ORACLE
OPTIMIZED SOLUTIONS

An Oracle Technical White Paper
March 2013

How to Accelerate Test and Development Through Rapid Cloning of Production Databases and Operating Environments

ORACLE

Introduction 1
Challenges with Creating and Maintaining Database Clones 2
Technical Steps Involved in Cloning an Oracle Database 2
Active Database Duplication 3
Backup-Based Duplication 3
Introducing the Oracle Featured Solution 6
Solution Components 8
Oracle Solaris 8
Oracle Solaris Zones 9
Oracle Solaris Network Virtualization 9
Deployment Example 10
Create a File System for Database Backup 12
Back Up the Production Database on the Sun ZFS Storage Appliance 17
Create a Clone of the Database Backup File System 18
Create an Oracle Solaris Master Zone 30
Perform the Clone Operation 32
Summary 33
For More Information 33
Appendix A—Solution Scripts 35
The `setup.sh` Script 35
The `cloneEnv.sh` Script 36
The `haltZone.sh` Script 37
The `exportCfg.sh` Script 37
The `cloneZone.sh` Script 38
The `cloneAllDB.sh` Script 39
The `cloneDB.sh` Script 40

The `runClone.sh` Script 41
The `addNFS.sh` Script 41
The `setenv.sh` Script 41
The `clonedb.pl` Script 42
The `createControl.sh` Script 46
The `verifyDB.sh` Script 47
The `delAll.sh` Script 47
The `delZone.sh` Script 47
Sample Output 48
Appendix B—Solution Hardware Component Descriptions 53
SPARC T4 Server Overview 53
SPARC T5-4 Server Overview 53
Comparison of SPARC T4 and T5 Server Models 54
Sun ZFS Storage Appliance 55

Introduction

There are a many reasons that IT organizations must clone a production database. Some of the most common include the following:

- **Supporting application development and testing.** Customers typically re-create a clone of their production databases from backups to support application development and testing, and to assess the performance impact as well as regression testing of operating systems patches and database updates on existing applications.
- **Updating application infrastructures without downtime.** Modern commercial Web and e-commerce applications require 100 percent availability. This presents a challenge for IT managers to keep the application infrastructure updated without incurring downtime. The database is typically cloned to create a foolproof way of testing updates or upgrades to the environment without impacting production systems.
- **Troubleshooting a system bug.** Sometimes IT personnel need to replicate an existing software environment with all installed patches from all components to re-create a bug or a system panic.

The process of cloning production databases can be time consuming for IT staff and, if governance is lacking, IT organizations can end up managing many varieties of database clones that are scattered throughout the IT infrastructure.

This paper describes a fast and efficient approach to clone the operating system and database environments, along with a hardware and software solution that can make the process more efficient with minimal risk.

The featured method for cloning an Oracle Solaris and Oracle Database environment utilizes Oracle Recovery Manager (Oracle RMAN) backup technology, the CloneDB feature of Oracle Database's Direct NFS Client (dNFS), the built-in virtualization feature of Oracle Solaris Zones, and the cloning capability of Oracle's Sun ZFS Storage Appliance to greatly simplify the process of replicating the Oracle Database environment. This paper describes the specific steps for deploying the server and storage infrastructure that take advantage of these technologies, to simplify the database cloning process and improve reliability and predictability for a production Oracle Database environment.

Challenges with Creating and Maintaining Database Clones

When a clone or a new database is required, IT personnel must procure additional hardware, servers, and storage, and then connect and integrate all the various components together. Administrators must install and patch operating systems to support the necessary application infrastructure, request additional storage to host the database files, and connect to or create a network environment. The entire process can take days, even weeks, negatively impacting user productivity and delaying strategic business goals. Over time, as new applications and supporting databases are implemented, the IT landscape becomes dotted with multiple deployment silos. Beyond the costs of acquiring hardware, software, storage, and networking, there are ongoing maintenance and support costs that can proliferate as servers and storage components multiply and the complexity of the landscape increases. As a result of sprawl and over-provisioning, traditional database deployments tend to exhibit underutilized systems, resulting in cost inefficiencies.

Another option is to virtualize the IT environment, and create a master virtual machine (VM) with a database installed. The master VM is then cloned as needed, thereby creating other VMs with installed databases on demand. This option might require shutting down the master VM before the cloning operation, and additional storage is needed to host the database files for the new, cloned VM. This option is also error-prone, as it requires modifications to the database-related files to reflect new directory locations, new hostnames, new IP addresses, and the restarting of services, often requiring the assistance of a DBA. Troubleshooting issues on the cloned VMs, and the time needed to make the database function properly, often negates the perceived benefits from cloning a working VM environment.

Technical Steps Involved in Cloning an Oracle Database

To address the issues described previously, DBAs duplicate the existing database by performing a clone operation on the source database. This section explains the steps typically involved in that process.

As explained in the Oracle Database 11*g* Release 1 documentation's "Duplicating a Database" topic, there are two possible ways to duplicate a database: active duplication and backup-based duplication.

Active database duplication copies the live source database over the network to the duplicate database instance, whereas backup-based duplication uses pre-existing Oracle RMAN–generated backups and copies. Both of these techniques require the completion of a number of prerequisite steps described in the documentation referenced above.

Active Database Duplication

In active database duplication, Oracle RMAN connects as TARGET to the source database instance and as AUXILIARY to the auxiliary instance associated with the duplicate database. Oracle RMAN copies the live source database over the network to the auxiliary instance, thereby creating the duplicate database. No backups of the source database are required.

Backup-Based Duplication

When using backup-based duplication, making database backups available to the auxiliary instance depends on whether both source and auxiliary database instances have access to a shared disk or an NFS-mounted disk. If the disk that holds the database backup is not accessible to the auxiliary instance, then the backup pieces need to be moved to the machine where the duplication of the database is to be performed. The main advantage of this method is that while duplicating the database, administrators don't need to connect to the source database, and hence there is no impact on the production system at all.

Duplication Steps

The process of duplicating a backed-up database includes the following steps:

- Create a backup of the source database.
- Procure a new server and install the OS and all the relevant patches required for database installation.
- Install the Oracle Database software on the duplicate or auxiliary host.
- Create the necessary directories for the auxiliary instance.
- Create an Oracle password file for the auxiliary instance.
- Create an initialization parameter file for the auxiliary instance.
- Modify parameters. All path parameters should be accessible on the duplicate host.
- Create the auxiliary instance.
- Establish Oracle Net Connectivity to the auxiliary instance.
- Start the auxiliary instance.
- Ensure access to the necessary backups and archive logs.
- Start Oracle RMAN and connect to the database instances.
- Mount or open the source database.

- Configure Oracle RMAN channels for use in the duplication.
- Run the Oracle RMAN DUPLICATE command.

As explained in the *Oracle Database Backup and Recovery User's Guide*, Oracle Database 11*g* Release 2 introduces the following mutually exclusive sub-techniques for backup-based duplication:

- Duplication without a target database connection. Oracle RMAN obtains the metadata about backups from a recovery catalog.
- Duplication without a target database connection and without a recovery catalog. A disk backup location containing all the backups or copies for duplication must be available to the destination host.
- Duplication with a target database connection. Oracle RMAN obtains the metadata about backups from the target database control file or from the recovery catalog.

For both Oracle Database 11*g* Release 1 and Release 2, the database duplication operation requires Oracle RMAN. Depending on the size of the database, this process can be time consuming. Oracle Database 11*g* Release 2 incorporates Direct NFS Client (dNFS) with the duplication process instead of Oracle RMAN, and this simplifies and speeds up the duplication operation.

Direct NFS

Direct NFS Client (dNFS) is an Oracle Database 11*g* implementation of the NFS client that runs as part of the Oracle Database 11*g* engine. Through this integration, the Oracle Database engine optimizes the I/O access path to the NFS server to provide improved scalability and reliability. By tuning the protocol to match typical database I/O, dNFS provides faster performance than can be provided by the operating system's NFS driver. In addition, by minimizing context swap between user space and kernel space, dNFS further reduces CPU utilization. Data is cached only once in user space, and no second copy exist in kernel space, preserving valuable memory space. Performance is further improved by load balancing across multiple network interfaces from within dNFS, rather than within the OS layer.

Oracle introduced the dNFS CloneDB feature as part of Oracle Database 11*g* Release 2. The CloneDB feature makes it possible to instantly clone an existing backup of a database mounted over dNFS. The clone process uses copy-on-write technology; so only the changed datafile blocks need to be stored separately; the unmodified data is referenced directly from the backup files. DBAs don't have to reserve storage equivalent to the size of production database to set up a cloned environment. The cloned storage space usage grows at the speed at which the data is modified. This drastically increases the speed of cloning a database, providing an unlimited number of separate clones that can function against a single set of backup datafiles, with minimal performance impact on the production database. Since the CloneDB feature uses the backup piece as the backing storage, there is no impact on the I/O subsystem that is servicing the production database.

When it is time to destroy the auxiliary database environment, all the files in the clone environment can be deleted without any impact on the production or backup environment. The CloneDB feature saves considerable amounts of space and time when cloning a database, and it greatly simplifies and improves the process of creating and deleting clones of production databases. This process is explained in My Oracle Support Metalink Note 1210656.1: "Clone your dNFS Production Database for Testing."

Using dNFS in the database cloning process involves the following steps:

- Procure a new server and install the OS and all the relevant patches required for database installation.
- Install the Oracle Database software on the duplicate or auxiliary host.
- On the NFS server, create a directory as the copy-on-write location for the cloned instance.
- Export the directory as an NFS share.
- Take an image copy of the production database using Oracle RMAN. The backup should be placed in a location available to the server that will run the clone.
- Create a parameter file (PFILE) from the contents of the production SPFILE.
- Amend the contents of the PFILE to reflect the cloned database.
- Make sure the dNFS client is enabled for the Oracle home on the server that will run the clone.
- Create the auxiliary instance.
- Start the auxiliary instance.
- Mount the backup share.
- Set the following environment variables to the appropriate values for the new setup:
    - ORACLE_SID
    - MASTER_COPY_DIR; this is the directory that contains the backed-up datafiles
    - CLONE_FILE_CREATE_DEST; this is the destination directory for the cloned database datafiles
    - CLONEDB_NAME; this is the name of the cloned database

- Run the `clone.pl` script downloaded from My Oracle Support, naming the correct `init.ora` file to start the instance, and specifying the cloning scripts that will be created. The `clone.pl` Perl script creates a database script and a rename script. The database script (`crtdb.sql`) contains a generated CREATE CONTROLFILE command and a list of datafiles. The rename script (`dbren.sql`) creates the datafiles for the auxiliary instance and associates them to the NFS-mounted location of the original backed-up files. The `clone.pl` script is run as follows:

```
$ perl clone.pl init.ora crtdb.sql dbren.sql
```

- Start SQL*Plus as SYSDBA and run the scripts created by the `clone.pl` script.

```
$ sqlplus / as sysdba  @crtdb.sql @dbren.sql
```

Introducing the Oracle Featured Solution

The dNFS CloneDB feature described in the previous section replaces the last steps of the previous duplication process involving Oracle RMAN. However, there is still a need for IT personnel to procure hardware platforms, integrate networking and storage components, and install and patch the operating system and Oracle Database to be able to provision the database clone or clones. The solution featured in this paper for cloning an Oracle Solaris and Oracle Database environment aims to greatly simplify and automate the entire process.

The featured solution bypasses the need to provision and administer hardware, install operating systems, and configure networking and storage. All the auxiliary database instances, as well as the separate operating environments, are automatically provisioned and instantiated after executing a *single* command script, following an Oracle RMAN–based backup. The `cloneEnv.sh` command, shown in Appendix A, will automatically create and instantiate independent Oracle Solaris Zones, perform the database cloning, and start all the zones and database instances in only a few minutes. The `cloneEnv.sh` script requires only the names of the Oracle Solaris Zones to be provisioned and their respective IP addresses to execute successfully. After the `cloneEnv.sh` script finishes execution, all the new zones are all up and running, including the cloned databases running inside them. Similarly, when the cloned environments are no longer needed, all the zones and database instances can be deleted in seconds, following the execution of the `delAll.sh` script, without any impact on the production or backup environment or on other applications.

Contrast this with a typical scenario without the featured Oracle solution. Administrators must install and patch operating systems to support the necessary application infrastructure. A DBA is still needed to install and configure the Oracle database on the auxiliary server, create and configure the necessary Oracle Database files, and configure networking and storage. This process is time consuming, costly, and error-prone. Beyond the costs of acquiring hardware, software, storage, and networking, there are ongoing maintenance and support costs that can proliferate as servers and storage components multiply and the complexity of the landscape increases.

The featured solution for cloning an Oracle Solaris and Oracle Database environment provides the following technical advantages:

- **Simplified database cloning.** The cloning features of the Sun ZFS Storage Appliance are used to provide the copy-on-write semantics needed by the cloning operation, which makes redundant the use of the rename script described in the previous “Direct NFS” section. This improves reliability and predictability in deployment, streamlines cloning management, and simplifies the cloning operation, leading to a faster execution.
- **Server density.** Built-in virtualization and 256 threads in just five rack units (5RU) with Oracle’s four-socket SPARC T4-4 server make the solution ideal for deploying a large number of databases. Oracle’s SPARC T5-4 server doubles the number of threads to 512 using four sockets, while occupying the same space. Oracle’s SPARC T5-8 server features eight processors running 1024 threads, and occupying 8RU of space.
- **Ability to replicate the entire stack with a golden image.** The use of Oracle Solaris Zones to host the database enables the creation a golden image that can be replicated as needed. The entire duplication process can be executed with a single script that includes creating, configuring, and provisioning independent Oracle Solaris Zones, as well as duplicating the Oracle database. The script takes only a few minutes to run, and can provision new servers up to 50 times faster than traditional methods, after an initial one-time configuration setup. This time is mainly taken by provisioning the Oracle Solaris 11 operating system; the database duplication is instantaneous since it is based on the copy-on-write technology. Similarly, reclaiming all the resources when the cloned environments are no longer needed is just as easy, clean, and fast with the execution of the `delAll.sh` script, without impacting other applications.
- **Ability to perform regression testing on a cloned database copy**. Administrators can quickly and efficiently preview the impact of patches and upgrades on a cloned copy of their production systems before going live. Once satisfied with their tests, they also have the option of making the cloned database the new production system. These capabilities help eliminate risks and provide crucial reliability and predictability for the production IT environment.

This solution can also be used with other Oracle applications that require databases as part of the installation, such as Oracle E-Business Suite and Oracle's PeopleSoft, Siebel, and JD Edwards applications. IT administrators can now clone an environment in minutes and use the newly created Oracle Solaris Zones to install Oracle business applications, further simplifying and speeding up the installation process, reducing risk, and enabling faster deployment of these business applications.

Solution Components

The solution described in this document supports the operating system and database versions described in Table 1. Further detail about the hardware components of this solution can be found in Appendix B.

TABLE 1: SUPPORTED VERSIONS

Operating System	Oracle Solaris 11
Oracle Database	Oracle Database 11*g* Release 2
Server	SPARC T4 or SPARC T5
Storage	Sun ZFS Storage 7320

Oracle Solaris

Oracle Solaris is an industry-leading operating system designed to handle enterprise, business-critical operations. Oracle Solaris provides key functionality for virtualization, optimal utilization, high availability, unparalleled security, and extreme performance for both vertically and horizontally scaled environments. Oracle Solaris runs on a broad range of SPARC (and x86-based) systems and compatibility with existing applications is guaranteed.

Oracle Solaris features the ZFS file system for superior data integrity, advanced security protection and management, and scalable performance due to efficient thread scheduling on multicore processors. Oracle Solaris also provides built-in virtualization with minimal overhead (Oracle Solaris Zones), which isolates applications and optimizes system resource allocations. Innovations such as DTrace, Predictive Self Healing, and the Service Management Facility (SMF) have made Oracle Solaris the operating system of choice for applications that demand business-critical performance and availability.

Oracle Solaris 11 features scalability enhancements, enhanced kernel data structures, and library optimizations to support large-scale database workloads.

Oracle Solaris Zones

An Oracle Solaris Zone is a virtual instance of the Oracle Solaris OS that provides an isolated and secure environment for running applications. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Oracle Solaris Zones are very flexible; with Oracle Solaris resource management, it is easy to move individual CPUs between zones as needed or configure a more sophisticated way to share CPUs and memory to handle additional workloads as business conditions dictate. The original operating environment, before any zones are created, is called the *global zone* to distinguish it from non-global zones. The global zone holds the Oracle Solaris kernel, the device drivers and devices, the memory management system, the file system and, in many cases, the network stack. With Oracle Solaris Zones, you can maintain the one-application-per-server deployment model while simultaneously sharing hardware resources. A SPARC T4 or T5 server support the creation of many non-global zones.

The global zone sees all physical resources and provides common access to these resources to non-global zones. Looking from the global zone, a non-global zone is just a bunch of processes grouped together by a tag called a zone ID. The non-global zones appear to applications as separate Oracle Solaris installations.

Non-global zones have their own file systems, process namespace, security boundaries, and network addresses. They can also have their own network stack with separate network properties. Virtual networks can be created between zones to help isolate data movement and prevent access to external networks. Each non-global zone has an administrative root login; however, a privileged root user in a non-global zone cannot break into a neighboring non-global zone.

