Hewlett Packard
Enterprise

Helion OpenStack Carrier Grade 4.0

SOFTWARE MANAGEMENT

16.10

Titanium Server

Cloud Administration, 16.10

Contents

1

System Backup and Restore

SystemBackup

System Data Backup with Controller Storage

A system data backup of a Titanium Server system with controller storage captures core system information needed to restore a fully operational Titanium Server cluster.

System data backups include:

- platform configuration details
- system databases
- patching and package repositories
- home directory for the **wrsroot** user and all LDAP user accounts. See *Titanium Server System Administration: Linux User Accounts* for additional information.

System data backups do not include:

- Modifications manually made to the file systems, such as configuration changes on the **/etc** directory. After a restore operation has been completed, these modifications have to be reapplied.
- Home directories and passwords of local user accounts. They must be backed up manually by the system administrator.
- The **/root** directory. Use the **wrsroot** account instead when root access is needed.

Procedure

1. Log in as user **wrsroot** to the active controller.

   You can log in directly on the console or remotely using **ssh**.

2. Execute the system backup.

   ```
   $ sudo config_controller --backup titanium_backup_20140918
   Performing backup (this will take several minutes):
   Step 13 of 13 [###########################################] [100%]
   System backup file created: /opt/backups/titanium_backup_20140918_system.tgz
   Images backup file created: /opt/backups/titanium_backup_20140918_images.tgz

   Backup complete
   ```

   In this example, the argument *titanium_backup_20140918* to the **--backup** option is an arbitrary identifier you want to use for the backup.

   Upon successful execution of the command, the following two files are available on the controller's file system:

   - **/opt/backups/titanium_backup_20140918_system.tgz**
   - **/opt/backups/titanium_backup_20140918_images.tgz**

   Together, these two files represent the entire system data.

3. On a Ceph system, export the image content.

   This step applies only to systems that use Ceph storage (systems with storage nodes).

   To perform this step, you must become the Keystone admin user.

   a)
   ```
   ~(keystone_admin)$ glance image-list

   +--------------------------------------+------------+
   | ID                                   | Name       |
   +--------------------------------------+------------+
   | 3b0cbb67-f794-4115-86e2-82b2276acb9c | cust-guest |
   | 63aa7396-0587-4ee0-9a26-56c212d583d0 | cirros     |
   +--------------------------------------+------------+
   ```

   b) Switch to the **root** account, and then export the content.

   NOTE: You must be logged in as root to run the **image-backup** command.

   ```
   $ sudo su -
   $ image-backup export 63aa7396-0587-4ee0-9a26-56c212d583d0
   Exporting image: 100% complete...done.
   Exporting image: 100% complete...done.
   Creating backup archive...done
   Backup archive /opt/backups/image_63aa7396-0587-4ee0-9a26-56c212d583d0.tgz
   created
   ```

4. Transfer the backup files to an external storage resource.

   You can use a command such as **scp** to transfer the backup files to a server reachable over the OAM network. You can also copy them to a locally attached storage device, such as an external USB drive.

5. Optional: Delete the backup files.

You may want to free up disk space on the controller to accommodate upcoming Cinder volume backups.

The system data backup is now complete. The backup files must be kept in a secured location, probably holding multiple copies of them for redundancy purposes.

Cinder Volume Backups

Cinder volume backups are executed selectively, one volume at a time. They can include virtual disk images and any other Cinder volumes available in the system.

Consider the following recommendations when executing Cinder volume backups.

Aim at backing up an idle system

In an idle system, there are no running virtual machines. All VMs are either shut down or terminated. This is the ideal state for ensuring that the current content of all volumes is captured for replication at the next restore operation. Volumes reported as *available* are not associated with any VMs, while volumes reported as *in-use* are associated only with shut-down VMs, and are not being actively modified.

In a non-idle system, at least one VM is running. Volumes reported as *in-use* may be subject to write operations. In this case, you can create snapshots and back them up, but there is a risk of changes after the snapshot that are not captured.

Instructions to back up Cinder volumes in *available* and *in-use* states are provided in the following sections.

Back up one volume at a time

Backing up Cinder volumes can be a time- and space- consuming process, as determined by the size of the volumes themselves. Having two or more backup processes running simultaneously is likely to result in a severe performance hit to the controller and the running instances. This not only slows the normal system operations, but also makes the backup operations themselves take much longer.

Clean up as soon as possible

Backup files for Cinder volumes can fill up the **/opt/backups** partition on the controller very quickly. This is likely to cause further backup operations to fail in the middle of the execution. Therefore, as soon as a Cinder volume is backed up, and its backup files transferred and securely stored elsewhere:

- remove any snapshots you may have created during the backup process
- remove the backup files themselves

Backing Up Available Cinder Volumes

You can back up Cinder volumes in the *available* state.

A Cinder volume in the *available* state is the ideal candidate for a backup operation in that there are no running instances using it.

Procedure

1. Log in as Keystone user **admin** to the active controller.

Log in as user **wrsroot** to the active controller and source the script **/etc/nova/openrc** to obtain administrative privileges. For more information, see *Titanium Server System Administration: Linux User Accounts*.

