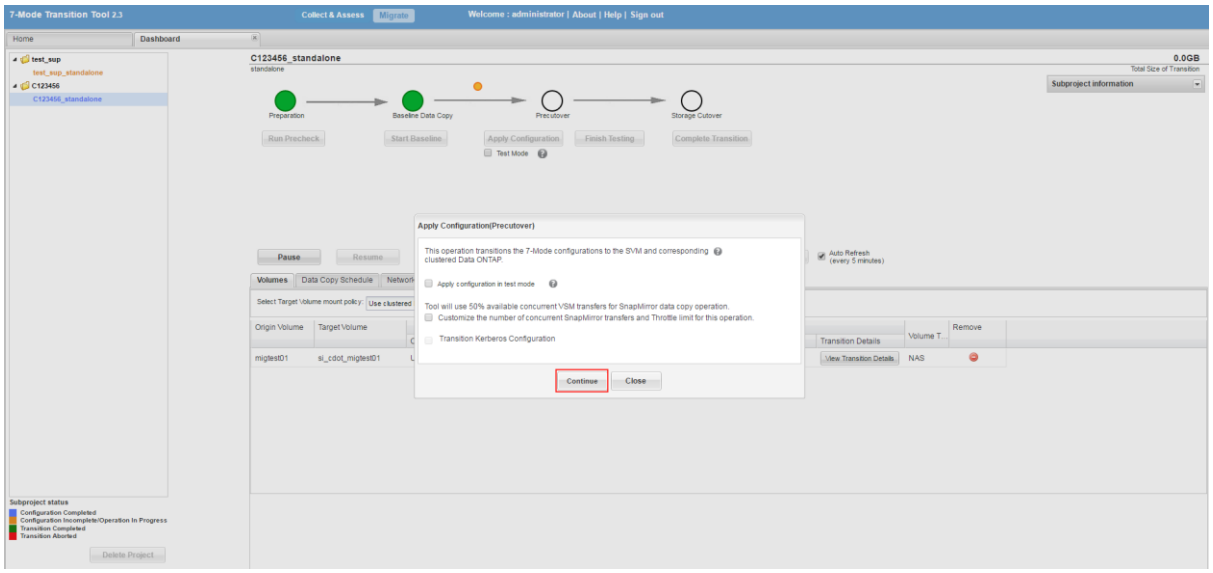
Source volumes have been unmounted on all hosts

Start the cutover process:

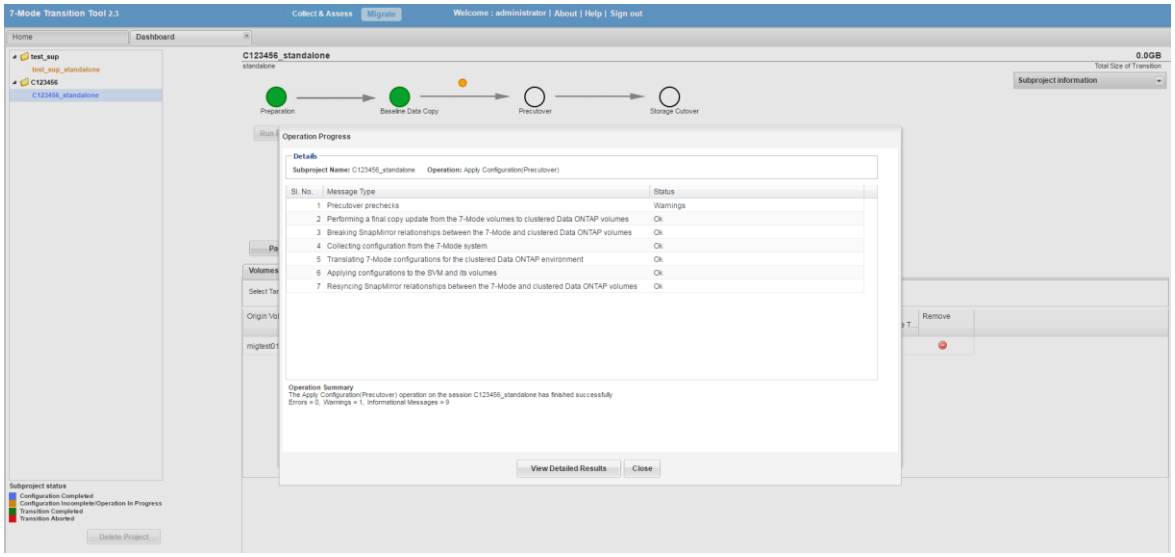
CLICK ‘Apply Configuration’