Oracle Solaris Network Virtualization

Oracle Solaris 11 added network virtualization features enhancements including virtual NICs (VNICs), virtual switching, network resource management, and an efficient and easy way to manage integrated Quality of Service (QoS) to enforce bandwidth limits on VNICs and traffic flows.

Oracle Solaris 11 introduced the ability to virtualize a physical NIC into multiple VNICs, which can be assigned to Oracle Solaris Zones sharing the physical NIC. This virtualization is implemented by the MAC layer and the VNIC pseudo driver of the Oracle Solaris network stack. VNICs appear to the operating system as physical NICs. Each VNIC is assigned its own MAC address and optional VLAN (VID). The MAC address and VID are used to identify a VNIC on the network.

VNICs can be created not only on physical NICs but also on virtual switches through Ethernet stubs (etherstubs) and on link aggregations. Etherstubs allow the creation of virtual networks that are completely independent from hardware NICs, enabling the construction of virtual network topologies within a single Oracle Solaris instance. The virtual switching is consistent with the behavior of a typical physical switch found on a physical network. Creating VNICs on top of link aggregations allows VNICs to benefit from high availability and higher throughput transparently to the zones that use these VNICs.

Oracle Solaris 11 network virtualization allows a bandwidth limit to be set on a VNIC to ensure that each VNIC will have a minimum bandwidth available, regardless of the bandwidth usage of other zones sharing the same physical NIC. This mechanism allows the administrator to configure the link speed of VNICs that are assigned to zones. This capability is most useful in ensuring that one interface does not exceed its expected use of the network and negatively impact other traffic.

Deployment Example

An example deployment of the Oracle solution for cloning an Oracle Solaris and Oracle Database environment is shown in Figure 1. The remainder of this section describes how to go about provisioning this environment.

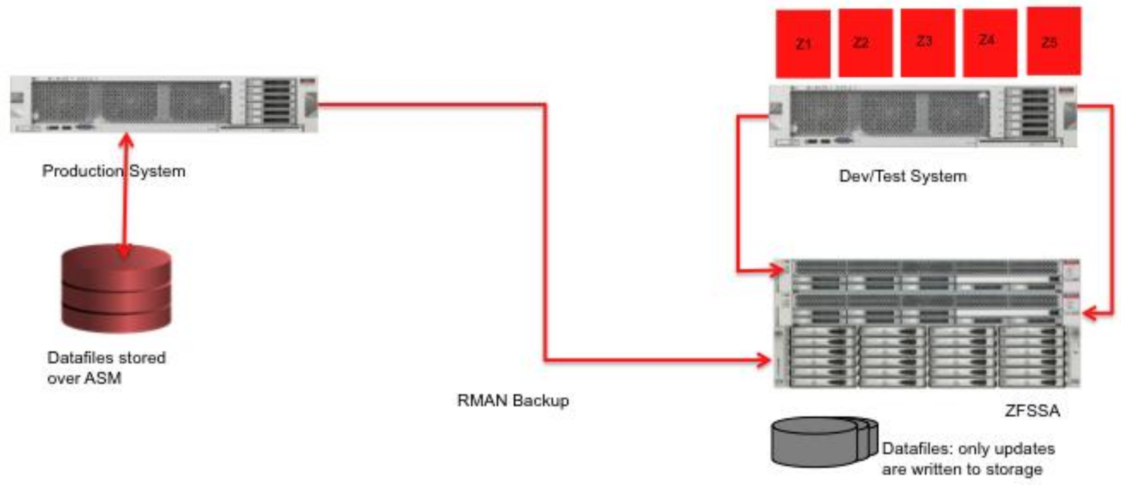

Z1
Z2
Z3
Z4
Z5
Production System
Dev/Test System
Datafiles stored over ASM
RMAN Backup
ZFSSA
Datafiles: only updates are written to storage

Figure 1: Oracle solution for cloning an Oracle Solaris and Oracle Database environment.

Figure 1 shows the production system running Oracle Database 11*g* Release 2 on a SPARC T4-4 server with the database files stored in a Fibre Channel storage array using Oracle Automatic Storage Management. The database is backed up using Oracle RMAN to the Sun ZFS Storage Appliance. The Dev/Test system is running on a SPARC T4-4 server that is connected via Ethernet using the NFS protocol to the Sun ZFS Storage Appliance.

Z1 in the Dev/Test system is an Oracle Solaris Zone that is set up with the same operating system, applications, and patch level set as the production environment. Z1 runs the same version of Oracle Database as the production system, using datafiles recovered via Oracle RMAN. Z2 through Z5 are cloned Oracle Solaris Zones running cloned databases created after the execution of the `cloneEnv.sh` script. These zones are created and deleted on demand, as business requires, without impacting other zones or databases.

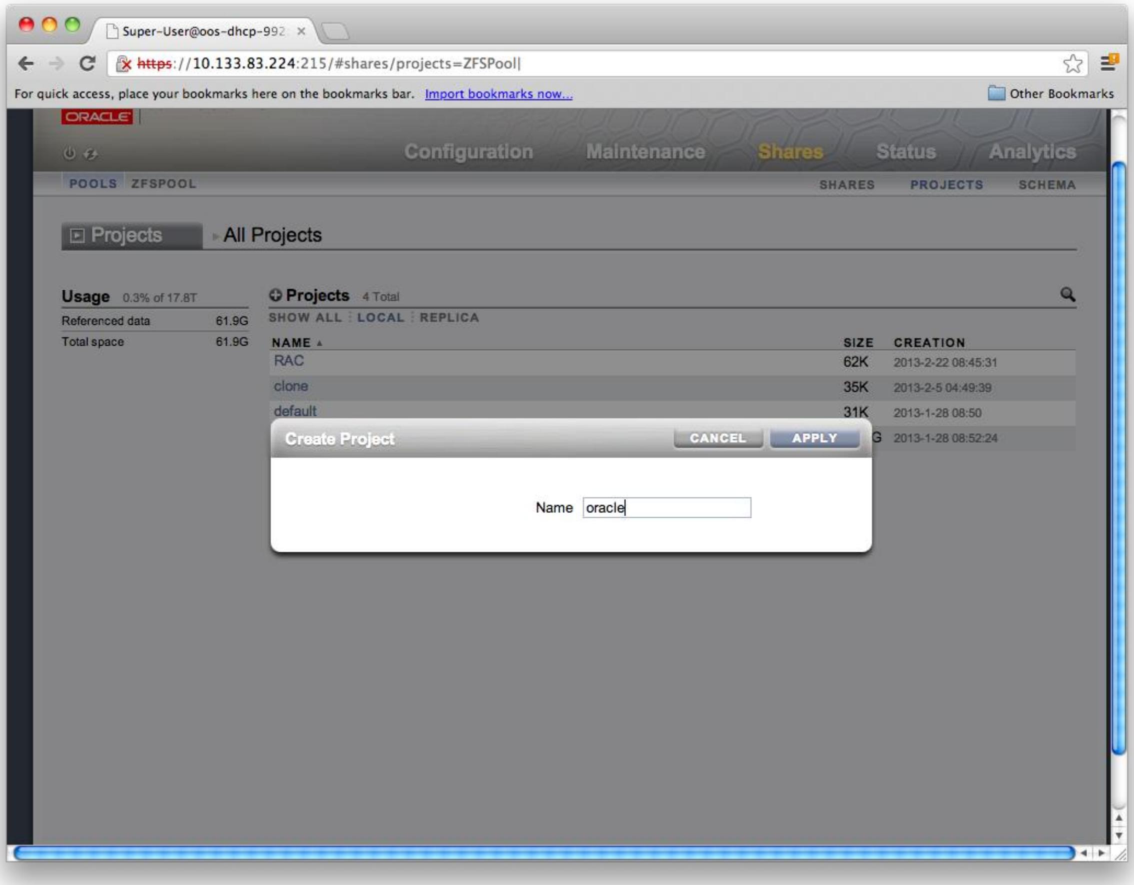
Figure 1 also shows the datafiles consuming a minimum amount of space because only new updates are written to storage. All the cloned zones and databases share the same original backup files.

To implement the solution shown in Figure 1, the following initialization steps need to be implemented one time only:

1. On the Sun ZFS Storage Appliance, create two file systems that will be mounted from the production system and will hold the backup of the production database and other configuration files.
2. Using Oracle RMAN, back up the production database to the Sun ZFS Storage Appliance.
3. On the Sun ZFS Storage Appliance, create a snapshot and clones of the file system that has the backup of production data. Repeat this step for every new backup or rerun of the cloning solution.
4. On the Dev/Test system, create a non-global Oracle Solaris Zone Z1 as a master zone that is identical to the production system, which will be cloned.

Create a File System for Database Backup

Create project `oracle` on the Sun ZFS Storage Appliance, by selecting **PROJECTS** and clicking the + sign. Then enter the name `oracle` and click **APPLY**, as shown in Figure 2.



Usage 0.3% of 17.8T

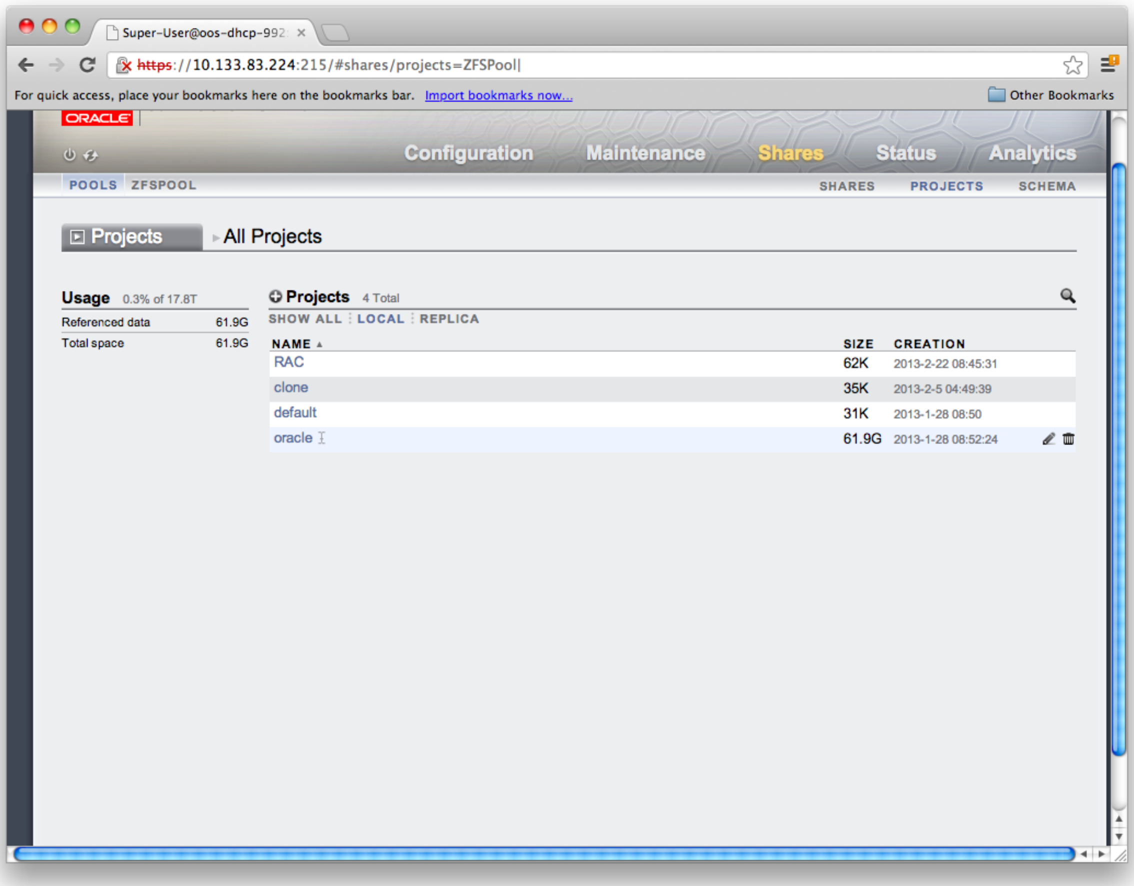
Referenced data	61.9G
Total space	61.9G

NAME	SIZE	CREATION
RAC	62K	2013-2-22 08:45:31
clone	35K	2013-2-5 04:49:39
default	31K	2013-1-28 08:50
	G	2013-1-28 08:52:24

Name: oracle

Figure 2. Creating project `oracle` on the Sun ZFS Storage Appliance.

After the project `oracle` is created, hover the mouse on the right side of the `oracle` row, and click the pencil (see Figure 3) to create a file system.



Usage 0.3% of 17.8T

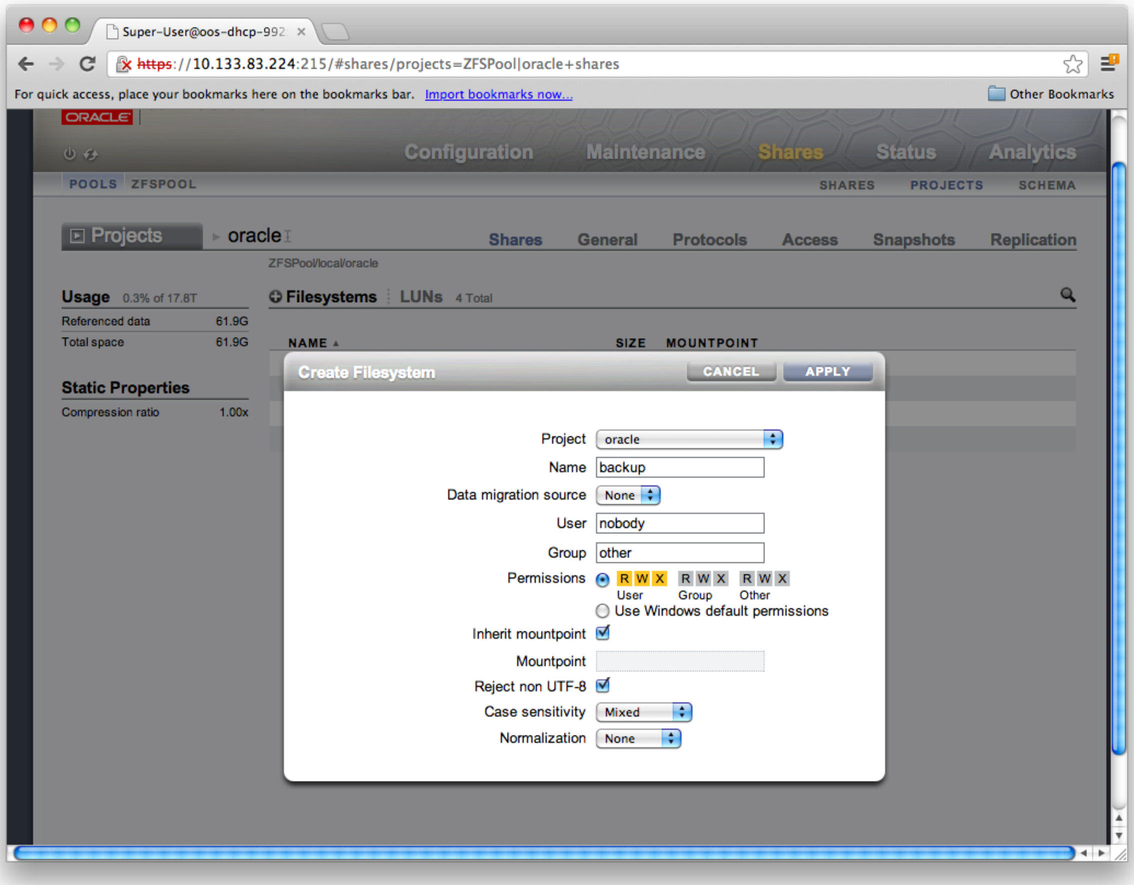
Referenced data	61.9G
Total space	61.9G

Projects 4 Total

NAME	SIZE	CREATION
RAC	62K	2013-2-22 08:45:31
clone	35K	2013-2-5 04:49:39
default	31K	2013-1-28 08:50
oracle	61.9G	2013-1-28 08:52:24

Figure 3. Clicking the pencil icon opens a dialog box for creating a file system.

Click the + sign next to **Filesystems** to create a file system called `backup` in project `oracle` on the Sun ZFS Storage Appliance, as shown in Figure 4.