2. Identify the volume you want to back up.
   a) List Cinder volumes from all tenants.

```
~(keystone_admin)$ cinder list --all-tenants
+--------------+-----------+-----------------+------+-------------+----------
|   ID         |   Status  |   Display Name  | Size | Volume Type | Bootable
+--------------+-----------+-----------------+------+-------------+----------
| 593111d8-... | in-use    | volume-1        |  1   |     None    |   true
| 53ad007a-... | available | volume-2        |  10  |     None    |   false
+--------------+-----------+-----------------+------+-------------+----------
+---------------+
| Attached to   |
+---------------+
|  b998107b-... |
|               |
+---------------+
```

   Cinder volumes are listed also on the Volumes page of the web administration interface.

   b) Identify the volume to back up.

   In the list above, **volume-1** is a disk image associated with a running instance. The volume **volume-2**, with UUID **53ad007a-...**, is available and can be safely backed up.

3. Export the target volume to the controller's file system.

```
~(keystone_admin)$ cinder export 53ad007a-841a-4fd3-a22b-813d4a6a8a85
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
| display_description |                                      |
|          id         | 53ad007a-841a-4fd3-a22b-813d4a6a8a85 |
|         size        |                  1                   |
|        status       |              exporting               |
|      updated_at     |      2014-09-18T21:36:15.247881      |
|     volume_type     |                None                  |
+---------------------+--------------------------------------+
```

   Exporting the volume takes some time, longer times for larger volumes. While it is taking place, the status of the volume is set to *exporting*, which you can verify by issuing the command **cinder show** *UUID*, or **cinder list --all-tenants** again. This new state is automatically reported on the Web administration interface.

   You must wait for the export operation to complete. Use the command **cinder show 53ad007a-...** and look for the **backup_status** field to determine whether the export operation was successful. The state of the volume should be set to *available* again.

   When the export operation completes, the backup file is ready, as illustrated in the following example:

```
~(keystone_admin)$ ls -lh /opt/backups
total 306M
drwx------ 2 root root  16K Sep 10 17:42 lost+found
drwxr-xr-x 2 root root 4.0K Sep 18 21:42 temp
-rw-r--r-- 1 root root 305M Sep 18 21:42 volume-53ad007a-841a-4fd3-
a22b-813d4a6a8a85-20140918-213918.tgz
```

4. Transfer the backup file to an external storage resource.

The following example uses the **scp** command to transfer the backup file to the directory **/backups/titanium** on a server named **backups-server.wrs.com**:

```
~(keystone_admin)$ scp /opt/backups/volume-53ad007a-841a-4fd3-
a22b-813d4a6a8a85-20140918-213918.tgz admin@backups-server.wrs.com:/backups/titanium
```

5. Delete the backup file.

   You may want to free up disk space on the controller to accommodate additional Cinder volume backups.

The volume backup is complete. Repeat this procedure with all other Cinder volumes in the *available* state in the system.

Backing Up In-Use Cinder Volumes

You can back up Cinder volumes in the *in-use* state.

A Cinder volume in the *in-use* state is associated with an existing instance, either as the instance's disk image or as an attached storage resource. The instance can be either running or shut down. If the instance is running, you must take a snapshot of the volume first, and then export it to the controller's file system. An effective and reliable strategy is to take snapshots of all *in-use* volumes, regardless of whether the associated instances are running. If you prefer, you can use the **nova list** command with the option **--all-tenants** to identify which instances are running, and then take snapshots selectively.

```
~[keystone_admin]$ nova list --all-tenants

+--------+--------+-----------+--------+------------+-------------+----------+
| ID     | Name   | Tenant ID | Status | Task State | Power State | Networks
|
                  |
+--------+--------+-----------+--------+------------+-------------+----------+
| 1f6... | t1-vm1 | 5e38b9... | ACTIVE | -          | Running     | tenan... |
| b92... | t1-vm2 | 5e38b9... | ACTIVE | -          | Running     | tenan... |
+--------+--------+-----------+--------+------------+-------------+----------+
```

Procedure

1. Log in as Keystone user **admin** to the active controller.

   Log in as user **wrsroot** to the active controller and source the script **/etc/nova/openrc** to obtain administrative privileges as described in *Linux User Accounts*.

2. Identify the volume you want to back up.
   a) List Cinder volumes from all tenants.

```
~(keystone_admin)$ cinder list --all-tenants
+--------------+-----------+--------------+... +----------+--------------+
|      ID      |   Status  | Display Name |... | Bootable | Attached to  |
+--------------+-----------+--------------+... +----------+--------------+
| 53ad007a-... |   in-use  |   volume-1   |... |  false   | c820df55-... |
| 578da734-... |   in-use  |              |... |  true    | c820df55-... |
| 593111d8-... | available |   volume-2   |... |  true    |              |
+--------------+-----------+--------------+... +----------+--------------+
```

   Cinder volumes are also listed on the Volumes page of the Web administration interface.

   b) Identify the volume to back up.