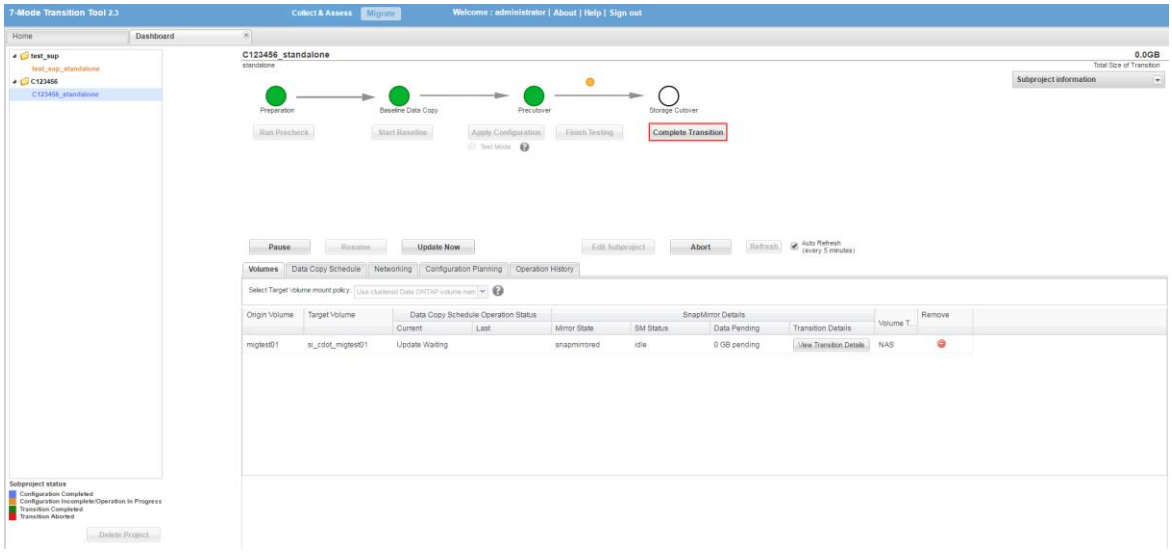
A popup box will appear, CLICK ‘Continue’



REVIEW log for errors and CLICK ‘Close’