Usage 0.3% of 17.8T

Referenced data	61.9G
Total space	61.9G

Static Properties

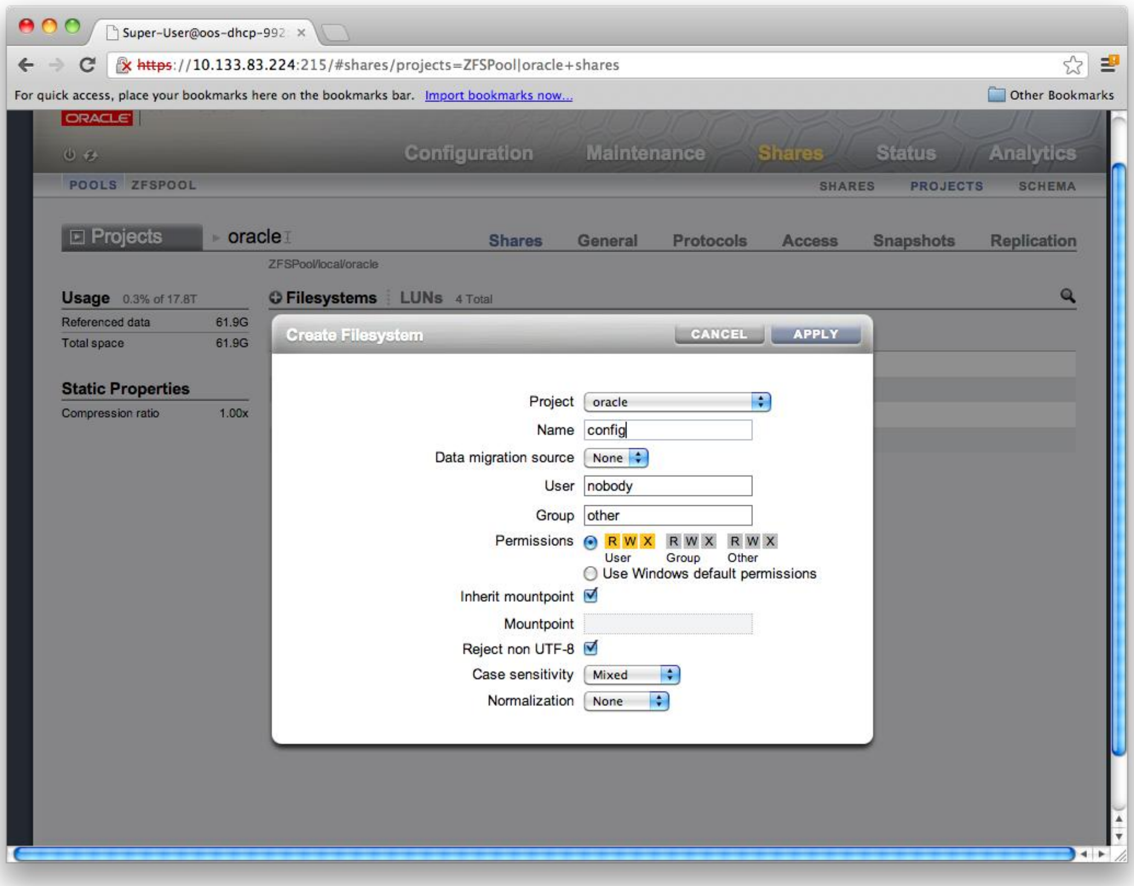
Compression ratio	1.00x

Create Filesystem

Project	oracle
Name	backup
Data migration source	None
User	nobody
Group	other
Permissions	[✓ R W X User] R W X Group R W X Other; Use Windows default permissions
Inherit mountpoint	✓
Mountpoint	
Reject non UTF-8	✓
Case sensitivity	Mixed
Normalization	None

Figure 4. Creating a `backup` file system.

Repeat the above steps to create another file system called `config` on the Sun ZFS Storage Appliance, as shown in Figure 5.



Usage 0.3% of 17.8T

Referenced data	61.9G
Total space	61.9G

Static Properties

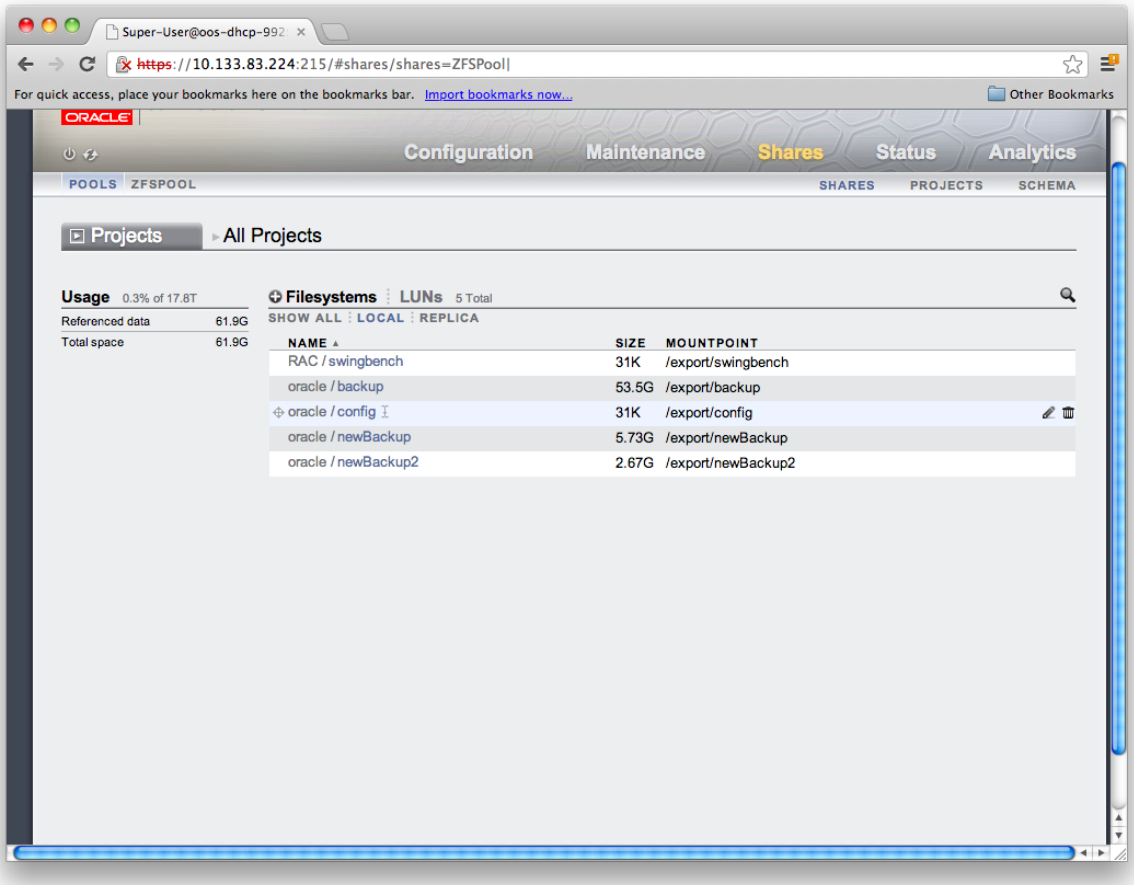
Compression ratio	1.00x

Create Filesystem

Project	oracle
Name	config
Data migration source	None
User	nobody
Group	other
Permissions	[✓ R W X User] R W X Group R W X Other
	Use Windows default permissions
Inherit mountpoint	✓
Mountpoint	
Reject non UTF-8	✓
Case sensitivity	Mixed
Normalization	None

Figure 5. Creating a `config` file system.

Clicking the **SHARES** submenu displays the newly created file systems, as shown in Figure 6.



Usage	0.3% of 17.8T
Referenced data	61.9G
Total space	61.9G

NAME	SIZE	MOUNTPOINT
RAC / swingbench	31K	/export/swingbench
oracle / backup	53.5G	/export/backup
oracle / config	31K	/export/config
oracle / newBackup	5.73G	/export/newBackup
oracle / newBackup2	2.67G	/export/newBackup2

Figure 6. The new file systems are listed.

The newly created file systems `/export/backup` and `/export/config` are to be mounted on the production system as described in the following section. The file system `/export/backup` will hold the production database backup, and the `/export/config` file system will hold configuration files to be shared on the Sun ZFS Storage Appliance.

Back Up the Production Database on the Sun ZFS Storage Appliance

Mount the newly created file systems `/export/backup` and `/export/config` on the production system using the following command, where `10.133.83.224` is the IP address of the Sun ZFS Storage Appliance, and `/backup` and `/config` are the mount points created on the production system:

```
root@production:~# mount -F nfs  -o
rw,bg,hard,nointr,rsize=32768,wsize=32768,proto=tcp,noac,forcedirectio,vers=3,suid
10.133.83.224:/export/backup /backup/

root@production:~# mount -F nfs  -o
rw,bg,hard,nointr,rsize=32768,wsize=32768,proto=tcp,noac,forcedirectio,vers=3,suid
10.133.83.224:/export/config /config/
```

To have these file systems mounted automatically after a reboot, update the file `/etc/vfstab` as follows:

```
#device         device          mount           FS      fsck    mount   mount

#to mount       to fsck         point           type    pass    at boot options

/devices        -               /devices        devfs   -       no      -

/proc           -               /proc           proc    -       no      -

ctfs            -               /system/contract ctfs   -       no      -

objfs           -               /system/object  objfs   -       no      -

sharefs         -               /etc/dfs/sharetab       sharefs -       no      -

fd              -               /dev/fd         fd      -       no      -

swap            -               /tmp            tmpfs   -       yes     -

/dev/zvol/dsk/rpool/swap        -               -               swap    -       no      -

/dev/zvol/dsk/rpool/swap2       -               -               swap    -       no      -

10.133.83.224:/export/backup    -       /backup         nfs     -       yes
rw,bg,hard,nointr,rsize=32768,wsize=32768,proto=tcp,noac,forcedirectio,vers=3,suid

10.133.83.224:/export/config    -       /config         nfs     -       yes
rw,bg,hard,nointr,rsize=32768,wsize=32768,proto=tcp,noac,forcedirectio,vers=3,suid
```

Log in to the Oracle database, and create an `init.ora` parameter file from SPFILE, and place it in the `/config` directory:

```
-bash-4.1$ sqlplus / as SYSDBA

SQL> create pfile='/config/inittpcc.ora' from spfile;
File created.
SQL> exit
```

In this example the Oracle database is called `tpcc`. The full path of the created parameter file will be included with the `oraLocation` variable in the `setup.sh` file listed in Appendix A.

Using Oracle RMAN, back up the production system database to the `backup` share on the Sun ZFS Storage Appliance as follows:

```
RMAN> backup as copy database format "/backup/back_%U";
```

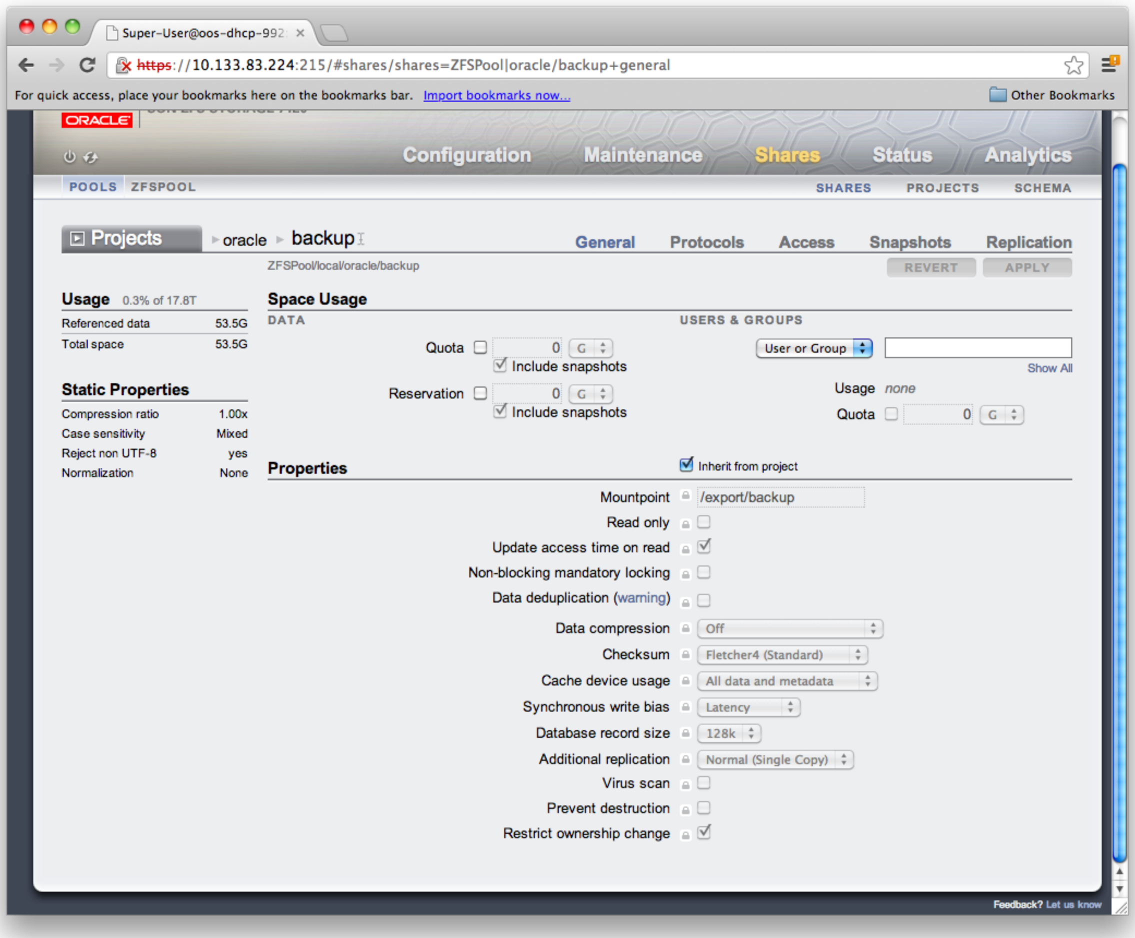
The `/backup` directory will be included with the `masterBackupLocation` variable in the `setup.sh` file listed in Appendix A.

Create a Clone of the Database Backup File System

To create a clone of the database backup file system `oracle/backup`, first create a snapshot of the file system that has the backup of production data on the Sun ZFS Storage Appliance, and then promote the snapshot to a clone.

Create a Snapshot of the Database Backup File System

Click **SHARES** in the submenu, and then click the pencil icon on the `oracle/backup` row to edit this file system.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Space Usage

DATA

Quota ☐ 0 G ☑ Include snapshots

Reservation ☐ 0 G ☑ Include snapshots

USERS & GROUPS

User or Group Show All

Usage none

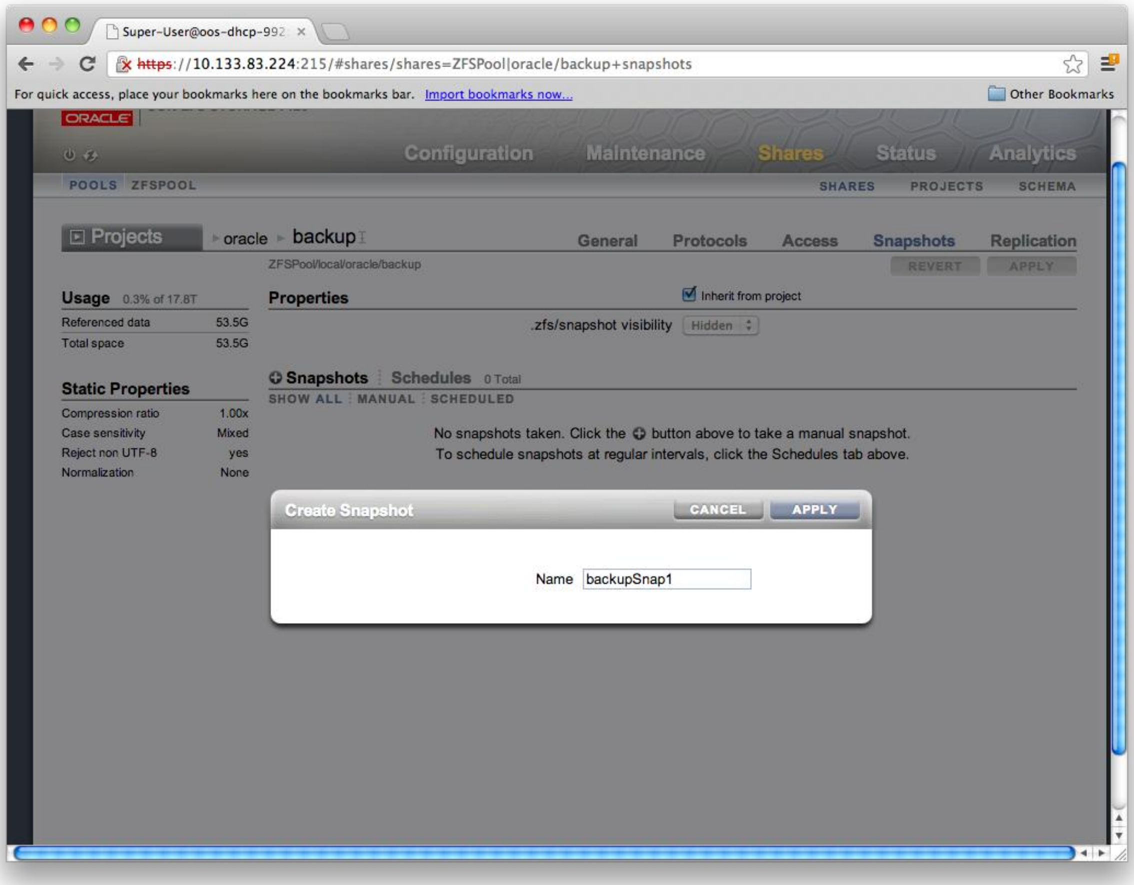
Quota ☐ 0 G

Properties ☑ Inherit from project

Mountpoint	/export/backup
Read only	☐
Update access time on read	☑
Non-blocking mandatory locking	☐
Data deduplication (warning)	☐
Data compression	Off
Checksum	Fletcher4 (Standard)
Cache device usage	All data and metadata
Synchronous write bias	Latency
Database record size	128k
Additional replication	Normal (Single Copy)
Virus scan	☐
Prevent destruction	☐
Restrict ownership change	☑

Figure 7. Editing the file system.