In the list above, two volumes are listed in the *in-use* state:

- A volume named **volume-1** with ID **53ad007a-...**, attached to the running image **c820df55-...**. The bootable flag is *false*, which indicates that this is not a disk image.
- An unnamed volume with ID **578da734-...**, attached to the running image **c820df55-...**. The bootable flag is *true*, which indicates that this is the instance's disk image. Unnamed volumes are dynamically created when the option **Instance Boot Source** in the Launch Instance window of the Web administration interface is set to *Boot from image (creates a new volume)* or *Boot from volume snapshot (creates a new volume)*.

This example uses the unnamed disk image to illustrate the backup procedure, but you must back up all *in-use* volumes appropriately.

3. Create a volume snapshot.

```
~(keystone_admin)$ cinder snapshot-create --force True --display-name test-vm-
snapshot \
578da734-a416-47e7-97de-137aa10b90f8
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|      created_at     |      2014-09-19T14:18:47.269583      |
| display_description |                 None                 |
|     display_name    |           test-vm-snapshot           |
|          id         | 53b5237c-5cd0-4939-ad6b-cf612e449216 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|        status       |               creating               |
|      volume_id      | 578da734-a416-47e7-97de-137aa10b90f8 |
+---------------------+--------------------------------------+
```

The status of the new snapshot is *creating*, which means that it is in process. You must wait until it becomes *available* before proceeding.

You can verify the status of all Cinder snapshots using the following command:

```
~(keystone_admin)$ cinder snapshot-list --all-tenants
+--------------+--------------+-----------+------------------+------+
|    ID        | Volume ID    |   Status  |   Display Name   | Size |
+--------------+--------------+-----------+------------------+------+
| 53b5237c-... | 578da734-... | available | test-vm-snapshot |  1   |
+--------------+--------------+-----------+------------------+------+
```

In this example, the state of the snapshot **test-vm-snapshot** is *available* already.

4. Export the snapshot to the controller's file system.

```
~(keystone_admin)$ cinder snapshot-export test-vm-snapshot
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
| display_description |                 None                 |
|          id         | 53b5237c-5cd0-4939-ad6b-cf612e449216 |
|        status       |              exporting               |
|      updated_at     |      2014-09-19T14:18:47.393220      |
|     volume_size     |                  1                   |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

The status of the snapshot is *exporting*, which means that it is not yet ready on the controller's file system. As before, you can use the **cinder snapshot-show** *UUID* command to verify the current status, and wait until the snapshot's state is *available* again.

When the export operation completes, the backup file is ready, as illustrated in the following example:

```
~(keystone_admin)$ ls -lh /opt/backups
total 258M
drwx------ 2 root root  16K Sep 10 17:42 lost+found
drwxr-xr-x 2 root root 4.0K Sep 19 14:30 temp
-rw-r--r-- 1 root root 258M Sep 19 14:30 volume-578da734-
a416-47e7-97de-137aa10b90f8-20140919-142732.tgz
```

Note that the UUID used in the file name is the volume's and not the snapshot's.

5. Transfer the backup file to an external storage resource.

   The following example uses the **scp** command to transfer the backup file to a server named **backups-server.wrs.com** in the directory **/backups/titanium**:

   ```
   ~(keystone_admin)$ scp /opt/backups/volume-578da734-
   a416-47e7-97de-137aa10b90f8-20140919-142732.tgz \
   admin@backups-server.wrs.com:/backups/titanium
   ```

6. Delete the snapshot.

   ```
   ~(keystone_admin)$ cinder snapshot-delete test-vm-snapshot
   ```

   You can confirm that the snapshot is deleted using **cinder snapshot-list --all-tenants**.

   If a standard deletion fails, you can force deletion using the following command:

   ```
   ~(keystone_admin)$ cinder snapshot-force-delete snapshot-id [snapshot-id]
   ```

   Once the snapshot is exported, you should remove it from the Cinder repositories to free up disk space.

7. Delete the backup file.

   You may want to free up disk space on the controller to accommodate additional Cinder volume backups.

The volume backup is complete. Repeat this procedure with all other Cinder volumes in the *in-use* state in the system.

System Restore

Performing a System Restore

You can restore a Titanium Server cluster from available system data and Cinder volume backup files to bring it back to the operational state it was when the backup procedure took place.

Restoring a Titanium Server cluster from a set of backup files is done by restoring one host at a time, starting with the controllers, then the storage nodes, and finally the compute nodes.

Prerequisites

Before you start the restore procedure you must ensure the following conditions are in place:

- All cluster hosts must be prepared for network boot and then powered down. You can prepare a host for network boot by erasing its disk boot image.
- All backup files are accessible from a USB flash drive locally attached to the controller where the restore operation takes place (**controller-0**).
- You have the original Titanium Server installation image available on a USB flash drive. It is mandatory that you use the exact same version of the software used during the original installation, otherwise the restore procedure will fail.
- The restore procedure requires all hosts but **controller-0** to boot over the internal management network using the PXE protocol. Ideally, the old boot images are no longer present, so that the hosts boot from the network when powered on. If this is not the case, you must configure each host manually for network boot immediately after powering it on.

Procedure

1. Install the Titanium Server software on **controller-0** from the USB flash drive.

   Refer to the *Titanium Server Software Installation Guide* for details.

   When the software installation is complete, you should be able to log in using the host's console and the web administration interface.

2. Log in to the console as user **wrsroot** with password **wrsroot**.

3. Ensure the backup files are available to the controller.

   Plug the USB flash drive containing the backup files into the system. The USB flash drive is mounted automatically. Use the command **df** to list the mount points.

   The following steps assume that the backup files are available from a USB flash drive mounted in **/media/wrsroot** in the directory **backups**.

4. Update the controller's software to the previous patching level.

   The current software version on the controller is compared against the version available in the backup files. If the backed-up version includes patches, the restore process automatically applies the patches and forces an additional reboot of the controller to make them effective.

   The following is the command output if patching is necessary:

```
$ sudo config_controller --restore-system \
/media/wrsroot/backups/titanium_backup_20140918_system.tgz
Restoring system (this will take several minutes):
Step  4 of 19 [#########                                  ] [21%]
This controller has been patched. A reboot is required.
After the reboot is complete, re-execute the restore command.
Enter 'reboot' to reboot controller:
```

   You must enter **reboot** at the prompt as requested. Once the controller is back, log in as user **wrsroot** as before to continue.

   After the reboot, you can verify that the patches were applied, as illustrated in the following example:

```
$ sudo sw-patch query
        Patch ID          Repo State  Patch State
========================  ==========  ===========
COMPUTECONFIG             Available       n/a
```

```
LIBCUNIT_CONTROLLER_ONLY   Applied        n/a
STORAGECONFIG              Applied        n/a
```

5. Restore the system configuration.

   With the controller software installed and patched to the same level that was in effect when the backup was performed, you can perform the restore procedure without interruption.

```
$ sudo config_controller --restore-system \
/media/wrsroot/backups/titanium_backup_20140918_system.tgz
Restoring system (this will take several minutes):
Step 19 of 19 [##########################] [100%]
Restoring node states (this will take several minutes):

Locking nodes:
Powering-off nodes:
Step 0 of 0 [##############################################] [100%]

System restore complete
```

6. Authenticate to the system as Keystone user **admin**.

   Source the **admin** user environment as follows:

```
$ cd; source /etc/nova/openrc
```

7. Restore the system images.

```
~(keystone_admin)$ sudo config_controller --restore-images \
/media/wrsroot/backups/titanium_backup_20140918_images.tgz
Step 2 of 2 [##############################################] [100%]

Images restore complete
```

   This step assumes that the backup file resides on the attached USB drive. If instead you copied the image backup file to the **/opt/backups** directory, for example using the **scp** command, you can remove it now.

8. Restore **controller-1**.

   a) List the current state of the hosts.

```
~(keystone_admin)$ system host-list
+----+--------------+-------------+----------------+-------------+--------------+
| id | hostname     | personality | administrative | operational | availability |
+----+--------------+-------------+----------------+-------------+--------------+
| 1  | controller-0 | controller  | unlocked       | enabled     | available    |
| 2  | controller-1 | controller  | locked         | disabled    | offline      |
| 3  | compute-0    | compute     | locked         | disabled    | offline      |
| 4  | storage-0    | storage     | locked         | disabled    | offline      |
| 5  | storage-1    | storage     | locked         | disabled    | offline      |
+----+--------------+-------------+----------------+-------------+--------------+
```

   b) Power on the host.

      Ensure that the host boots from the network, and not from any disk image that may be present.

      The software is installed on the host, and then the host is rebooted. Wait for the host to be reported as **Locked**, **Disabled**, and **Online**.

   c) Unlock the host.

```
~(keystone_admin)$ system host-unlock 2
+-----------------+--------------------------------------+
| Property        | Value                                |
```

```
+-----------------+--------------------------------------+
| action          | none                                 |
| administrative  | locked                               |
| availability    | online                               |
| ...             | ...                                  |
| uuid            | 5fc4904a-d7f0-42f0-991d-0c00b4b74ed0 |
+-----------------+--------------------------------------+
```

d) Verify the new state of the hosts.