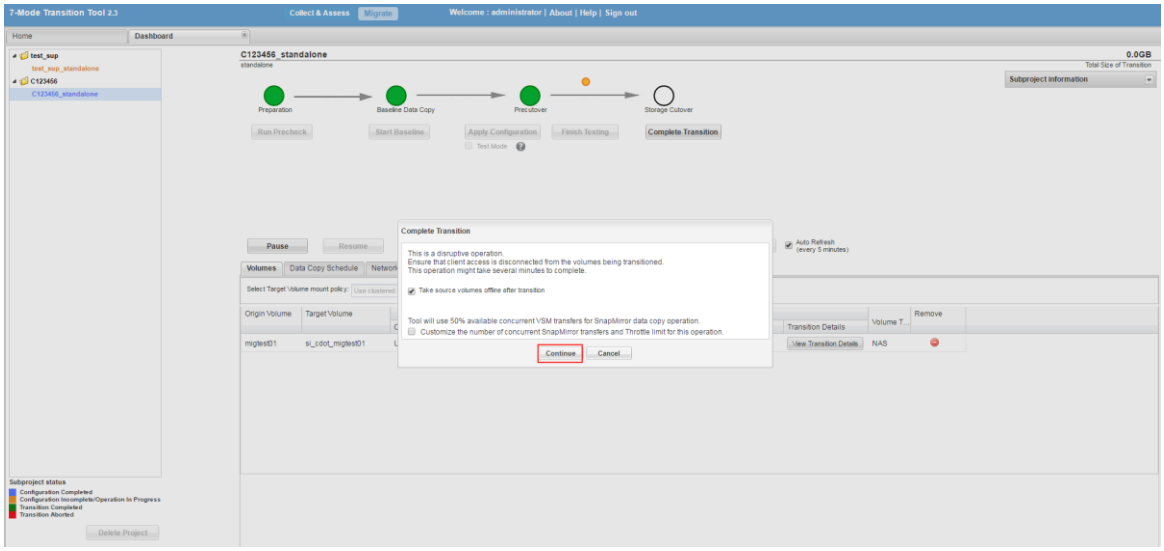


CLICK ‘Complete Transition’



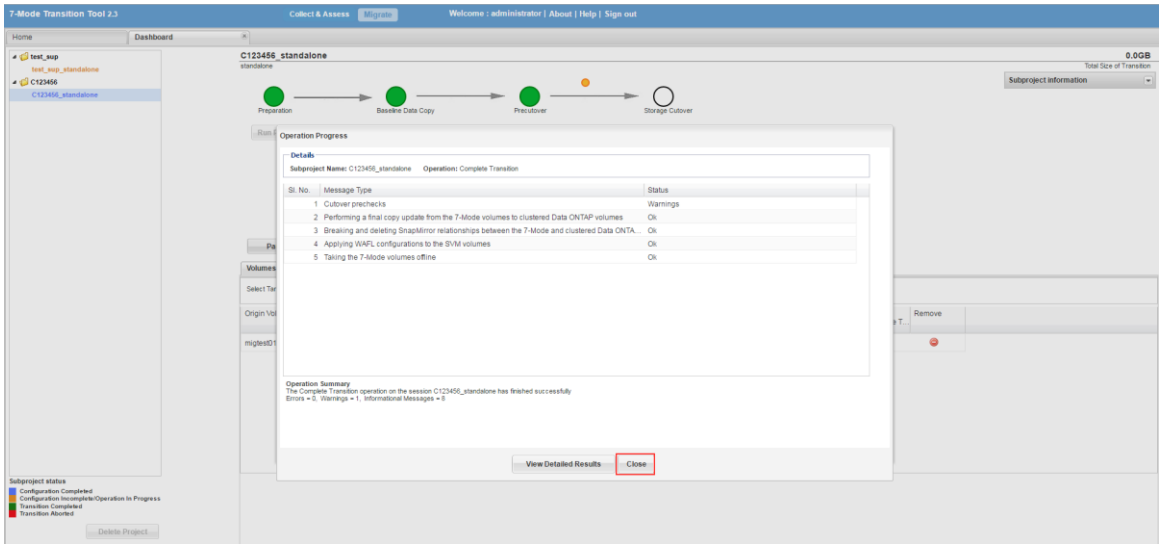
A popup box will appear, CLICK ‘yes’

A popup box will appear stating that the source volume will be made offline, CLICK ‘Continue’



REVIEW the log output for a successful transition

CLICK ‘Close’



2.9 Post cut-over activities

2.9.1 Post Checks

- Check Only default export policy exists on destination
- Check any local users and groups were transferred
- Check any netgroups were transferred

2.9.2 Create job schedule and snapshot policy for SNAP volume(s)

```
job schedule cron create -name <volume_name> -minute <min> -hour <hour>
```

```
volume snapshot policy create -vserver <vsname> -policy <volume_name> -enabled true -schedule1 <volume_name> -count1 7 -snapmirror-label1 snapvault -prefix1 sv_<volume_name>
```

```
job schedule cron show -name <vol_name>
```

```
volume snapshot policy show –vserver <vsname>
```

2.9.3 Export Policy Script

There is a script which can be run post migration to create volume export policies and also populate the default and volume export policy rules. The script arguments are shown below.

```
7-c_nfsexports.pl
```

```
Usage: ./7-c_nfsexports.pl --file <file name> --cb <CBEntry> --vserver <vserver name> --type <DB or STD> --user <user name> --cluster <C-DOT clustername>
```