Click **Snapshots**, and then click the + sign next to **Snapshots** to create a snapshot called `backupSnap1`. Then click **APPLY**.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Properties ✓ Inherit from project

.zfs/snapshot visibility: Hidden

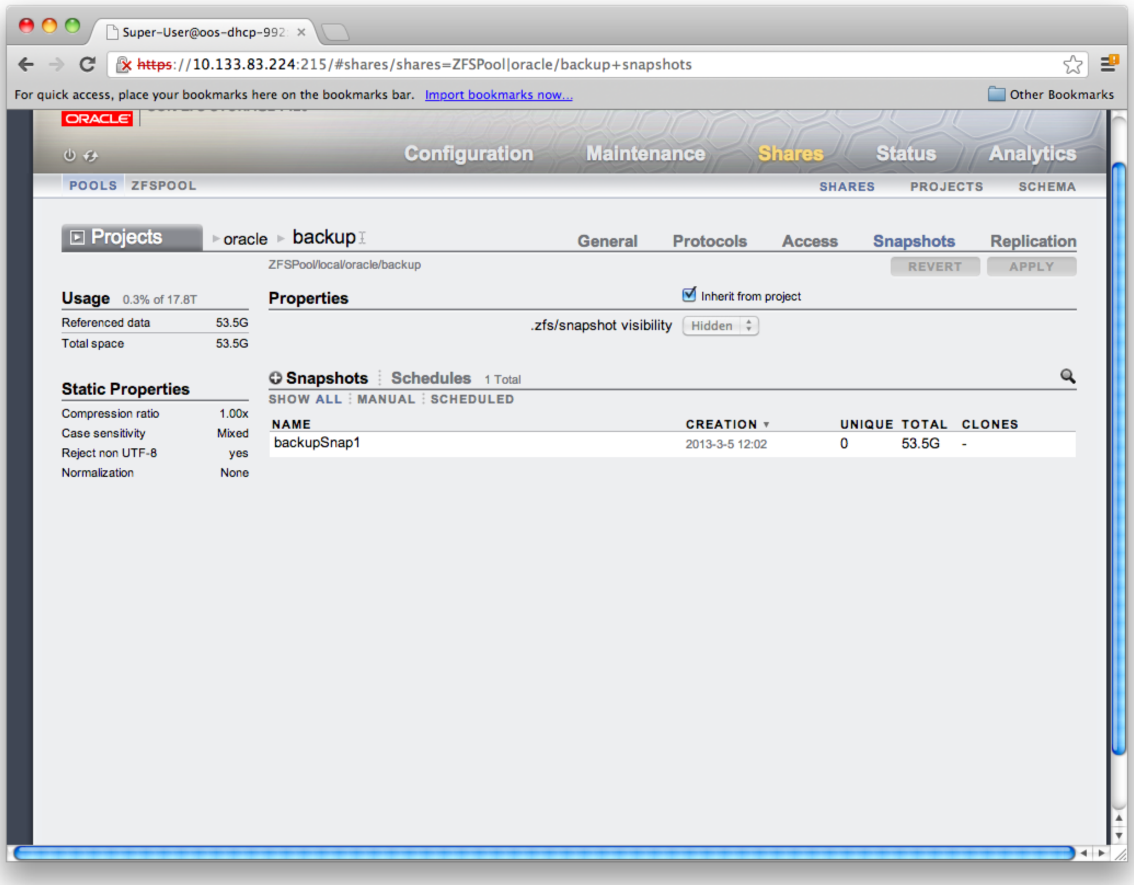
No snapshots taken. Click the ⊕ button above to take a manual snapshot.
To schedule snapshots at regular intervals, click the Schedules tab above.

Create Snapshot

Name: backupSnap1

Figure 8. Creating a snapshot.

Figure 9 shows the created snapshot.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Properties ✓ Inherit from project

.zfs/snapshot visibility: Hidden

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Snapshots 1 Total

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	-

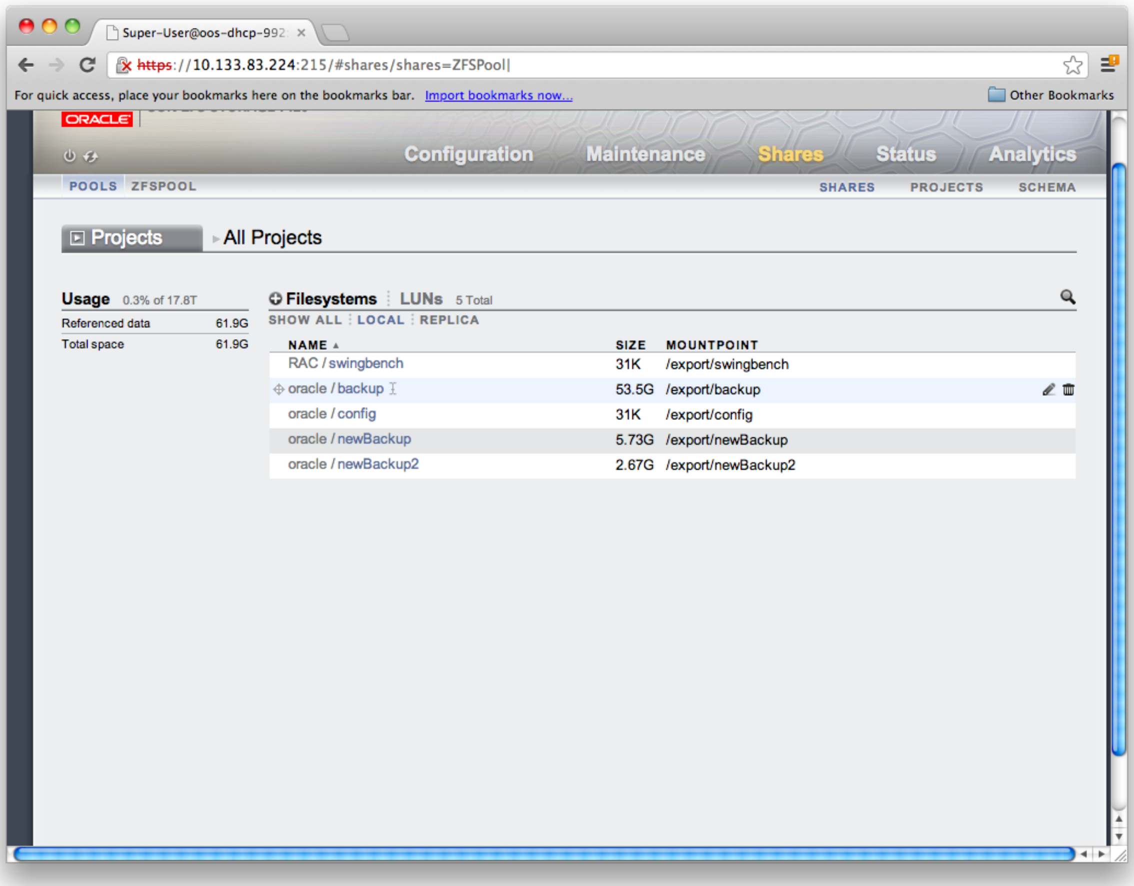
Figure 9. The new snapshot is listed.

Create Clones of the Database Backup File System

The goal of this section is to create clones of the file system that has the backup of production data on the Sun ZFS Storage Appliance.

Create a project called `clone` by following the same steps used to create the `oracle` project previously. This project will include all the clones that will be created.

To promote the snapshot `backupSnap1` to a clone, select **SHARES** from the submenu, and then click the pencil icon as if to edit the `oracle/backup` file system.



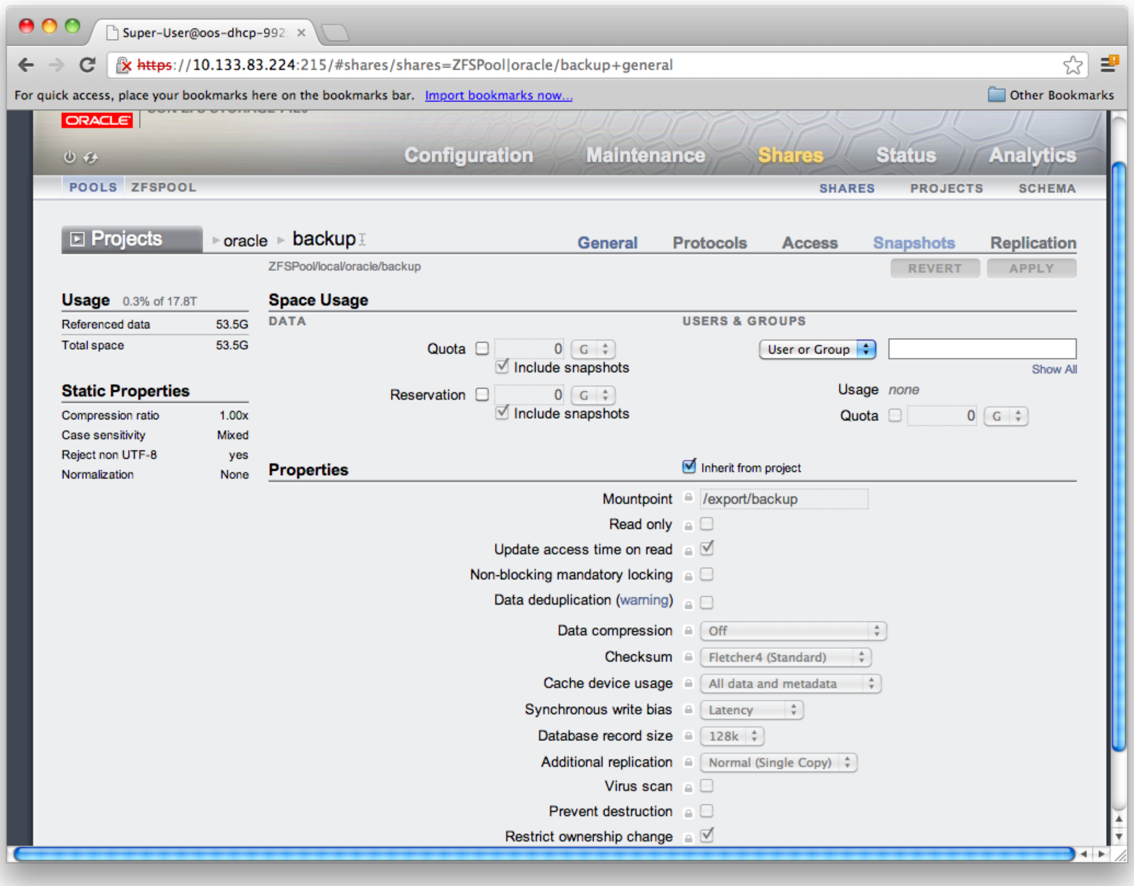
Usage 0.3% of 17.8T

Referenced data	61.9G
Total space	61.9G

NAME	SIZE	MOUNTPOINT
RAC / swingbench	31K	/export/swingbench
oracle / backup	53.5G	/export/backup
oracle / config	31K	/export/config
oracle / newBackup	5.73G	/export/newBackup
oracle / newBackup2	2.67G	/export/newBackup2

Figure 10. Clicking the pencil icon opens a screen from which the snapshot can be promoted to a clone.

Click **Snapshots**.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Space Usage

DATA

Quota ☐ 0 G
☑ Include snapshots

Reservation ☐ 0 G
☑ Include snapshots

USERS & GROUPS

User or Group

Usage *none*

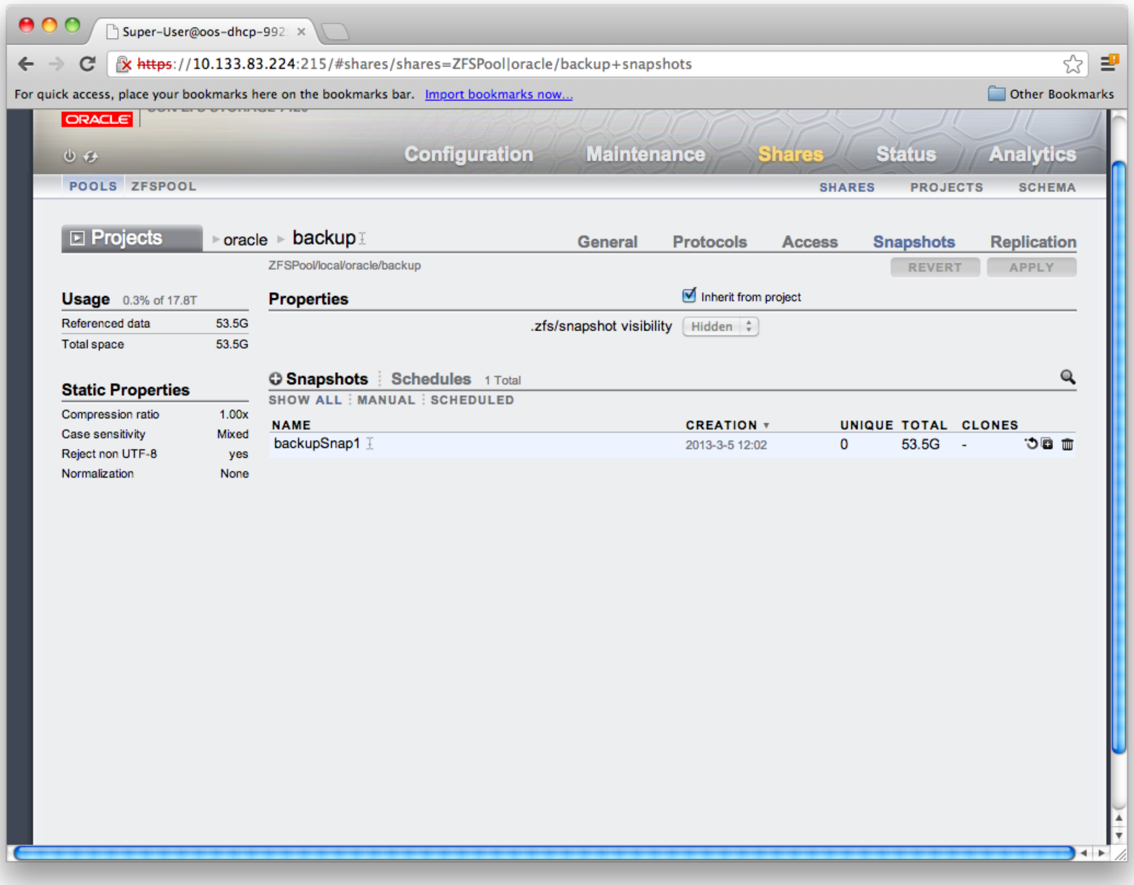
Quota ☐ 0 G

Properties ☑ Inherit from project

Mountpoint	/export/backup
Read only	☐
Update access time on read	☑
Non-blocking mandatory locking	☐
Data deduplication (warning)	☐
Data compression	Off
Checksum	Fletcher4 (Standard)
Cache device usage	All data and metadata
Synchronous write bias	Latency
Database record size	128k
Additional replication	Normal (Single Copy)
Virus scan	☐
Prevent destruction	☐
Restrict ownership change	☑

Figure 11. Clicking **Snapshots** opens a screen that lists all the snapshots that were taken.

In the `backupSnap1` snapshot, click the + sign to select the **Clone snapshot as a new share** option.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

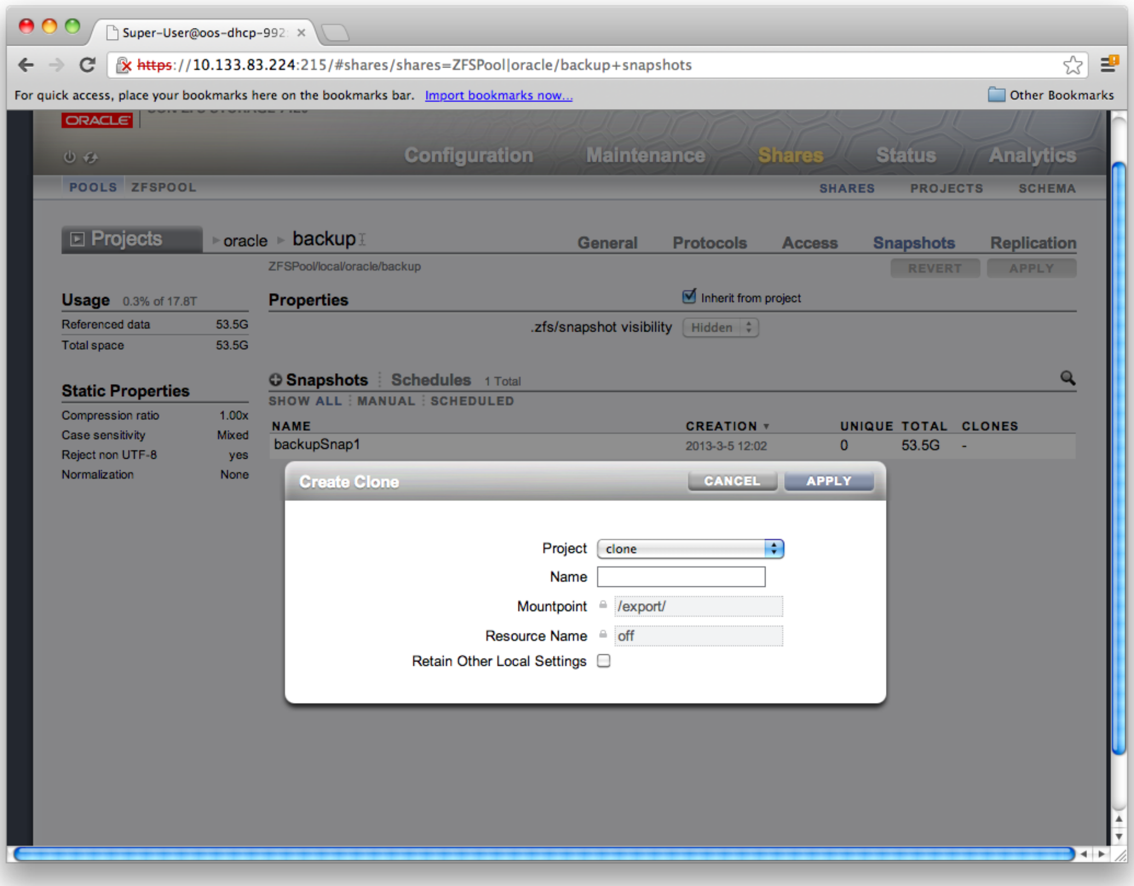
Properties [✓ Inherit from project]

.zfs/snapshot visibility: Hidden

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	-

Figure 12. Screen where you can clone the `backupSnap1` snapshot.