```
~(keystone_admin)$ system host-list
+----+--------------+-------------+----------------+-------------+--------------+
| id | hostname     | personality | administrative | operational | availability |
+----+--------------+-------------+----------------+-------------+--------------+
| 1  | controller-0 | controller  | unlocked       | enabled     | available    |
| 2  | controller-1 | controller  | unlocked       | enabled     | available    |
| 3  | compute-0    | compute     | locked         | disabled    | offline      |
| 4  | storage-0    | storage     | locked         | disabled    | offline      |
| 5  | storage-1    | storage     | locked         | disabled    | offline      |
+----+--------------+-------------+----------------+-------------+--------------+
```

The unlocking operation forces a reboot of the host, which is then initialized with the corresponding image available from the system backup.

You must wait for the host to become enabled and available before proceeding to the next step.

9. Restore the storage nodes.

You need to restore the storage nodes if you are using the Cinder Ceph backend. Follow the same procedure used to restore **controller-1**, first restoring host **storage-0** and then **storage-1**.

The state of the hosts when the restore operation is complete is as follows:

```
~(keystone_admin)$ system host-list
+----+--------------+-------------+----------------+-------------+--------------+
| id | hostname     | personality | administrative | operational | availability |
+----+--------------+-------------+----------------+-------------+--------------+
| 1  | controller-0 | controller  | unlocked       | enabled     | available    |
| 2  | controller-1 | controller  | unlocked       | enabled     | available    |
| 3  | compute-0    | compute     | locked         | disabled    | offline      |
| 4  | storage-0    | storage     | unlocked       | enabled     | available    |
| 5  | storage-1    | storage     | unlocked       | enabled     | available    |
+----+--------------+-------------+----------------+-------------+--------------+
```

10. On systems with Ceph storage, switch to the **root** account, and then restore the image content.

This step is required only for systems using Ceph storage (systems with storage nodes).

NOTE: You must be logged in as root to run the **image-backup** command.

```
$ sudo su -
$ image-backup import image_63aa7396-0587-4ee0-9a26-56c212d583d0.tgz

Extracting files...done
Importing image: 100% complete...done.
Importing image: 100% complete...done.
```

11. Restore Cinder volumes.

You restore Cinder volumes by importing them from the backup files. You must import all volume backup files, one at a time.

a) Delete any snapshots that may exist in the system.

The restore operation fails if a snapshot exists for a volume that is about to be restored. To list available snapshots, use the following command:

```
~(keystone_admin)$ cinder snapshot-list --all-tenants
```

To remove snapshots, use the following command:

```
~(keystone_admin)$ cinder snapshot-delete snapshot-id [snapshot-id]
```

If a standard deletion fails, you can force deletion using the following command:

```
~(keystone_admin)$ cinder snapshot-force-delete snapshot-id [snapshot-id]
```

b) List the current state of the Cinder volumes.

```
~(keystone_admin)$ cinder list --all-tenants
+--------------+-----------+--------------+... +----------+--------------+
|      ID      |   Status  | Display Name |... | Bootable | Attached to  |
+--------------+-----------+--------------+... +----------+--------------+
| 53ad007a-... |   error   |   volume-1   |... |  false   | c820df55-... |
| 578da734-... |   error   |              |... |  true    | c820df55-... |
| 593111d8-... |   error   |   volume-2   |... |  true    |              |
+--------------+-----------+--------------+... +----------+--------------+
```

All volumes are reported to be in the *error* state because they are not available yet, but they are registered in the storage database.

c) Copy the Cinder volumes to the **/opt/backups** folder.

```
~(keystone_admin)$ sudo cp /media/wrsroot/backups/volume-* /opt/backups
```

d) Change to the backups directory, and then import a volume.

```
~(keystone_admin)$ cd /opt/backups
~(keystone_admin)$ cinder import volume-53ad007a-841a-4fd3-a22b-813d4a6a8a85.tgz
+---------------------+----------------------------------------+
|       Property      |                Value                   |
+---------------------+----------------------------------------+
| display_description |                 None                   |
|          id         | 53ad007a-841a-4fd3-a22b-813d4a6a8a85 |
|         size        |                  1                     |
|        status       |              importing                 |
|      updated_at     |      2014-09-19T16:10:24.791220        |
|     volume_type     |                 None                   |
+---------------------+----------------------------------------+
```

The volume remains in the *importing* state while the operation takes place. Use the **backup_status** field on the output of the **cinder show** command to monitor the progress. You must wait until the original volume's state, *in-use* or *available*, is restored. Note that *in-use* volumes are fully restored even if their corresponding instances are not running yet.

e) Import all other volumes in the backup files.

Once finished, all volumes are listed in their original states, as illustrated below.

```
~(keystone_admin)$ cinder list --all-tenants
+--------------+-----------+--------------+... +----------+--------------+
|      ID      |   Status  | Display Name |... | Bootable | Attached to  |
+--------------+-----------+--------------+... +----------+--------------+
| 53ad007a-... |   in-use  |   volume-1   |... |  false   | c820df55-... |
| 578da734-... |   in-use  |              |... |  true    | c820df55-... |
| 593111d8-... | available |   volume-2   |... |  true    |              |
+--------------+-----------+--------------+... +----------+--------------+
```

12. Remove any *in-use* volumes that remain in error.

You must remove all *in-use* volumes that for any reason failed to recover, and their associated virtual machine instances. This is necessary to prevent errors from occurring when restoring the compute nodes used to launch the virtual machines. If an *in-use* volume is in error at the time its virtual machine is launched, the Nova scheduler reports an error and the restore operation fails.

For the purposes of this example, assume that volume **53ad007a-...** failed to restore. Its status is then reported as **error**.

a) Find the associated virtual machine instance.

The ID of the associated instance is available from the output of the **cinder list --all-tenants** command in the previous step. In this case it is **c820df55-...**.

b) Remove the virtual machine instance.