Example

The following example will create volume export policies and populate the rules in both the default policy and volume export policies. The 7-Mode exports file used is below:

```
#Auto-generated by setup Tue Aug 25 23:10:21 GMT 2015
/vol/si_nastest_01_root -sec=sys,rw,anon=0,nosuid
#/vol/si_nas_test01     -sec=sys,rw,nosuid
#/vol/Image1    -sec=sys,rw,nosuid
#/vol/nfstest01 -sec=sys,rw,nosuid
#/vol/nfstest02 -sec=sys,rw,nosuid
/vol/nfstest01  -sec=sys,rw=10.220.177.29,root=10.220.177.29
/vol/nfstest02/testqt01 -sec=sys,rw=10.220.177.29,root=10.220.177.29
```

You provide the cluster admin name/IP and the admin user. It is assumed you have SSH public key access.

```
7-c_nfsexports.pl --file ./expfile  --cb cb123 --vserver si-8040-test-02 --type STD --user admin --
cluster 10.220.194.42


Extacting the data from supplied exports file.........
Extracting the hosts list by options(ro,rw,root)......
Bringing up the qtree hosts to volume.................
Eliminating the duplicate hosts at vol level..........
generating volume export policies........
Collect default and volume based exports: OUT ............
```

The script creates all of the objects on the cluster as shown below:

Default Export Policy Rules

```
eg-si-clsn-e01::> export-policy rule show -vserver si-8040-test-02 -policyname default
  (vserver export-policy rule show)
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
si-8040-test-02
             default         1       any      10.220.177.29         sys
```

Volume Export Policy Rules

```
eg-si-clsn-e01::> export-policy rule show -vserver si-8040-test-02 -policyname cb123_nfstest01  -
fields rorule ,rwrule ,superuser
  (vserver export-policy rule show)
vserver         policyname      ruleindex rorule rwrule superuser
--------------- --------------- --------- ------ ------ ---------
si-8040-test-02 cb123_nfstest01 1         sys    sys    sys

eg-si-clsn-e01::> export-policy rule show -vserver si-8040-test-02 -policyname cb123_nfstest02  -
fields rorule ,rwrule ,superuser
  (vserver export-policy rule show)
vserver         policyname      ruleindex rorule rwrule superuser
--------------- --------------- --------- ------ ------ ---------
si-8040-test-02 cb123_nfstest02 1         sys    sys    sys
```

Volume Export Policies

```
vol show -vserver si-8040-test-02 -fields policy
  (volume show)
vserver         volume          policy
--------------- --------------- ---------------
si-8040-test-02 cb123_nfstest01 cb123_nfstest01
si-8040-test-02 cb123_nfstest02 cb123_nfstest02
si-8040-test-02 rootvol         default
3 entries were displayed.
```

2.9.4 Setup snap autodelete on volumes (run for each SNAP volume)

```
volume modify -vserver <vsname> -volume <volname> -space-mgmt-try-first snap_delete

volume snapshot autodelete modify -vserver <vsname> -volume <vol_name> -enabled true

volume snapshot autodelete modify -vserver <vsname> -volume <vol_name>  -trigger snap_reserve

volume snapshot autodelete show -vserver <vsname>

volume show -vserver <vsname> -fields space-mgmt-try-first

volume snapshot autodelete show -vserve <vsname>
```

2.10 QoS Policy Group Creation For Shared Filers

All shared filer volumes have a QoS policy set at 6000iops as shown below. Exceptions to this policy must be agreed by D&E management on a case by case basis.

2.10.1 Create QoS policy group and apply it at the volume level (create one QOS policy for each volume)

```
qos policy-group create -policy-group <volname> -vserver <vsname> -max-throughput 6000iops
volume modify -vserver <vsname> -volume <volume> -qos-policy-group <volname>
qos policy-group show
volume show –vserver <vsname> -fields qos-policy-group
```

2.11 QoS Policy Group Creation For Dedicated Filers

All dedicated filer volumes have a QoS policy set at INF as shown below. This enables statistics to be collected but no limits are placed on the volume with regard to IOPS.