Select the project `clone` from the drop-down menu in the Create Clone dialog box.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Properties

Inherit from project ✓

.zfs/snapshot visibility: Hidden

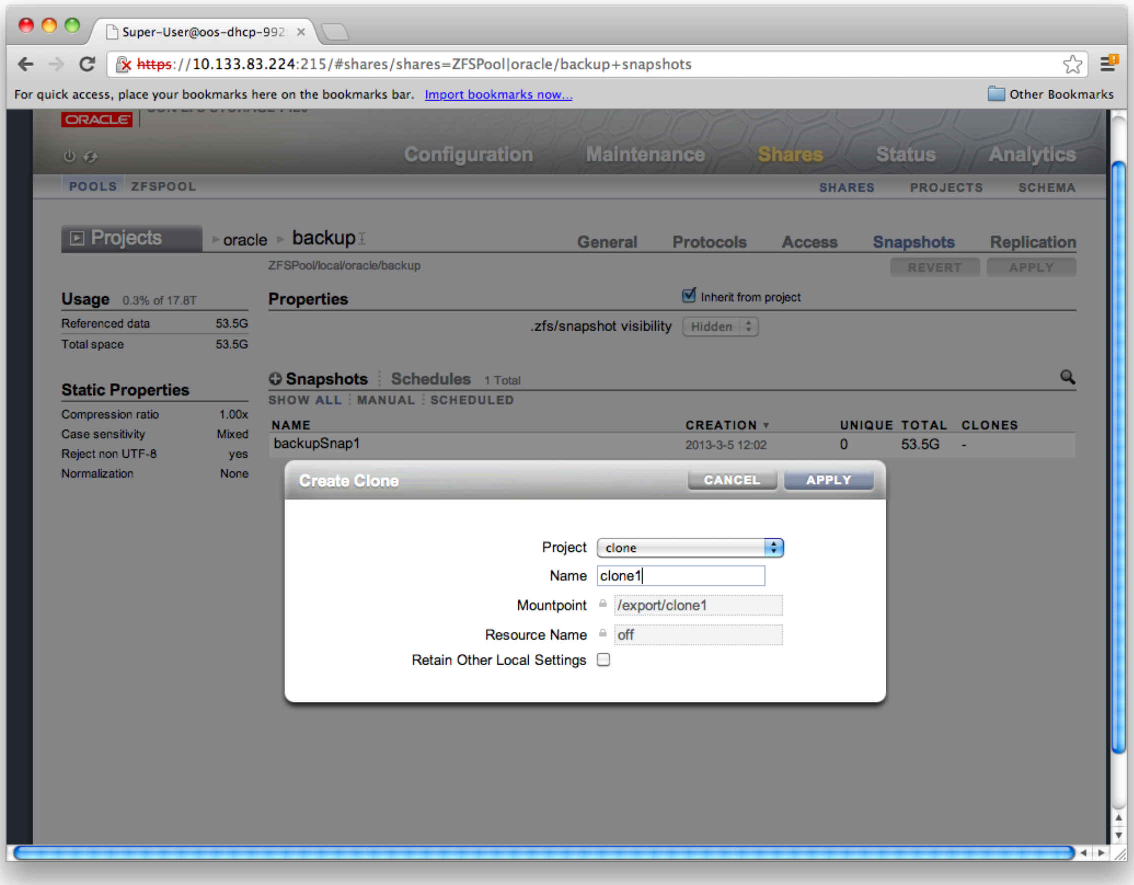
Snapshots 1 Total

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	-

Project	clone
Name	
Mountpoint	/export/
Resource Name	off
Retain Other Local Settings	☐

Figure 13. Screen for creating a clone.

Enter the name of the clone, and click **APPLY**. This name will be included with the `cloneZones` variable in the `setup.sh` file listed in Appendix A.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Properties

[✓] Inherit from project

.zfs/snapshot visibility: Hidden

Snapshots Schedules 1 Total

SHOW ALL MANUAL SCHEDULED

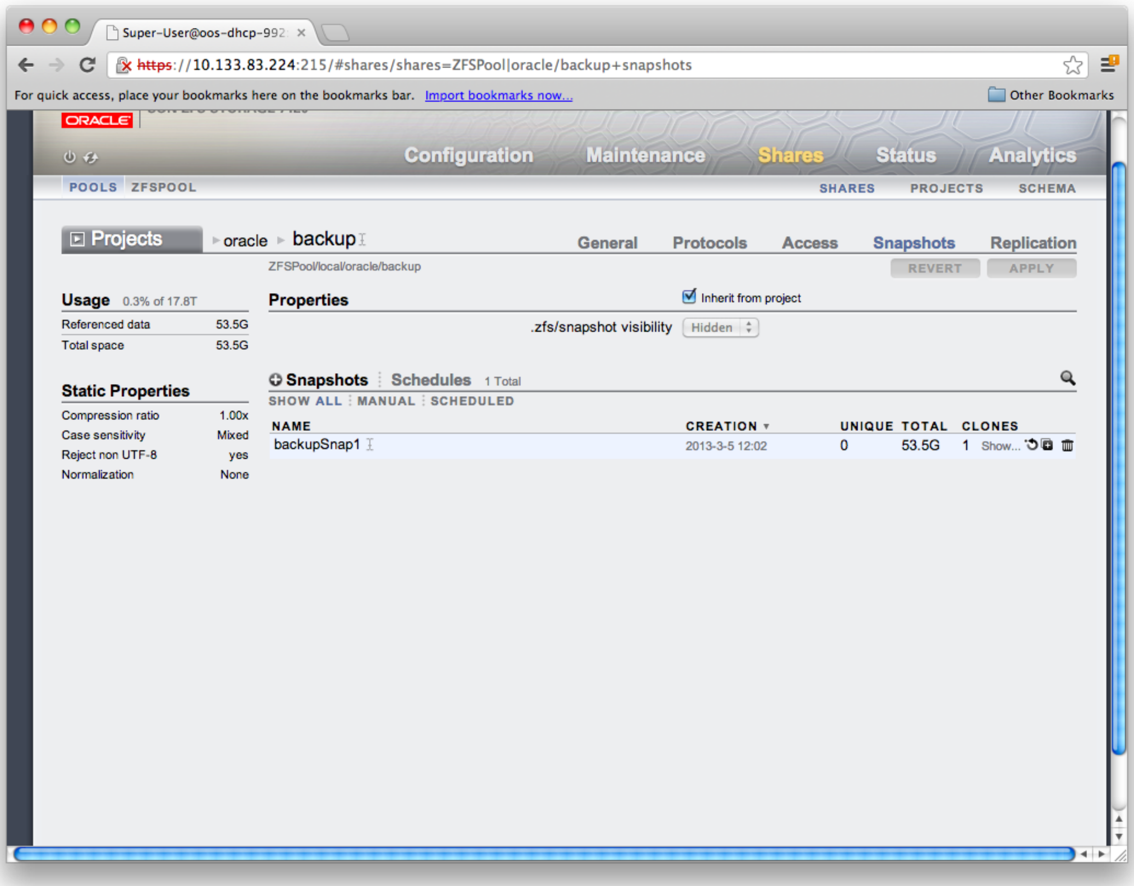
NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	-

Create Clone CANCEL APPLY

Project	clone
Name	clone1
Mountpoint	/export/clone1
Resource Name	off
Retain Other Local Settings	☐

Figure 14. Entering a name for the clone.

Click **Show** in the `backupSnap1` snapshot to see the clone that was just created.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Properties ✓ Inherit from project

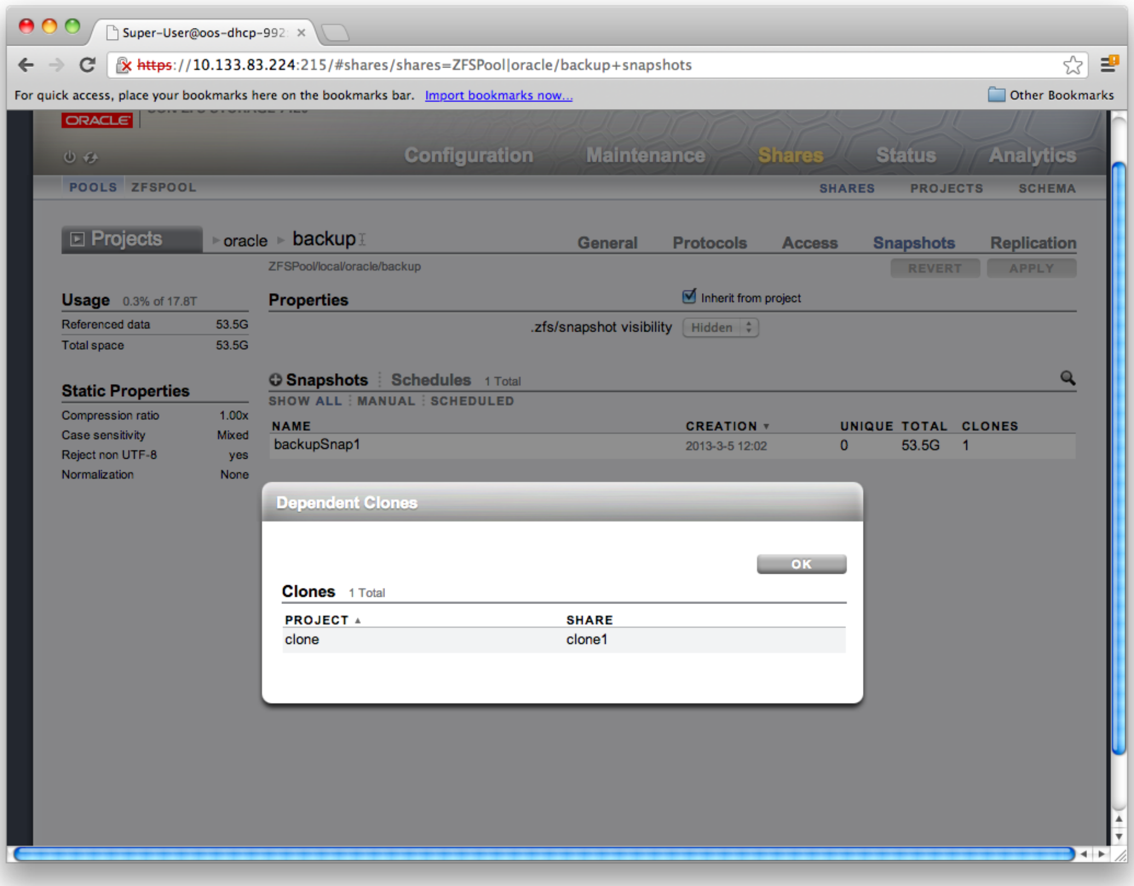
.zfs/snapshot visibility: Hidden

Snapshots 1 Total

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	1 Show...

Figure 15. A clone was created of the `backupSnap1` snapshot.

Figure 16 shows that a clone called `clone1` has been created in project `clone`.



ZFSPool/local/oracle/backup

Usage 0.3% of 17.8T

Referenced data	53.5G
Total space	53.5G

Properties

Inherit from project ✓

.zfs/snapshot visibility Hidden

Static Properties

Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Snapshots 1 Total

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	1

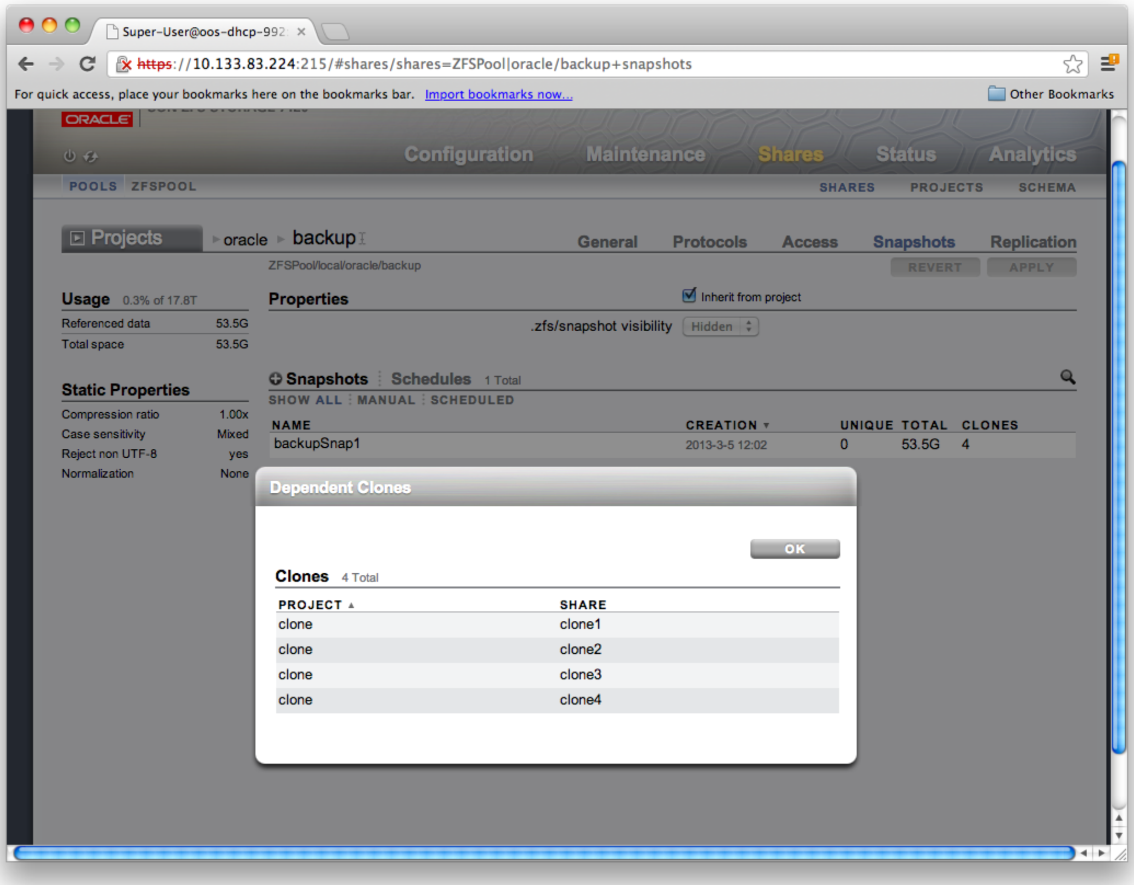
Dependent Clones

Clones 1 Total

PROJECT	SHARE
clone	clone1

Figure 16. The clone that was just created is listed.

Repeat the previous steps to create as many clones of `backupSnap1`, as needed. The names of the clones will be listed in the `cloneZones` entry in the `setup.sh` file listed in Appendix A. Figure 17 shows that four clones—named `clone1`, `clone2`, `clone3`, and `clone4`—were created in the project `clone`.



https://10.133.83.224:215/#shares/shares=ZFSPool|oracle/backup+snapshots

Projects ▸ oracle ▸ backup

ZFSPool/local/oracle/backup

Usage	0.3% of 17.8T
Referenced data	53.5G
Total space	53.5G

Static Properties	
Compression ratio	1.00x
Case sensitivity	Mixed
Reject non UTF-8	yes
Normalization	None

Properties

☑ Inherit from project

.zfs/snapshot visibility: Hidden

Snapshots | Schedules 1 Total

NAME	CREATION	UNIQUE	TOTAL	CLONES
backupSnap1	2013-3-5 12:02	0	53.5G	4

Dependent Clones

Clones 4 Total

PROJECT	SHARE
clone	clone1
clone	clone2
clone	clone3
clone	clone4

Figure 17. The four created clones are listed.