```
~(keystone_admin)$ nova delete c820df55-...
```

c) Remove the volume in error status.

```
~(keystone_admin)$ cinder delete 53ad007a-...
```

13. Restore the compute nodes, one at a time.

You restore these hosts following the same procedure used to restore **controller-1**.

The state of the hosts when the restore operation is complete is as follows:

```
~(keystone_admin)$ system host-list
+----+--------------+-------------+----------------+-------------+--------------+
| id | hostname     | personality | administrative | operational | availability |
+----+--------------+-------------+----------------+-------------+--------------+
| 1  | controller-0 | controller  | unlocked       | enabled     | available    |
| 2  | controller-1 | controller  | unlocked       | enabled     | available    |
| 3  | compute-0    | compute     | unlocked       | enabledd    | available    |
| 4  | storage-0    | storage     | unlocked       | enabled     | available    |
| 5  | storage-1    | storage     | unlocked       | enabled     | available    |
+----+--------------+-------------+----------------+-------------+--------------+
```

As each compute node is restored, the original instances at the time the backups were done are started automatically.

The Titanium Server cluster is fully restored. The state of the system, including storage resources and virtual machines, is identical to the state the cluster was in when the backup procedure took place.

Postrequisites

Passwords for local user accounts must be restored manually since they are not included as part of the backup and restore procedures. See *Titanium Server System Administration: Linux User Accounts* for additional information.

Titanium Server CPE Systems

Performing a System Data Backup for Titanium Server CPE

A system data backup captures core system information needed to restore a fully operational Titanium Server CEP configuration.

System data backups include:

- platform configuration details
- system databases
- patching and package repositories
- home directory for the **wrsroot** user and all LDAP user accounts. See *Titanium Server System Administration: Linux User Accounts* for additional information.

System data backups do not include:

- Modifications manually made to the file systems, such as configuration changes on the **/etc** directory. After a restore operation has been completed, these modifications have to be reapplied.
- Home directories and passwords of local user accounts. They must be backed up manually by the system administrator.
- The **/root** directory. Use the **wrsroot** account instead when root access is needed.

Procedure

1. Log in as user **wrsroot** to the active controller.

   You can log in directly on the console or remotely using **ssh**.

2. Execute the system backup.

   ```
   $ sudo config_controller --backup titanium_backup_20140918
   Performing backup (this will take several minutes):
   Step 13 of 13 [###############################################] [100%]
   System backup file created: /opt/backups/titanium_backup_20140918_system.tgz
   Images backup file created: /opt/backups/titanium_backup_20140918_images.tgz