2.11.1 Create QoS policy group and apply it at the volume level (create one QOS policy for each volume)

```
qos policy-group create -policy-group <volname> -vserver <vsname> -max-throughput INF
volume modify -vserver <vsname> -volume <volume> -qos-policy-group <volname>
qos policy-group show
volume show –vserver <vsname> -fields qos-policy-group
```

3 Snapvault Configuration

3.1 Cluster and Vserver Peering

3.1.1 Confirm that cluster peering has been enabled

```
cluster peer show
```

3.1.2 Create the cluster peer (skip this step if cluster peering has been configured)

```
cluster peer create -peer-addrs <remote_ICL_IP1,remote_ICL_IP2> -username admin

cluster peer show
```

3.1.3 Confirm if vserver peering has been configured

```
vserver peer show
```

3.1.4 Create vserver peering on the destination system (skip this step if vserver peering has been configured)

```
vserver peer create -vserver <destination_vserver> -peer-cluster <source_cluster> -peer-vserver
<source_vserver> -applications snapmirror

vserver peer show
```

3.1.5 Accept the vserver peering on the source system

```
vserver peer accept -vserver <source_vserver> -peer-vserver <destination_vserver>

vserver peer show
```

3.2 SnapVault configuration

Volumes names in TR have ‘SNAP’ or ‘NOSNAP’ incorporated into them. SNAP volumes must have snapvault configured for disk based backups. NOSNAP do NOT require backups.

3.2.1 Create secondary volumes for SnapVault as type “DP” on the destination cluster

```
volume create -vserver <vserver> -volume <volume_name> -aggregate <aggr_name> -size <size> -
security-style unix -space-guarantee none -percent-snapshot-space 0 -language <vol_language> -type
DP

volume show
```

3.2.2 Create a cron job schedule if it does not exist in the destination

```
job schedule cron create -name xdp_<hour> -minute 00 -hour <hour>
job schedule show
```

3.2.3 Configure a snapmirror policy on the destination

```
snapmirror policy create -vserver <vserver> -policy <volume>

snapmirror policy add-rule -vserver <vserver> -policy <volume> -snapmirror-label snapvault -keep
<retention#>

snapmirror show -destination-path * -fields Schedule

snapmirror policy show
```

3.2.1 Initialize SnapVault relationship on the destination

```
snapmirror create -source-path <source_vserver>:<source_volume> -destination-path
<destination_vserver>:<destination_volume> -type XDP -schedule <schedule_name> -policy
<policy_name>

snapmirror initialize -destination-path <destination_vserver>:<destination_volume>
```

```
snapmirror show
```