Create an Oracle Solaris Master Zone

As `root`, first create a VNIC for the non-global master zone and all the clones, as follows, where `net0` is the link identified as being up after running the `dladm show-link` command, which lists all the physical interfaces that are configured and up on the system:

```
# dladm create-vnic -l net0 public1

# dladm create-vnic -l net0 clone1

# dladm create-vnic -l net0 clone2

# dladm create-vnic -l net0 clone3

# dladm create-vnic -l net0 clone4
```

Then create a template for the master zone named `master.cfg` using the following information, where `zonepath` points to zone pool created using the `zpool` command and `public1` is the name of the VNIC created previously:

```
create –b

set zonepath=/rpool/zones/master

set brand=solaris

set autoboot=false

set limitpriv=default,proc_priocntl

set ip-type=exclusive

add net

        set configure-allowed-address=true

        set physical=public1

end

add rctl

        set name=zone.max-shm-memory

        add value (priv=privileged,limit=53687091200,action=deny)

end
```

Create the master zone by issuing the following command:

```
# zonecfg -z master -f master.cfg
```

To restrict the number of CPUs in the zone for licensing purposes, use the `dedicated-cpu` command to specify the number of virtual CPUs. The following snippet sets `dedicated-cpu` to `2`.

```
# zonecfg -z master

# zonecfg:master> add dedicated-cpu

# zonecfg:master:dedicated-cpu> set ncpus=2

# zonecfg:master:dedicated-cpu> end

# zonecfg:master:dedicated-cpu> commit

# zonecfg:master:dedicated-cpu> exit
```

Then issue the following commands to install and boot the zone:

```
# zoneadm -z master install

# zoneadm -z master boot
```

After logging on to the master zone, create the system configuration file for the master zone. The `sysconfig create-profile` command will invoke the System Configuration Interactive (SCI) tool, which allows administrators to configure parameters, such as the system's host name, time zone, root and user accounts, networking, and name services. These configuration parameters are saved in the `master.xml` file. This file is critical, as it will be used for cloning zones later on. This will make the process of creating zones automatic and saves time during deployment.

```
# zlogin master

master: # sysconfig create-profile -o ./master.xml
```

After the profile is successfully created, configure the system to use the created profile, and then reboot the system:

```
master: # sysconfig configure -c ./master.xml

master: # reboot
```

For more details on these commands, please check the *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* guide and the complete Oracle Solaris 11 documentation library.

Once the master zone is booted, proceed with the Oracle Database installation as well as the installation of any other application that is used in the production system. Set this zone exactly the way you need the other zones to be configured.

The name of the master zone is listed as the `masterZone` variable in the `setup.sh` file listed in Appendix A. Copy the `master.xml` file into the `/config` mounted directory from the Sun ZFS Storage Appliance .The full path of the `master.xml` system configuration file is listed as the `configFile` variable in the same file.

At this point, four clones of the backup of the production Oracle database have been created on the Sun ZFS Storage Appliance, and a master zone has been set up on the SPARC server. Refer to the section below and to Appendix A for a list of all the scripts that will be needed to implement this solution and for sample output.

The setup above is done only once to prepare and initialize the environment.

Perform the Clone Operation

To clone an Oracle Solaris Zone environment and its database, all you need to do is execute the `cloneEnv.sh` script as `root` on the system that will host the clones.

The first execution of the `cloneEnv` script lists some files in the `/backup` directory that need to be removed in order to complete a successful clone operation. Copy and save the complete backup files to a separate directory, and then delete the listed files, as indicated.

On the Sun ZFS Storage Appliance, delete the snapshots and all the clones by clicking the garbage can icon shown in Figure 12, and create them again, as described previously. Execute the `delAll.sh` script to remove all the cloned zones, and then execute the `cloneEnv.sh` script.

This script took seven minutes to complete the cloning of four environments. Once completed, all the Oracle Solaris Zones and the Oracle databases are up and running.

To perform a successful clone operation repeatedly using the same clone hostnames and IP addresses identified in the `setup.sh` file, the following steps are required:

1. On the Sun ZFS Storage Appliance, delete the snapshots and all the clones by clicking the garbage can icon shown in Figure 12, and create them again, as described previously, every time a new clone operation is performed.
2. As `root`, execute the `delAll.sh` script to reset the environment.
3. As `root`, execute the `cloneEnv.sh` script.

To add a new clone, create a new clone on the Sun ZFS Storage Appliance, as shown in Figure 13, and update the following variables in the `setup.sh` file to include the new information: `cloneZones`, `cloneHostNames`, `cloneIpAddress`, and `cloneVnics`. Make sure a VNIC is created for the new zone, as shown at the beginning of the section "Create an Oracle Solaris Master Zone"; the name of the new VNIC needs to be reflected in the `cloneVnics` variable in the `setup.sh` file.

To delete a cloned zone and its database, run the `delZone.sh` script after providing the name of the zone as input. To delete all the zones and their databases, run the `delAll.sh` script. Deleting a zone has no impact on other cloned zones or on the master zone.

Summary

This featured solution for cloning an Oracle Solaris and Oracle Database environment aims to streamline and simplify the development and testing of applications in the data center by providing a simple, foolproof, and cost-effective way to test upgrades and patches on systems that are virtually identical to production systems, without impacting production. The simplicity and effectiveness of this unique solution further exemplifies the benefits of using "Oracle on Oracle" technology.

For More Information

For more information on Oracle's technology stack, see the references in Table 2.

TABLE 2. REFERENCES FOR MORE INFORMATION

WEBSITES AND SUPPORT NOTE	
Oracle Optimized Solutions	http://oracle.com/optimizedsolutions
Oracle's SPARC T-Series servers	http://www.oracle.com/goto/tseries
Oracle Solaris 11	http://www.oracle.com/solaris
Oracle Solaris 11 information on the Oracle Technology Network Website	http://www.oracle.com/technetwork/server-storage/solaris11/overview/index.html
Oracle's Sun ZFS Storage Appliance	http://www.oracle.com/us/products/servers-storage/storage/unified-storage/
My Oracle Support note (requires login)	https://support.oracle.com/epmos/faces/DocumentDisplay?id=1210656.1
ORACLE SOLARIS WHITE PAPERS	
"Oracle Solaris 11 Network Virtualization and Network	http://www.oracle.com/technetwork/server-

Resource Management"	storage/solaris11/documentation/o11-137-s11-net-virt-mgmt-525114.pdf
"Integrated Application-to-Disk Management with Oracle Enterprise Manager Cloud Control 12c"	http://www.oracle.com/technetwork/oem/enterprise-manager/wp-em-a2d-mgmt-12-1-1585513.pdf
"Oracle Solaris and Oracle SPARC T4 Servers—Engineered Together for Enterprise Cloud Deployments"	http://www.oracle.com/us/products/servers-storage/solaris/solaris-and-sparc-t4-497273.pdf

Appendix A—Solution Scripts

The following scripts are used in this solution. As `root` in the global zone, copy the scripts into a directory (in this example, `/demo/cloneEnv`), and modify the access mode as follows:

```
root@dat01:/demo/cloneEnv# ls -l
-rwxr-xr-x  1 root  root   525 Mar  5 11:22 addNFS.sh
-rwxr-xr-x  1 root  root  1534 Mar  5 11:22 cloneAllDB.sh
-rwxr-xr-x  1 root  root  1704 Mar  5 11:22 cloneDB.sh
-rwxr-xr-x  1 root  root  2418 Mar  5 11:42 cloneEnv.sh
-rwxr-xr-x  1 root  root  3070 Mar  5 11:22 cloneZone.sh
-rwxr-xr-x  1 root  root  7851 Mar  5 11:23 clonedb.pl
-rwxr-xr-x  1 root  root   220 Mar  5 11:22 createControl.sh
-rwxr-xr-x  1 root  root   339 Mar  5 11:22 delAll.sh
-rwxr-xr-x  1 root  root  1565 Mar  5 11:22 delZone.sh
-rwxr-xr-x  1 root  root   485 Mar  5 11:22 exportCfg.sh
-rwxr-xr-x  1 root  root   410 Mar  5 11:22 haltZone.sh
-rwxr-xr-x  1 root  root   206 Mar  5 11:22 runClone.sh
-rwxr-xr-x  1 root  root   411 Mar  5 11:22 setenv.sh
-rwxr-xr-x  1 root  root  1040 Mar  6 18:12 setup.sh
-rwxr-xr-x  1 root  root   328 Mar  5 11:22 verifyDB.sh
```

The `setup.sh` Script

```


#This script lists the names of the zones to be created, and includes their IP addresses.
#The content of this script needs to be modified to reflect your environment every time
#you create a new clone. This file contains information to create 4 new clones named
#"clone1", "clone2", "clone3", and "clone4".

#Name of the master zone that needs to be cloned
masterZone=master

#The full path of the configuration file of the master zone mounted from ZFS Storage
Appliance
configFile=/config/master.xml

#List the names of the clones that need to be cloned. The names must match the names entered
in Figure 17
cloneZones=( "clone1" "clone2" "clone3" "clone4" )

#Host names of the clones
cloneHostNames=("dat-zone2" "dat-zone3" "dat-zone4" "dat-zone5" )

#IP Addresses of the clones
cloneIpAddress=( "10.133.82.45" "10.133.82.46" "10.133.82.47" "10.133.82.48" )

#VNICs to be used by each clone, and created in the "Create an Oracle Solaris Master Zone"
section
cloneVnics=("clone1" "clone2" "clone3" "clone4" )

#Variable to enable debugging of the scripts
debug=0

#Location of the master database backup mounted from ZFS Storage Appliance
masterBackupLocation="/backup"

#Name of master Zone ORACLE_SID that will be cloned in all zones
databaseName="tpcc"

#Full path and name of init.ora file that will be used by the cloned database and mounted
```

```
from ZFS Storage Appliance
oraLocation=/config/inittpcc.ora

#IP address of Sun ZFS Storage Appliance that hosts the backup and clones
ZfsSA="10.133.83.224"
```

The `cloneEnv.sh` Script

```
#cloneEnv.sh is the top level script that invokes the following scripts:
#haltZone.sh
#exportCfg.sh
#cloneZone.sh
#cloneAllDB.sh

#cloneEnv.sh script clones a master zone. It also starts a database in that cloned zone.
#In order to clone the master zone, this script takes the following input
# The name of the master zone
# The name of the configuration file of the master zone
# List of names of the clones that needs to be cloned
#  A list of VNICs that each zone will use
# Host names of the clones to be cloned
# IP Addresses of the clones to be cloned.

#The cloneEnv.sh script assumes that all clones are created on the same network as the master
zone.
#It uses the configuration file used by the master zone and updates the host name, VNICs and
IP Address information.

. ./setup.sh
count=0

#Start nfs/client services in masterZone
zlogin -l root $masterZone "svcadm enable svc:/network/nfs/status:default;svcadm enable
svc:/network/nfs/client:default "

#Halt master zone
./haltZone.sh  $masterZone
#Export the configuration of master zone
./exportCfg.sh $masterZone

start=`gdate +"%s"`

for i in "${cloneZones[@]}"
do
        cloneZone=$i
        cloneHostName=${cloneHostNames[$count]}
        cloneIpAddr=${cloneIpAddress[$count]}
        cloneVnic=${cloneVnics[$count]}

        if [ $debug == 1 ]
then
        echo " \n Zone to Clone : $cloneZone"
        echo "Clone host name : $cloneHostName"
        echo " Clone IP : $cloneIpAddr Clone Vnic : $cloneVnic "
fi
        #clone the zone from masterzone
        ./cloneZone.sh $masterZone $cloneZone $cloneHostName $cloneIpAddr $cloneVnic
$configFile $debug
        count=$(($count+1))

done

end=`gdate +"%s"`
```

```
duration=$(($end - $start ))
duration=$(($duration/60))
echo "\n\t$count Zones cloned in $duration minutes \n"

echo "\n\tVerify all clones are up and running\n"
zoneadm list -icv | grep clone

echo "\n\tWait for two minutes for SMF services to start in all zones\n"
sleep 120

echo "\n\tVerify network connectivity for all clones\n"
count=0
for i in "${cloneZones[@]}"
do
        cloneIpAddress=${cloneIpAddress[$count]}
        s=`ping $cloneIpAddress`
        echo "\t$i : $s"
        count=$count+1
done
#Clone the database in all the zones
./cloneAllDB.sh
#echo "\n\t Boot Master Zone  $masterZone\n"
echo "\n"
zoneadm -z $masterZone boot
if [ $? != 0 ]
then
        echo "Error booting $masterZone ...."
fi
```

The `haltZone.sh` Script

```
#haltZone.sh takes zoneName as an input parameter and halts that zone

zoneName=$1

#echo "Halt $zoneName so it can be cloned"
state=`zoneadm -z "${zoneName}" list -p | cut -d: -f3`
if [ $state ==  "running" ]
then
        echo "Halt $zoneName"
        zoneadm -z ${zoneName} halt
        if [ $? != 0 ]
        then
                echo "Error halting $zoneName ...Exiting "
                exit
        fi

fi
```

The `exportCfg.sh` Script

```
#exportCfg.sh takes a zone name as input and saves its zone configuration in a file named
master in /tmp directory

zoneName=$1
zoneExist=`zoneadm list -icv | grep "$zoneName" | awk '{print $2}'`
if [ ! $zoneExist  ]
then
        echo "$zoneName does not exist .. Exiting ..."
        exit
```

```
fi

echo "Read ${zoneName} zone configuration "
zonecfg -z ${zoneName} export -f /tmp/master
if [ $? != 0 ]
then
        echo "Error exporting the config info for ${zoneName} "
        exit 1
fi
```

The `cloneZone.sh` Script

```
#cloneZone.sh clones a single zone from the master zone.
#It takes master zone name and master zone's system configuration file as input.
#It also takes clone name, clone hostname, clone IP address and clone's VNIC as input.
#It updates the master zone's zone configuration file with clone name and clone VNIC.
#It then uses the updated zone configuration file to configure the clone.
#cloneZone.sh also updates the master zone's system configuration file with clone
#name, clone hostname, clone IP address and clone VNICs. It uses the updated system
#configuration file to clone the new zone from the master zone. It then boots up the cloned
zone.

masterZone=$1
cloneZone=$2
cloneHostName=$3
cloneIpAddress=$4
cloneVnic=$5
configFile=$6  #system configuration file for the master zone
masterRoot=oraclePool
cloneRoot=oraclePool
debug=$7

echo "\n\n\tClone ${cloneZone} from ${masterZone} "

if [ $debug == 1 ]
then
        echo "Master zone: $masterZone"
        echo "clone Zone: $cloneZone"
        echo "clone Host Name : $cloneHostName"
        echo "clone IP : $cloneIpAddress"
        echo "Clonevnic : $cloneVnic"
        echo "Config file : $configFile"
fi


echo "Update ${masterZone} zone configuration for ${cloneZone} configuration"
sed "s/${masterZone}/${cloneZone}/g" /tmp/master > /tmp/m1
sed "s/.*set physical.*/set physical=\\${cloneVnic}/g" /tmp/m1 > /tmp/m2
sed "s/${masterRoot}/${cloneRoot}/g" /tmp/m2 > /tmp/clone

echo "Configure ${cloneZone}"
zonecfg -z ${cloneZone} -f /tmp/clone > /tmp/$0.log
if [ $? != 0 ]
then
        echo "Configuring ${cloneZone} failed "
        exit 1
fi

echo "Update $masterZone system configuration file for $cloneZone configuration"
search="<propval type=\"astring\" name=\"nodename\" value=\".*\"\/>"
replace="<propval type=\"astring\" name=\"nodename\" value=\"${cloneHostName}\"\/>"
sed "s/$search/$replace/g" ${configFile} > /tmp/c1
```

```
search="<propval type=\"net_address_v4\" name=\"static_address\" value=\"[0-9]\{1,3\}\.[0-
9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}"
replace="<propval type=\"net_address_v4\" name=\"static_address\" value=\"${cloneIpAddress}"
sed "s/$search/$replace/g" /tmp/c1 > /tmp/c2


search="<propval type=\"astring\" name=\"name\" value=\".*\/v4\"\/>"
replace="<propval type=\"astring\" name=\"name\" value=\"${cloneVnic}\/v4\"\/>"
sed "s/$search/$replace/g" /tmp/c2 > /tmp/c3


search="<propval type=\"astring\" name=\"name\" value=\".*\/v6\"\/>"
replace="<propval type=\"astring\" name=\"name\" value=\"${cloneVnic}\/v6\"\/>"
sed "s/$search/$replace/g" /tmp/c3 > /tmp/c4.xml


echo "Clone ${cloneZone} from ${masterZone}"
zoneadm -z ${cloneZone} clone -c /tmp/c4.xml ${masterZone} >> /tmp/$0.log
if [ $? != 0 ]
then
        echo "Cloning the zone ${cloneZone} ...Exiting"
        exit
fi
echo "Boot ${cloneZone}"
zoneadm -z ${cloneZone} boot  >> /tmp/$0.log
if [ $? != 0 ]
then

        #echo "Booting again"
        zoneadm -z ${cloneZone} boot >> /tmp/$0.log
        if [ $? != 0 ]
        then
                echo "Booting of $cloneZone has failed"
                exit 1
        fi
fi
```