   Backup complete
   ```

   In this example, the argument *titanium_backup_20140918* to the **--backup** option is an arbitrary identifier you want to use for the backup.

   Upon successful execution of the command, the following two files are available on the controller's file system:

   - **/opt/backups/titanium_backup_20140918_system.tgz**
   - **/opt/backups/titanium_backup_20140918_images.tgz**

   Together, these two files represent the entire system data.

3. Transfer the backup files to an external storage resource.

   You can use a command such as **scp** to transfer the backup files to a server reachable over the OAM network. You can also copy them to a locally attached storage device, such as an external USB drive.

4. Optional: Delete the backup files.

   You may want to free up disk space on the controller to accommodate upcoming Cinder volume backups.

The system data backup is now complete. The backup files must be kept in a secured location, probably holding multiple copies of them for redundancy purposes.

2

Patches and Upgrades

Patches

You apply patches between Titanium Server software upgrades to provide updates to the code.

A patch typically modifies a small portion of the code to address the following items:

- bugs
- security vulnerabilities
- feature enhancements

Patches can be installed manually or by the Patch Orchestrator which automates a rolling install of a patch across all of the Titanium Server hosts.

The Titanium Server handles multiple patches being applied and removed at once.

Patches can modify and update any area of Titanium Server software, including the kernel itself.

For information on populating, installing and removing patches, see Managing Software Patches on page 17.

In-Service Patches

In-Service Patching provides a mechanism to issue patches that do not require a reboot, allowing the patch to be installed on in-service nodes and restarting affected processes as needed.

Depending on the area of software being patched and the type of software change, installation of the patch may or may not require the Titanium Server hosts to be rebooted. Patches are classified as reboot required or reboot not required (also refered to as in-service) type patches to indicate this.

Upgrades

You apply Titanium Server software upgrades to move Titanium Server software between major releases of Titanium Server software.

Software upgrades move Titanium Server software from one release to the next and change the version of updated software components. The upgrade typically updates components that may include the kernel, Operating System packages, OpenStack, and Titanium Server specific software. The software upgrade process manages several complexities such as conversion of database schemas, conversion of database data, and API compatibility management between Titanium Server Hosts.

For information on upgrading the Titanium Server software, see Overview of Titanium Server Upgrades on page 39.

3

Patching

Managing Software Patches

Patches to the system software become available as needed to address issues associated with a current Titanium Server software release. They must be uploaded to the active controller and applied to all required hosts in the cluster.

The following elements form part of the patching environment:

Reboot-Required Patches

Reboot-required patches are typically major patches that require hosts to be locked during the patching process and rebooted to complete the process. Note that when a Titanium Server host is locked and rebooted for patching, the VMs hosted by that Titanium Server host are live migrated to an alternate host in order minimize the impact to the VMs. Also note that if the VM cannot be live migrated due to the type of resources it is using, then the VM will be cold migrated to an alternate host.

In-Service Patches

In-service, or reboot not required, patches are patches that do not require the locking and rebooting of hosts. The required Titanium Server software is updated and any required Titanium Server processes are re-started. Hosted VMs are completely unaffected.

Patching Commands

The **sw-patch** command is available on both active controllers. It must be run as root using **sudo**. It provides the user interface to process the patches, including querying the state of a patch, listing affected hosts, and applying, installing, and removing patches.

Patch Storage Area

A central storage area maintained by the patch controller. Patches are initially uploaded to the patch storage area and remain there until they are deleted.

Software Updates Repository

A central repository of software updates associated with patches applied to the system. This repository is used by all hosts in the cluster to identify the software updates and rollbacks required on each host.

Patching Logs

The following logs are used to record patching activity:

patching.log

This records patch agent activity on each host.

patching-api.log

This records user actions that involve patching, performed using either the CLI or the REST API.

The overall flow for installing a patch from the command line interface on a working Titanium Server cluster is the following:

1. Download the patch from the Wind River servers to a workstation that can reach the active controller through the OAM network.

   Consult the Wind River support personnel for details on the availability of new patches.

2. Copy the patch to the active controller using the cluster's OAM floating IP address as the destination point.

   You can use a command such as **scp** to copy the patch. The patching workflows presented in this document assume that this step is complete already, that is, they assume that the patch is already available on the file system of the active controller.

3. Upload the new patch to the patching storage area.

   This step makes the new patch available within the patching system, but does not install it to the cluster yet. For all purposes, the patch is dormant.

4. Apply the patch.

   This step adds the patch to the Software Updates Repository, making it visible to all hosts in the cluster.

5. Install the patch on each of the affected hosts in the cluster.

Patching the system can be done using the web administration interface or the command line interface on the active controller. When using the web administration interface you upload the patch directly from your workstation using a file browser window provided by the patch upload facility.

A special case occurs during the initial provisioning of a cluster, when you want to patch **controller-0** before the system software is configured. This can only be done from the command line interface. See Installing Patches Before Initial Commissioning on page 22 for details.

Identifying the Software Version and Patch Level

You can view the current software version and patch level from the web administration interface. The system type is also shown.

NOTE: The system type (Standard or CPE) is selected at installation.

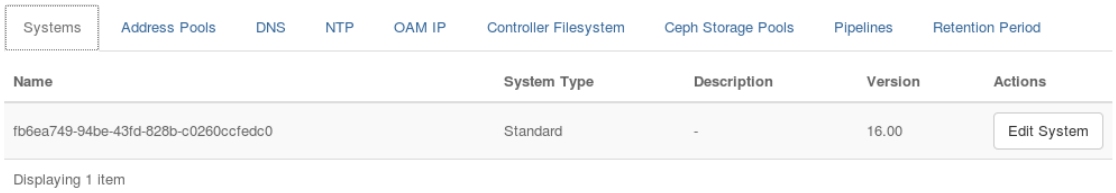
Procedure

1. In the Titanium Server Web administration interface, open the System Configuration page.

   The System Configuration page is available from **Admin** > **Platform** > **System Configuration** in the left-hand pane.

2. Select the **Systems** tab to view the software version.

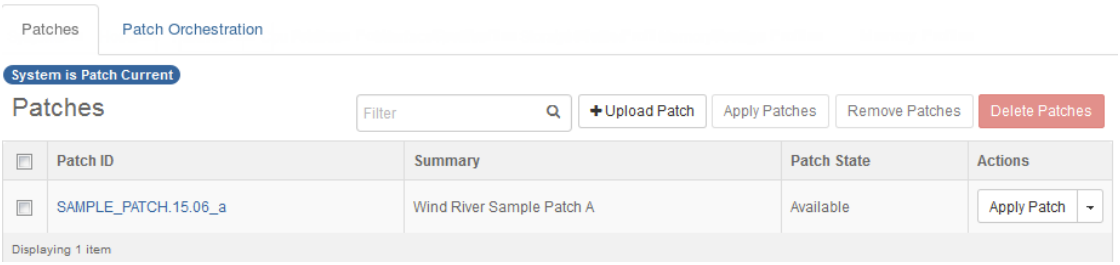
   The software version is shown in the **Version** field. The type of system is shown in the **System Type** field.



3. In the Titanium Server Web administration interface, open the Software Management page.

   The Software Management page is available from **Admin** > **Platform** > **Software Management** in the left-hand pane.

4. Select the **Patches** tab to view patch information.



Identifying the Software Version and Patch Level Using the CLI

You can view the the current software version and patch level from the CLI. The system type is also shown.

To find the software version from the CLI, use the **system show** command.