4 Current Post Migration Configuration Script (4.7)

The following script can be run post 7MTT migration and will create all the required export policies and rules.

```
#!/usr/bin/perl
#Scriptname:7-c_nfsexports.pl
#Author:Sridhar Chevendra
#Version:4.5
#Version release
#V1.0 -02/06/2016 - Created by SCDraft version
#V2.0 -02/07/2016 - Updated by SCExtract hosts by access
#v3.0 -02/08/2016 - Updated by SCRemove duplicates hosts by volumes,Generate default and vol based
commands
#v4.0 -02/09/2016 - Updated by SCAdded command line options, Address multi-volume iterations and Final
release
#v4.2 -02/09/2016 - Updated by SC   Introduced vserver based out,log and error collection.
#v4.4 -02/13/2016 - Updated by SC   Removed #, added commands for DB LION/LOON, added ro wide open
option
#v4.5 -02/19/2016 - Updated by SC   Removed 0.0.0.0/0 entry
#v4.6 -04/07/2016 - Updated by SC       Creating the export policy,removing duplicate host entries bug
#v4.7 -04/08/2016 - Updated by SC       Script to support 7MTT 2.3, Additional fields for cluster, user
name and Applying export policy on the vservers

#use strict;
use warnings;
use Data::Dumper;
use Getopt::Long;
use Array::Utils qw(:all);

(@ARGV) or die "Usage: $0 --file <file name> --cb <CBEntry> --vserver <vserver name> --type <DB or STD>
--user <user name> --cluster <C-DOT clustername>";

 GetOptions(             "file=s" => \my $file,        #7-mode NFS exports file
                    "cb=s" => \my $cb,          #Charge back code for volumes
                    "vserver=s" => \my $vname,  #C-DOT vserver name
                                "type=s" => \my $type,               #DB vs. standard NAS
                                "user=s" => \my $user,               #User name to login to C-DOT
cluster
                                "cluster=s" => \my $cluster,  #C-DOT DNS clustername or IP address
                        ) or die "Usage: $0 --file <file name> --cb <CBEntry>  --vserver <vserver name>
--type <DB or STD> --user <user name> --cluster <C-DOT clustername> \n";

(-f $file) or die "7-mode exports file is needed to proceed";
($cb) or die "CB code is needed in order to proceed";
($vname) or die "C-dot filer vserver name is needed to proceed";
($type) or die "Specify DB or STD";
($user) or die "Username required";
($cluster) or die "C-DOT cluster name is missing";

open ( STDERR, '>', "$vname\_err" ) or die "can't open STDERR file: err:$! \n";
open ( OUT, '>', "$vname\_out" ) or die "can't open OUT file: out:$! \n";
open ( LOG, '>', "$vname\_log" ) or die "can't open OUT file: out:$! \n";
my $conn = "$user\@$cluster";

#my @keys = qw/rw ro root/;
#my $wanted = join "|",@keys;

my %data;
my $rid=1;                      #Default rule index
my $riv;                        #Default volume index

print "Extacting the data from supplied exports file..... \n";
print LOG "Extacting the data from supplied exports file..... \n";

&exportsfile_dataextraction ();

print "Extracting the hosts list by options(ro,rw,root)..... \n";
print LOG "Extracting the hosts list by options(ro,rw,root)..... \n";
print "Bringing up the qtree hosts to volume.................. \n";
print LOG "Bringing up the qtree hosts to volume.................. \n";

&extracthosts_byaccess('ro','rw','root');

print "Eliminating the duplicate hosts at vol level.......... \n";
print LOG "Eliminating the duplicate hosts at vol level.......... \n";
```

```
#&eliminateduplicatehosts();

#print "Generating Default policy host access commands........ \n";
#print LOG "Generating Default policy host access commands........ \n";

&generatedefaultexportcommands();

print  "generating volume export policies........ \n";
print LOG "generating volume export policies........ \n";


#&generatevolexportcommands ();
&generatevolexportcommands () if ( $type eq 'STD' );
&generatevoldbexportcommands () if ( $type eq 'DB' );

print LOG "Collect default and volume based exports: T............ \n";
print "Collect default and volume based exports: OUT ............ \n";

sub exportsfile_dataextraction (){

   #my $file='exportfs.txt';
   my @keys = qw/rw ro root/;
   my $wanted = join "|",@keys;

   open (my $r_fh, $file) or die "can not open file:$file:$! \n";

   while(<$r_fh>){

                next unless /^(\/)/;
                next if /_root/;
                #$_ =~ s/ro(\,|\s)/ro=0.0.0.0\/0\,/;
                $_ =~ s/ro(\,|\s)//;
                my ($path, $options) = split;
                my ($vol, $q) = (split '/', $path) [2,3];
                my %tmp = map {split /=/} grep /^(?:$wanted)/, split /,/, $options;

                $data{$vol}{$q || 'nq'} = \%tmp;
        }
   print LOG Dumper \%data;

   close $r_fh;
}

sub extracthosts_byaccess () {
   while( $option = shift ( @_ ) ){
   #print $option;
   foreach my $vol ( sort keys %data ){

               foreach my $qtree ( keys %{$data {$vol} } ){

                        # print  "$data{$vol}{$qtree}{rw}";
                               $arrayname = "$vol\_$option";
                               # print $arrayname, "\n";
                               my @tmp = split /:/, $data{$vol}{$qtree}{$option};