The `cloneAllDB.sh` Script

```
#cloneAllDB.sh invokes cloneDB.sh script to clone the DB in all the zones
# provided in cloneZones array in setup.sh. It then verifies that database has been
#successfully started in all the zones by logging in to each database. It invokes
# verifyDB.sh script to log in to database in each zone and print its status

. ./setup.sh

echo "\n\tVerify if there are any extra files in $masterBackupLocation\n\n"
#remove the / from masterBackupLocation for ignore clause of find commad
masterBack=${masterBackupLocation#'/'}

fileList=` find $masterBackupLocation \\( ! -iname "*data*"  ! -iname $masterBack  ! -iname
".*" \\)`

if [[ -n $fileList ]]
then
        echo  "\tBackup Directory $masterBackupLocation contains more files than the needed
data backup files. Copy and save this backup to a different location. Remove the following
extra files. Recreate the snapshot and clones on ZFSSA. Execute delAll.sh script, and try
again:\n"
        echo "${fileList// /$'\n'}"
        exit 2
fi
```

```
start=`gdate +"%s"`
count=0

for i in "${cloneZones[@]}"
do
        ./cloneDB.sh  $i
        ret=$?
        count=$(($count+1))
done
if [ $ret = 2 ]
then
        exit 2
fi
end=`gdate +"%s"`
duration=$(($end - $start ))
duration=$(($duration/60))
echo "\n\t$Oracle Database is  cloned in $count Zones in $duration minutes \n"

echo "\n\tVerify that  databases are up in all zones"
list=`ps -Zef | grep ora_smon | grep -v grep`
echo "$list"

for i in "${cloneZones[@]}"
do
        echo "\n\tLogin to database in $i\n"
        zlogin -l oracle ${i}  /tmp/verifyDB.sh
done
```

The `cloneDB.sh` Script

```
#The cloneDB.sh scripts takes the zone name as an input parameter and clones the database
#in that zone. cloneDB.sh uses the following scripts:
#addNFS.sh to add an NFS mount point to the zone
#setenv.sh script to set the environment to execute clone.pl scripts
#clonedb.pl to clone the database

. ./setup.sh
oraFile=`basename $oraLocation`
oraDir=`dirname $oraLocation`
zoneName=$1
zpath=`zonecfg -z "$zoneName" info zonepath | cut -d: -f2`

if [ $debug == 1 ]
then
        echo "zpath: $zpath"
        echo "oraFile : $oraFile"
        echo "oraLocation : $oraLocation"
fi

##Copy clonedb.pl verifyDB.sh creaetcontrol.sh to the the tmp directory of zone
cp clonedb.pl ${zpath}/root/tmp
cp verifyDB.sh ${zpath}/root/tmp
cp runClone.sh ${zpath}/root/tmp
cp createControl.sh  ${zpath}/root/tmp

#Dynamically generate setenv file from .setenv.sh and copy it to /export/home/oracle
directory of clone

./setenv.sh  $zoneName
cp setenv ${zpath}/root/export/home/oracle

#Dynamically generate addNFS file from addNFS.sh and copy it to /tmp directory of the zone
```

```
./addNFS.sh ${zoneName}
chmod 777 addnfs
cp addnfs ${zpath}/root/tmp

if [ $debug == 1 ]
then
        echo "\n\t Contents of addnfs file :\n"
        cat $zpath/root/tmp/addnfs
        echo "\n\tContents of setenv\n"
        cat $zpath/root/export/home/oracle/setenv
fi
#Login to the zone and mount that nfs directory
echo "\n\tMount NFS filesystem $zoneName in $zoneName\n"
zlogin ${zoneName} /tmp/addnfs
echo "\n\tClone database in $zoneName  \n"
zlogin -l oracle ${zoneName} "  /tmp/runClone.sh $zoneName $oraLocation"
ret=$?
exit $ret
```

The `runClone.sh` Script

```
#runclone.sh takes zonename and location of ora file as input and invokes
#the clonedb.pl script for that particular zone using the ora file
zoneName=$1
oraLocation=$2
. ./setenv
cd /tmp
perl /tmp/clonedb.pl $oraLocation /tmp/createControl.sql /tmp/openDB.sql
if [ $? == 0 ]
then
        createControl.sh
        cp /tmp/init*.ora /$zoneName
else
        exit 2
fi
```

The `addNFS.sh` Script

```
#addNFS.sh script adds an NFS mount point to the /etc/vfstab file of the clone and mounts
#that file system. It takes the zone name as its input parameter and mounts the
#ZFSSA/export/<clonename> file system on that zone.

zoneName=$1

. ./setup.sh
echo "echo "$ZfsSA:/export/$zoneName    -       /$zoneName      nfs     -       yes
rw,bg,hard,nointr,rsize=32768,wsize=32768,proto=tcp,noac,forcedirectio,vers=3,suid " >>
/etc/vfstab" > addnfs
echo "mkdir /$zoneName " >> addnfs
echo "mount /$zoneName >> /tmp/$0.log"  >> addnfs
echo "ls -ltr /$zoneName >> /tmp/$0.log" >> addnfs
```

The `setenv.sh` Script

```
#setenv.sh sets up the environmental variable required to execute the clonedb.pl script.
#it takes zoneName as its input parameter and sets CLONE_FILE_CREATE_DEST to /$zonename

. ./setup.sh
zoneName=$1
echo "export MASTER_COPY_DIR=$masterBackupLocation " > setenv
echo "export CLONEDB_NAME=$databaseName " >> setenv
echo "export CLONE_FILE_CREATE_DEST=/$zoneName" >> setenv
echo "export S7000_TARGET=1" >> setenv
```

The `clonedb.pl` Script

```
# This script is a slightly modified version of the clone.pl script
#that is downloaded from My Oracle Support Metalink?Note 1210656.1
#Changes made
# Saves the new ora file in /tmp directory so when you run create control file
#command, the ora file is not included in list of data files
#set db_recovery_file_dest to $clonedbdir in new ora file
#exit the script if the clonedbdir has any files in addition to data files
# clonedb.pl - This script generates two SQL scripts that can be used to
# create your test clones. Run this from your testdb Oracle Home environment
#
# Before running this script make sure the following environment variables are set:
#
# MASTER_COPY_DIR - environment variable to point to the directory where the
#                   backup/snapshot of your Master database are kept
#
# CLONE_FILE_CREATE_DEST - environment variable to point to the directory where
#                          clonedb files will be created including datafiles,
#                          log files, control files
#
# CLONEDB_NAME  - Cloned database name
#
# S7000_TARGET - Set if the nfs host providing the filesystem for the backup
#                and the clones is an S7000 series machine and we wish to
#                employ its cloning features.
#
# perl clonedb.pl <master_db.ora> <crtdb.sql> <dbren.sql>
#
# Arg1 - Full path of the Master db init.ora file from your production env
# Arg2 - sqlscript1
# Arg3 - sqlscript2
#
# This script copies the init.ora file from your master db env to your
# clonedb  env in CLONE_FILE_CREATE_DEST directory.
#
# After running this script go through the test database parameter file to
# make sure all parameters look correct
#
# Go through crtdb.sql to make sure the log names are correct.
# If all files look good do the following steps
#
# sqlplus system/manager
# @crtdb.sql
# @dbren.sql
#
# Now your test database should be available for use.
#

if ($#ARGV != 2) {
 print "usage: perl clonedb.pl <master_init.ora> <crtdb.sql> <dbren.sql> \n";
 exit;
}

if (!$ENV{'MASTER_COPY_DIR'}) {
 print "MASTER_COPY_DIR env not set. Set this and rerun it again \n";
 exit;
}
```

```
if (!$ENV{'CLONE_FILE_CREATE_DEST'}) {
 print "CLONE_FILE_CREATE_DEST env not set. Set this and rerun it again \n";
 exit;
}

if (!$ENV{'CLONEDB_NAME'}) {
 print "CLONEDB_NAME env not set. Set this and rerun it again \n";
 exit;
}



($orafile)=$ARGV[0];
($sqlfile1)=$ARGV[1];
($sqlfile2)=$ARGV[2];

# Set environment variable specific to your clone database
#$neworafile ="$ENV{'CLONE_FILE_CREATE_DEST'}/init$ENV{'CLONEDB_NAME'}.ora";
$neworafile ="/tmp/init$ENV{'CLONEDB_NAME'}.ora";
$cldboh = "$ENV{'ORACLE_HOME'}";
$cldbosid = "$ENV{'ORACLE_SID'}";
$cldbname = "$ENV{'CLONEDB_NAME'}";
$cldbctlfl = "$ENV{'CLONE_FILE_CREATE_DEST'}/$ENV{'CLONEDB_NAME'}_ctl.dbf";
$mastercopydir = "$ENV{'MASTER_COPY_DIR'}";
$clonedbdir ="$ENV{'CLONE_FILE_CREATE_DEST'}";
$s7000 = $ENV{S7000_TARGET} ? 1 : 0 ;

# Check if the CLONE_FILE_CREATE_DEST exists
if (! open(CLONEDIR, $clonedbdir))
{
  print("CLONE_FILE_CREATE_DEST directory does not exist.\n");
  print("Create this directory and rerun the script \n");
  exit;
}
close(CLONEDIR);

#remove the / from clonedbdir for ignore clause of find command
$clonedb=substr $clonedbdir,1 ;
#List all files in $clonedbdir directory except $clonedbdir, any file
#having data in its name and any hidden file
$fileList = ` find $clonedbdir \\( ! -iname "*data*"  ! -iname $clonedb  ! -iname ".*" \\)` ;
#if $fileList is not empty that means the directory has more files than
#data files, this is a fatal error
if ($fileList)
{
        print("\n\tFATAL ERROR : Clone Directory $clonedbdir contains more files than data
backup files." );
        print(" Recreate the snapshot and clones on ZFSSA. Execute delAll.sh script, and try
again. \n\n ");
        print($fileList);
        exit 2;
}

# Rename the parameters in the copied production init.ora and open a new init.ora with new
values
open (INFILE,$orafile);
open (OUTFILE,">$neworafile");
@skipparam=("instance_name","service_names","diagnostic_dest");
@inparam=("db_name","control_files");
@outparm=($cldbname,$cldbctlfl);
$skiplen = @skipparam;
$inlen = @inparam;

for $ln (<INFILE>)
{
  $newln = $ln;

#Look for any include files and read their contents
  if ($newln =~ "ifile")
  {
    @lnsp = split("=",$newln);
    open(INCFILE, $lnsp[1]);
    print OUTFILE "# Copy from $lnsp[1] \n";

    for $ln (<INCFILE>)
    {
      $newln = $ln;
      for ($i=0; $i<$skiplen; $i++){
        if ($newln =~ /$skipparam[$i]/)
        {
          $newln="\n";
        }
      }

      for ($i=0; $i<$inlen; $i++){
        if ($newln =~ /$inparam[$i]/)
        {
          @lnsp = split("=",$newln);
          $lnsp[1]=$outparm[$i];
          $newln=$inparam[$i]."=".$lnsp[1]."\n";
        }
      }
      print OUTFILE "$newln";
    }
    close INCFILE;
    print OUTFILE "# End Copy";
  }
  else
  {

    for ($i=0; $i<$skiplen; $i++){
      if ($newln =~ /$skipparam[$i]/)
      {
        $newln="\n";
      }
    }

    for ($i=0; $i<$inlen; $i++){
      if ($newln =~ /$inparam[$i]/)
      {
        @lnsp = split("=",$newln);
        $lnsp[1]=$outparm[$i];
        $newln=$inparam[$i]."=".$lnsp[1]."\n";
      }
    }
    print OUTFILE "$newln";
  }
}

# Add db_create_file_dest, log_arhive_dest parameter
print OUTFILE "db_create_file_dest=$clonedbdir\n";
print OUTFILE "log_archive_dest=$clonedbdir\n";
print OUTFILE "db_recovery_file_dest=$clonedbdir\n";
#print OUTFILE "clonedb=TRUE\n";

close INFILE;
close OUTFILE;

# Create clone db raneame file sql
if (!$s7000)
{
  $target=$mastercopydir;
} else {
  $target=$clonedbdir;
}

# XXX Needs to be modified to just deal with data files.
system ("cd $target; ls -d $target/* >> dnfsa1axxx.log");
system ("cp $target/dnfsa1axxx.log .;rm $target/dnfsa1axxx.log");
open(INPFILE,"dnfsa1axxx.log");
open(INTFILE,">filenamexxx.txt");
open(OUTFILE1,">$sqlfile2");
open(OUTFILE,">dnfsa2axxx.log");

for $ln (<INPFILE>)
{
       print INTFILE "$ln";
}
close INTFILE;
close INPFILE;

open(INTFILE,"filenamexxx.txt");
$refline=" ";
for $line (<INTFILE>)
{
  $line =~ s/\s+$//;
  if ($refline ne " ")
  {
   print OUTFILE  "'"."$refline"."'".", \n";
  }
  $refline = $line;
}
if ($refline ne " ")
{
 print OUTFILE  "'"."$refline"."' \n";
}
close INTFILE;

if (!$s7000)
{
  print OUTFILE1 "declare \n";
  print OUTFILE1 "begin \n";
  open(INTFILE,"filenamexxx.txt");
  $i = 0;
  for $lne (<INTFILE>)
  {
    $lne =~ s/\s+$//;
    print OUTFILE1 "dbms_dnfs.clonedb_renamefile('$lne' ,
'$clonedbdir/\ora_data_$cldbname$i.dbf'); \n";

    $i++;
  }
  print OUTFILE1 "end; \n";
  print OUTFILE1 "/ \n";
  print OUTFILE1 "show errors; \n";
}

print OUTFILE1 "alter database open resetlogs;\n";

#Add a default temp tablespace in the clone env
print OUTFILE1 "drop tablespace TEMP;\n";
print OUTFILE1 "create temporary tablespace TEMP;";

close OUTFILE;
close OUTFILE1;
close OUTFILE1;
close OUTFILE1;
```

```
# Create the create controlfile script
open(INPFILE1,"dnfsa2axxx.log");
open(INPSQLFILE,">interm.sql");
open (OUTSQLFILE,">$sqlfile1");
print INPSQLFILE ("
SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100

STARTUP NOMOUNT PFILE=$neworafile
CREATE CONTROLFILE REUSE SET DATABASE $cldbname RESETLOGS
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXINSTANCES 1
    MAXLOGHISTORY 908
LOGFILE
  GROUP 1 '$clonedbdir/$cldbname\_log1.log' SIZE 100M BLOCKSIZE 512,
  GROUP 2 '$clonedbdir/$cldbname\_log2.log' SIZE 100M BLOCKSIZE 512
DATAFILE
CHARACTER SET WE8DEC; ");
close INPSQLFILE;

open(INPSQLFILE,"interm.sql");
for $ln (<INPSQLFILE>)
{
   print OUTSQLFILE "$ln";
   if ($ln =~ /DATAFILE/)
   {
     for $ln0 (<INPFILE1>)
        {
        print OUTSQLFILE "$ln0";
        }
   }
}

close OUTSQLFILE;
close INPFILE1;
close INPSQLFILE;

unlink("interm.sql");
unlink("dnfsa1axxx.log");
unlink("dnfsa2axxx.log");
unlink("filenamexxx.txt");
```

The `createControl.sh` Script

```
#createControl.sh executes createcontrol.sql and resets the log for the cloned database.

sqlplus -s / as sysdba << !
spool /tmp/createControl.log
set echo off
@createControl.sql
alter database open resetlogs;
exit
!
```

The `verifyDB.sh` Script

```
#verifyDB.sh logs in to the database and prints out the
#hostname, instance_name and status of the database
sqlplus  -s / as sysdba << !
set  define off
set escape on
column instance_name format a20
column host_name format a20
column status format a10
select instance_name, host_name,status from v\$instance ;
set define on
!
```

The `delAll.sh` Script

```
#This scripts deletes all the zones that are specified in input array cloneZones, which
# can be specified in setup.sh file
. ./setup.sh
for i in "${cloneZones[@]}"
do
        cloneZone=$i
        if [ $debug == 1 ]
        then
                echo "  Zone to Delete : $cloneZone"
        fi
        ./delZone.sh  $cloneZone
done
exit
```

The `delZone.sh` Script

```
#This script deletes a specific zone.
if [ $# != 1 ]
then
        echo "Zone name not provided : ${zoneName}. Exiting ..."
        exit 1
fi
zoneName=$1

echo "\n\n\t\t Deleting ${zoneName}"

zoneExist=`zoneadm list -icv | grep "$zoneName" | awk '{print $2}'`
if [ ! $zoneExist  ]
then
        echo "$zoneName does not exist .. Exiting ..."
        exit

fi
state=`zoneadm -z "${zoneName}" list -p | cut -d: -f3`
if [ $state ==  "running" ]
then
        echo "Halting $zoneName"
        zoneadm -z ${zoneName} halt
        if [ $? != 0 ]
        then
                echo "Error halting $zoneName ...Exiting "
                exit
        fi
        #check return code

fi
```

```
state=`zoneadm -z "${zoneName}" list -p | cut -d: -f3`
if [ $state ==  "installed"  -o  $state == 'incomplete' ]
then
        echo "Uninstalling $zoneName"
        zoneadm -z ${zoneName} uninstall -F > /tmp/$0.log
        if [ $? != 0 ]
        then
                echo "Error uninstalling $zoneName ...Exiting "
                exit
        fi
        #check return code
fi

state=`zoneadm -z "${zoneName}" list -p | cut -d: -f3`
if [ $state == "configured" ]
then
        echo "Unconfiguring $zoneName"
        zonecfg -z ${zoneName} delete -F
        if [ $? != 0 ]
        then
                echo "Error unconfiguring ${zoneName} ...Exiting"
                exit
        fi
fi

zoneExist=`zoneadm list -icv | grep "$zoneName" | awk '{print $2}'`
if [  $zoneExist  ]
then
        echo " Error deleting $zoneName  Exiting ..."
        exit
fi
echo "\n\t\t$zoneName successfully deleted"
```