                               @ $arrayname = ( @ $arrayname, @tmp);

                        }
                               print LOG "============= \n";
                               print LOG "$arrayname \n";
                               print LOG Dumper @ $arrayname;
                }
        }
   }


sub eliminateduplicatehosts () {
                                        $vol = shift ( @_ );
                                        print LOG $vol, "\n";

                                        $rohosts = "$vol\_ro";
                                        print LOG $rohosts,"\n";
                                        print LOG Dumper @ $rohosts;

                                        $rwhosts = "$vol\_rw";
                                        print LOG $rwhosts,"\n";
                                        print LOG Dumper @ $rwhosts;

                                        $roothosts = "$vol\_root";
                                        print LOG $roothosts,"\n";
```

```
                                        print LOG Dumper @ $roothosts;


                                        #Remove duplicate in an array
                                        print LOG "Refined list and remove duplicate host entries.......
\n";
                                        @ $rohosts = unique( @ $rohosts );
                                        @ $rwhosts = unique( @ $rwhosts );
                                        @ $roothosts = unique( @ $roothosts );


                                        print LOG $rohosts, "\n";
                                        print LOG @ $rohosts, "\n";
                                        print LOG $rwhosts, "\n";
                                        print LOG @ $rwhosts, "\n";
                                        print LOG $roothosts, "\n";
                                        print LOG @ $roothosts, "\n";

                                        $frohosts = "$vol\_rof";
                                        $frwhosts = "$vol\_rwf";
                                        $froothosts = "$vol\_rootf";
                print LOG "Remove duplicates between RO and RW hosts and genetate final lists.......
\n";

                if ( !array_diff ( @ $rohosts, @ $rwhosts ) ) {
                        print LOG "RW and RO hosts are same and RO Entries will be empty....... \n";
                        @ $frwhosts = @ $rwhosts;
                        @ $frohosts = ();

                } else {


                        my @tmpisect = intersect ( @ $rohosts, @ $rwhosts );
                        @ $frohosts = array_diff ( @tmpisect, @ $rohosts );
                        @ $frwhosts = @ $rwhosts;

                        print LOG "Finding the common elements between RO and RW arrays .... \n";
                        print LOG Dumper @tmpisect;
                        print LOG "Final RO hosts...\n";
                        print LOG @ $frohosts, "\n";
                        print LOG "Final list of RW hosts... \n";
                        print LOG Dumper @ $frwhosts, "\n";

                        }
                print LOG "Remove duplicates between RW and ROOT hosts and generate final lists...\n";

                if ( !array_diff ( @ $frwhosts, @ $roothosts ) ) {

                                #@ $froothosts = @ $rwhosts;
                                                @ $froothosts = @ $frwhosts;
                                                @ $frwhosts = ();

                                                print LOG "ROOT and RW hosts are same and RO
Entries will be empty....... \n";
                                                print LOG @ $froothosts, "\n";
                                                print LOG @ $frwhosts, "\n";
                        } else {

                                                my @tmpisect = intersect ( @ $frwhosts, @
$roothosts );
                                @$frwhosts = array_diff ( @tmpisect, @ $frwhosts );
                                @$froothosts = @ $roothosts;

                                                print LOG "Finding the common elements between
ROOT and RW arrays .... \n";
                                                print LOG Dumper @tmpisect;
                                                print LOG "Final RW hosts...\n";
                                                print LOG @ $frwhosts, "\n";
                                                print LOG "Final list of Root hosts... \n";
                                                print LOG Dumper @ $froothosts, "\n";

                                                }
          print LOG Dumper @ $frohosts, @ $frwhosts, @ $froothosts;
   }



sub generatedefaultexportcommands (){
   foreach my $vol ( sort keys %data ){
```

```
          print LOG "Eliminating duplpicates...., /n";

          &eliminateduplicatehosts ($vol);

          my @default = ( @ $frohosts, @ $frwhosts, @ $froothosts );
          @fdefault = ( @fdefault, @default);
          @fdefault = unique ( @fdefault );
          }

          @fdefault = qw(0.0.0.0/0) if ( grep ( /0.0.0.0\/0/, @fdefault) );

                  while ( $client = shift ( @fdefault )){

                           #print OUT "vserver export-policy rule create -vserver $vname -policyname
default -ruleindex $rid -protocol any -clientmatch $client -rorule sys -rwrule never -superuser never
\n";
                           #print OUT "vserver export-policy rule create -vserver $vname -policyname
default -ruleindex $rid -protocol any -clientmatch $client -rorule sys -rwrule never -superuser none
\n";
                           my $cmd = "vserver export-policy rule create -vserver $vname -policyname
default -ruleindex $rid -protocol any -clientmatch $client -rorule sys -rwrule never -superuser none";
                           print OUT "ssh $conn $cmd \n";
                           `ssh $conn $cmd \n`;
                           $rid++;
          }
   }


sub generatevolexportcommands (){
   foreach my $vol ( sort keys %data ){

          print LOG "$vol \n";
          print LOG "========== \n";
          $riv=1;
          print LOG $riv;

          print OUT "Creating volume export policy";
          my $cmdvol = "export-policy create -vserver $vname -policyname $cb\_$vol";
          my $cmdvola = "vol modify -vserver $vname -volume $cb\_$vol -policy $cb\_$vol";
          print OUT "ssh $conn $cmdvol \n";
          `ssh $conn $cmdvol`;
          print OUT "ssh $conn $cmdvola \n";
          `ssh $conn $cmdvola`;

          &eliminateduplicatehosts ($vol);

          while ( $client = shift (@ $froothosts) ){

                  #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol -
ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys \n";
                  my $cmdrule = "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys";
                  print OUT "ssh $conn $cmdrule \n";
                  `ssh $conn $cmdrule`;
                  $riv++;
          }

          #Generating commands for rw access;

          while ( $client = shift (@ $frwhosts) ){

                  #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol -
ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys \n";
                  my $cmdrule = "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys";
                  print OUT "ssh $conn $cmdrule \n";
                  `ssh $conn $cmdrule`;
                  $riv++;
          }

          #Generating commands for ro access;

                while ( $client = shift (@ $frohosts) ){

                    #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys \n";
                                my $cmdrule = "vserver export-policy rule create -vserver $vname -
policyname $cb\_$vol -ruleindex $riv -protocol any -clientmatch $client -rorule sys";
```

```
                                print OUT "ssh $conn $cmdrule \n";
                                `ssh $conn $cmdrule`;
                                $riv++;
                }

                  }
}


sub generatevoldbexportcommands (){
   foreach my $vol ( sort keys %data ){

           print LOG "$vol \n";
           print LOG "========== \n";
           $riv=1;
           print LOG $riv;

           print OUT "Creating volume export policy";
           my $cmdvol = "export-policy create -vserver $vname -policyname $cb\_$vol";
           my $cmdvola = "vol modify -vserver $vname -volume $cb\_$vol -policy $cb\_$vol";
           print OUT "ssh $conn $cmdvol \n";
           `ssh $conn $cmdvol`;
           print OUT "ssh $conn $cmdvola \n";
           `ssh $conn $cmdvola`;

           &eliminateduplicatehosts ($vol);

           while ( $client = shift (@ $froothosts) ){

                   #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol -
ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys \n";
                   my $cmdrule = "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys";
                   print OUT "ssh $conn $cmdrule \n";
                   `ssh $conn $cmdrule`;
                   $riv++;
           }

           #Generating commands for rw access;

           while ( $client = shift (@ $frwhosts) ){

                   #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol -
ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys \n";
                   my $cmdrule = "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys -rwrule sys -superuser sys";
                   print OUT "ssh $conn $cmdrule \n";
                   `ssh $conn $cmdrule`;
                   $riv++;
           }

           #Generating commands for ro access;

                while ( $client = shift (@ $frohosts) ){

                     #print OUT "vserver export-policy rule create -vserver $vname -policyname $cb\_$vol
-ruleindex $riv -protocol any -clientmatch $client -rorule sys \n";
                                my $cmdrule = "vserver export-policy rule create -vserver $vname -
policyname $cb\_$vol -ruleindex $riv -protocol any -clientmatch $client -rorule sys";
                                print OUT "ssh $conn $cmdrule \n";
                                `ssh $conn $cmdrule`;
                                $riv++;
                }

                  }
}
```