Sample Output

The following sample output is obtained after running the `cloneEnv.sh` script:

```
root@dat01:/demo/cloneEnv# ./cloneEnv.sh
Halt master

Read master zone configuration



        Clone clone1 from master
Update master zone configuration for clone1 configuration
Configure clone1
Update master system configuration file for clone1 configuration
Clone clone1 from master
Boot clone1


        Clone clone2 from master
Update master zone configuration for clone2 configuration
Configure clone2
Update master system configuration file for clone2 configuration
Clone clone2 from master
Boot clone2


        Clone clone3 from master
Update master zone configuration for clone3 configuration
Configure clone3
```

```
Update master system configuration file for clone3 configuration
Clone clone3 from master
Boot clone3


        Clone clone4 from master
Update master zone configuration for clone4 configuration
Configure clone4
Update master system configuration file for clone4 configuration
Clone clone4 from master
Boot clone4

        4 Zones cloned in 2 minutes


        Verify all clones are up and running

  338 clone1           running    /rpool/zones/clone1            solaris  excl
  340 clone2           running    /rpool/zones/clone2            solaris  excl
  342 clone3           running    /rpool/zones/clone3            solaris  excl
  344 clone4           running    /rpool/zones/clone4            solaris  excl

        Wait for two minutes for SMF services to start in all zones


        Verify network connectivity for all clones

        clone1 : 10.133.82.45 is alive
        clone2 : 10.133.82.46 is alive
        clone3 : 10.133.82.47 is alive
        clone4 : 10.133.82.48 is alive

        Mount NFS filesystem clone1 in clone1


        Clone database in clone1

Oracle Corporation      SunOS 5.11      11.1    December 2012
ORACLE instance started.

Total System Global Area 9055346688 bytes
Fixed Size                  2166760 bytes
Variable Size            1677725720 bytes
Database Buffers         7079985152 bytes
Redo Buffers              295469056 bytes

Control file created.


Database altered.


        Mount NFS filesystem clone2 in clone2


        Clone database in clone2

Oracle Corporation      SunOS 5.11      11.1    December 2012
ORACLE instance started.

Total System Global Area 9055346688 bytes
Fixed Size                  2166760 bytes
Variable Size            1677725720 bytes
Database Buffers         7079985152 bytes
Redo Buffers              295469056 bytes
```

```
Control file created.


Database altered.


        Mount NFS filesystem clone3 in clone3


        Clone database in clone3

Oracle Corporation      SunOS 5.11      11.1    December 2012
ORACLE instance started.

Total System Global Area 9055346688 bytes
Fixed Size                  2166760 bytes
Variable Size            1677725720 bytes
Database Buffers         7079985152 bytes
Redo Buffers              295469056 bytes

Control file created.


Database altered.


        Mount NFS filesystem clone4 in clone4


        Clone database in clone4

Oracle Corporation      SunOS 5.11      11.1    December 2012
ORACLE instance started.

Total System Global Area 9055346688 bytes
Fixed Size                  2166760 bytes
Variable Size            1677725720 bytes
Database Buffers         7079985152 bytes
Redo Buffers              295469056 bytes

Control file created.


Database altered.


         Database is  cloned in 4 Zones in 2 minutes


        Verify that  databases are up in all zones
  clone3  0000200  5454     1   0 18:50:56 ?           0:00 ora_smon_tpcc
  clone2  0000200  5231     1   0 18:50:20 ?           0:00 ora_smon_tpcc
  clone1  0000200  5019     1   0 18:49:44 ?           0:00 ora_smon_tpcc
  clone4  0000200  5674     1   0 18:51:33 ?           0:00 ora_smon_tpcc

        Login to database in clone1

Oracle Corporation      SunOS 5.11      11.1    December 2012

INSTANCE_NAME        HOST_NAME            STATUS
-------------------- -------------------- ----------
tpcc                 dat-zone2            OPEN


        Login to database in clone2
```

```
Oracle Corporation      SunOS 5.11      11.1    December 2012

INSTANCE_NAME        HOST_NAME            STATUS
-------------------- -------------------- ----------
tpcc                 dat-zone3            OPEN


        Login to database in clone3

Oracle Corporation      SunOS 5.11      11.1    December 2012

INSTANCE_NAME        HOST_NAME            STATUS
-------------------- -------------------- ----------
tpcc                 dat-zone4            OPEN


        Login to database in clone4

Oracle Corporation      SunOS 5.11      11.1    December 2012

INSTANCE_NAME        HOST_NAME            STATUS
-------------------- -------------------- ----------
tpcc                 dat-zone5            OPEN
```

The following output is obtained after running the `delAll.sh` script:

```
root@dat01:/demo/cloneEnv# ./delAll.sh


                 Deleting clone1
Halting clone1
Uninstalling clone1
Unconfiguring clone1

                clone1 successfully deleted


                 Deleting clone2
Halting clone2
Uninstalling clone2
Unconfiguring clone2

                clone2 successfully deleted


                 Deleting clone3
Halting clone3
Uninstalling clone3
Unconfiguring clone3

                clone3 successfully deleted


                 Deleting clone4
Halting clone4
Uninstalling clone4
Unconfiguring clone4

                clone4 successfully deleted
```

Appendix B—Solution Hardware Component Descriptions

SPARC T4 Server Overview

The SPARC T4-4 server, shown in Figure 18, includes four sockets (each with an eight-core, 3.0 GHz, SPARC T4 processor), two solid-state disks, and up to a 1 TB memory footprint. Fifth-generation multicore, multithreading technology supports 8 threads per core and up to 256 threads per four-socket server, providing high compute density in only five rack units (5RU) with low power and cooling. The large number of cores and virtual CPUs coupled with the large memory footprint, integrated on-chip I/O technology, and built-in supported virtualization make the SPARC T4-4 server ideal for deploying large number of databases. With breakthrough levels of price/performance (over 20 performance world records), the SPARC T4-4 server is capable of providing high throughput within significant power, cooling, and space constraints.

The SPARC T4 processor includes an integrated cryptographic accelerator unit in each of the eight cores. This means Oracle Solaris applications can run securely without the extra cost of a separate cryptographic processor and without the CPU overhead associated with secure operation. The SPARC T4 processor's integrated cryptographic units support seventeen of the most common ciphers and secure hashing functions, and they outperform solutions based on add-in accelerator cards by more than 10x.

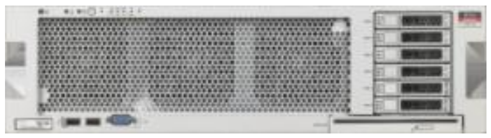


Figure 18. Oracle's SPARC T4-4 server.

The SPARC T4 processor offers a multithreaded hypervisor that interacts directly with the underlying multicore and multithreading processor. This makes is possible to context-switch between multiple threads in a single core, which normally requires additional software and considerable overhead in competing architectures. In addition to the processor and hypervisor, Oracle provides fully multithreaded networking and the fully multithreaded Oracle Solaris ZFS file system.

SPARC T5-4 Server Overview

In a dense 5U form-factor, the SPARC T5-4 is a high-performing, four-socket server optimized for data-intensive, large Oracle Database workloads. It delivers unsurpassed single-thread and multithread throughput performance, with a 1.2x improvement in single-thread performance, 2.5x throughput improvement, and 2x increase in I/O bandwidth compared to the SPARC T4-4 system. For large-scale environments that require extremely high service levels, the SPARC T5-4 is an optimal platform for mission-critical server and application consolidation.

Figure 19. Oracle's SPARC T5-4 server.

Comparison of SPARC T4 and T5 Server Models

Tables 3 and 4 give a quick comparison of SPARC T4 and T5 server models. SPARC T4 servers are available in one-, two-, and four-socket implementations: the SPARC T4-1, T4-2, and T4-4 servers, respectively. SPARC T5 servers are available in blade, two-, four-, and eight-socket implementations: the SPARC T5-1, T5-2, T5-4, and T5-8 servers, respectively. For additional details and resources, see http://www.oracle.com/sparc.

TABLE 3. SPARC T4 SERVER FEATURES

	SPARC T4-1	**SPARC T4-2**	**SPARC T4-4**
Size (Rack Units)	2U	3U	5U
Processor	SPARC T4 2.85 GHz		SPARC T4 3.0 GHz
Max. Processor Chips	1	2	4
Max. Cores/Threads	8/64	16/128	32/256
Max. Memory	256 GB	512 GB	1 TB
PCIe Gen2 Slots	6	10	16
1 GbE/10 GbE Ports	4/2	4/4	4/8
Drive Bays (SAS)	8	6	8
Service Processor	Oracle ILOM		
Operating System	Oracle Solaris 11.1 or Oracle Solaris 10 1/13		
Virtualization Features	Oracle VM Server for SPARC (formerly known as Logical Domains), Oracle Solaris Zones		
Key RAS Features	Oracle ILOM, RAID 0/1, ECC correction; Redundant, hot-plug fans and power supplies		

TABLE 4. SPARC T5 SERVER FEATURES

	SPARC T5-1B	SPARC T5-2	SPARC T5-4	SPARC T5-8
Size (Rack Units)	Blade server	3U	5U	8U
Processor	SPARC T5 3.6GHz			
Max. Processor Chips	1	2	4	8
Max. Cores/Threads	16/128	32/256	64/512	128/1024
Max. Memory	512 GB	1 TB	2 TB	4 TB
Drive Bays	2	6	8	8
PCIe 3.0 Slots	2 modules	8	16	
10 GbE Ports	2x 10/100/1000 Ethernet connection	4x 10 GbE ports	4x 10 GbE ports	4x 10 GbE ports
Service Processor	Oracle ILOM			
Operating System	Oracle Solaris 11.1 or Oracle Solaris 10 1/13			
Virtualization Features	Oracle VM Server for SPARC, Oracle Solaris Zones			
Key RAS Features	Oracle ILOM, RAID 0/1, ECC correction			
	In blade chassis	Redundant, hot-plug fans and power supplies		
	N/A		Hot-plug disks and PCIe cards	

Sun ZFS Storage Appliance

The Sun ZFS Storage Appliance from Oracle provides enterprise-class network-attached storage (NAS) appliances that supports NAS protocols, and it provides storage-area network (SAN) connectivity via the iSCSI, Fibre Channel, and InfiniBand protocols.

The Sun ZFS Storage Appliance enhances performance while reducing both initial capital expenses and ongoing operating expenses. Key features include the following:

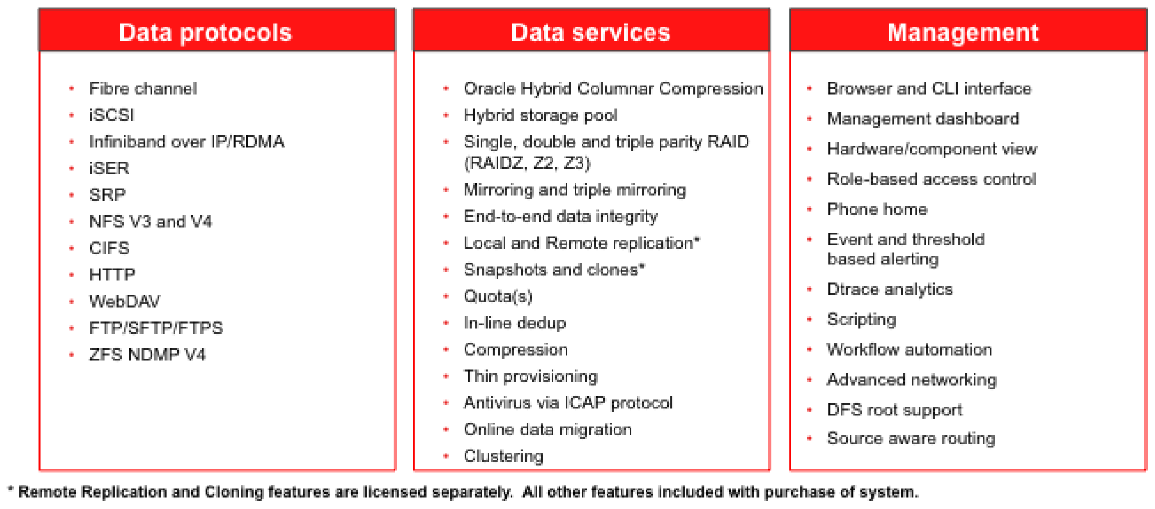
- An advanced, intuitive browser user interface (BUI) as well as a simple command line interface (CLI) that aim to reduce the time involved in provisioning and managing storage.
- A comprehensive DTrace Analytics environment, along with an intuitive interface, that provides real-time visibility into the CPU, memory, data, data protocol, disk, and network performance to help diagnose, troubleshoot, and resolve issues before they have an impact on business.
- Hybrid Storage Pool technology that optimizes the way data is spread across memory, solid state disks (SSDs), and disk storage.

- Tight integration with Oracle's stack, which further reduces operating expenses in Oracle environments through the use of standardized, well-developed, Oracle-supported configurations along with unique Oracle-on-Oracle features, such as Hybrid Columnar Compression, which is a feature of Oracle Database.
- Superior performance, as demonstrated from the results of benchmark tests featuring SPECsfs (a file protocol performance test) and block protocol workloads, which demonstrated both excellent random transactional performance (SPC-1) and extreme throughput performance (SPC-2). These tests were independently validated and are publicly available from the Standard Performance Evaluation Corporation and the Storage Performance Council.

Hybrid Storage Pool technology uses an intelligent and adaptive set of algorithms to automatically and dynamically manage read and write operations. The Sun ZFS Storage Appliance is able to cache data stored on disk in either DRAM or flash-based memory for low latency, high I/O, and high throughput access. Similarly, write operations can be made to low latency, nonvolatile flash so that they can be quickly acknowledged, allowing the system to move on to the next operation more quickly. As the data movement occurs, end-to-end Sun ZFS Storage Appliance checksumming prevents silent data corruption. This technology allows the Sun ZFS Storage Appliance to extract maximum performance from the hardware, enabling extreme performance while minimizing price.

Sun ZFS Storage Appliances are part of the complete, integrated Oracle stack, featuring Hybrid Columnar Compression for Oracle Database 11*g* and later releases. This feature is available only on Oracle storage and can result in substantially greater compression levels than can be obtained in other vendors' systems. Oracle's Hybrid Columnar Compression technology is a new method for organizing data within a database block. Hybrid Columnar Compression enables the highest levels of data compression and provides enterprises with tremendous cost savings and performance improvements due to reduced I/O. Hybrid Columnar Compression is optimized to use both database and storage capabilities to deliver tremendous space savings and revolutionary performance. Average storage savings can range from 20x to 50x, depending on the nature of the data. Taking a conservative average savings of 10x from Hybrid Columnar Compression, IT managers can drastically reduce and often eliminate their need to purchase new storage for a significant amount of time.

The rich set of data services and appliance options shown in Figure 20 further enhance the economic benefits of the Sun ZFS Storage Appliance. All protocols (such as CIFS, NFS, iSCSI, FC, and others) are included in the base price of the system. Various data protection options and several data compression options, as well as inline deduplication, are also included in the base price. Customers can choose between high-capacity and high-performance SAS disk drives and various memory (DRAM and flash) options to optimize for their performance and capacity needs while staying within budget. Advanced data services such as remote replication and snapshot cloning are also available.



Data protocols

- Fibre channel
- iSCSI
- Infiniband over IP/RDMA
- iSER
- SRP
- NFS V3 and V4
- CIFS
- HTTP
- WebDAV
- FTP/SFTP/FTPS
- ZFS NDMP V4

Data services

- Oracle Hybrid Columnar Compression
- Hybrid storage pool
- Single, double and triple parity RAID (RAIDZ, Z2, Z3)
- Mirroring and triple mirroring
- End-to-end data integrity
- Local and Remote replication*
- Snapshots and clones*
- Quota(s)
- In-line dedup
- Compression
- Thin provisioning
- Antivirus via ICAP protocol
- Online data migration
- Clustering

Management

- Browser and CLI interface
- Management dashboard
- Hardware/component view
- Role-based access control
- Phone home
- Event and threshold based alerting
- Dtrace analytics
- Scripting
- Workflow automation
- Advanced networking
- DFS root support
- Source aware routing

* Remote Replication and Cloning features are licensed separately. All other features included with purchase of system.

Figure 20. Data services and management features of the Sun ZFS Storage Appliance.

The Sun ZFS Storage 7320 appliance, shown in Figure 21, offers a high-availability active-active cluster option with scalability of up to 432 TB of disk storage capacity, and it can be configured with up to 4 TB of read-optimized flash cache, along with up to 1.2 TB write-optimized flash cache for enhanced application performance.

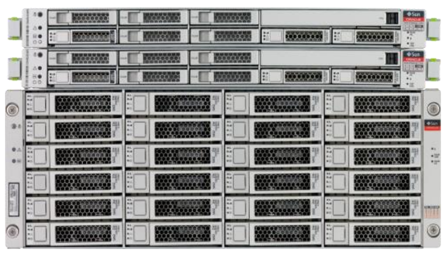


Figure 21: Sun ZFS Storage 7320 appliance.

How to Accelerate Test and Development Through Rapid Cloning of Production Databases and Operating Environments
March 2013

Authors: Roger Bitar, Ritu Kamboj

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com