```
~(keystone_admin)$ system show
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| name             | 57409bbc-c85c-4dbf-9eeb-49ee925c4cdf |
| system_type      | Standard                             |
| description      | None                                 |
| location         | None                                 |
```

```
| contact          | None                                 |
| software_version | 16.00                                |
| uuid             | c6977e1e-49da-4ed2-8c3d-7ec201d45613 |
| created_at       | 2016-04-13T21:30:38.042086+00:00     |
| updated_at       | 2016-04-13T21:30:40.429541+00:00     |
+------------------+--------------------------------------+
```

To list applied patches from the CLI, use the **sw-patch query** command.

```
~(keystone_admin)$ sudo sw-patch query

   Patch ID                Release  Patch State
========================  =======  ===========
SAMPLE_PATCH_15.06_a       15.12     Available
```

For more about working with patches, see Managing Software Patches on page 17

Populating the Patch Storage Area

Patches have to be uploaded to the Titanium Server patch storage area before they can be applied.

Procedure

1. Log in as **wrsroot** to the active controller.
2. Upload the patch file to the patch storage area.

   ```
   $ sudo sw-patch upload /tmp/patches/TITANIUM_CONTROLLER_14.10_PATCH_0001.patch
   TITANIUM_CONTROLLER_15.12_PATCH_0001 is now available
   ```

   This example uploads a single patch to the storage area. You can specify multiple patch files on the same command separating their names with spaces.

   Alternatively, you can upload all patch files stored in a directory using a single command, as illustrated in the following example:

   ```
   $ sudo sw-patch upload-dir /tmp/patches
   ```

   The patch files are available now in the patch storage area, but they have not yet been applied to the cluster.

3. Verify the status of the patch.

   ```
   $ sudo sw-patch query

      Patch ID                Release   Patch State
   ==============             =======   ===========
   TITANIUM_15.12_PATCH_0001   15.12     Available
   ```

   The patch state is *Available* now, indicating that it is included in the patch storage area. Further details about the patch can be retrieved as follows:

   ```
   ~(keystone_admin)$ sudo sw-patch show TITANIUM_15.12_PATCH_0001
   TITANIUM_15.12_PATCH_0001:
       Patch State:    Available
       Release Status: Released
       Summary:        Add logs to hbsClient and libalarm-clock
       Description:    Modified log in hbsClient. Expect to see this log on
                       all blade types  'hbsClient patches by
                       TITANIUM_15.12_PATCH_0001' when hbsClient
   ```

```
                restarts. Also added a log to ac_init() function
                that reads "TITANIUM_15.12_PATCH_0001 patched
                libalarm-clock".  Should be seen when heartbeat
                process is restarted on controller and compute.
    Install Instructions:
                install text here
    Requires:
                TITANIUM_15.12_PATCH_0001
    Contents:
                titanium-mtce-common-1.0-r23.x86_64.rpm
                titanium-mtce-common-dev-1.0-r23.x86_64.rpm
                titanium-mtce-common-pmon-1.0-r23.x86_64.rpm
                titanium-mtce-common-rmon-1.0-r23.x86_64.rpm
                libalarm-clock-1.0-r1.x86_64.rpm
```

4. Delete a patch from the patch storage area.

   Patches in the *Available* state can be deleted as illustrated in the following command:

   ```
   $ sudo sw-patch delete TITANIUM_15.12_PATCH_0001
   ```

   The patch to delete from the patch storage area is identified by the patch ID reported by the **sw-patch query** command. You can provide multiple patch IDs to the delete command separated by spaces.

Patch Status and Lifecycle

After adding a patch to the patch storage area, you must move it to the software updates repository, which manages patch distribution for the cluster. From there, you can install patches to the hosts that require them.

Some of the available patches may be required on controller hosts only, while others may be required on compute or storage hosts. When you *apply* an available patch, it is added to the software updates repository, ready for installation on any host type that requires it. You can then install all applicable patches to each host in turn.

To keep track of patch installation, you can use the **sw-patch query** command.

```
~(keystone_admin)$ sudo sw-patch query

   Patch ID                Release  Patch State
========================  =======  ===========
TITANIUM_15.12_PATCH_0001   15.12     Applied
TITANIUM_15.12_PATCH_0002   15.12     Available
TITANIUM_15.12_PATCH_0003   15.12     Available
```

This shows the **Patch State** for each of the patches in the patch storage area:

Available

A patch in the *Available* state has been added to the patch storage area, but has not yet been added to the software updates repository or installed on the hosts.

Partial-Apply

A patch in the *Partial-Apply* state has been added to the software updates repository using the **sw-patch apply** command, but has not been installed on all hosts that require it. It may have been installed on some but not others, or it may not have been installed on any hosts. Patches that require a reboot can not be installed on an unlocked node. In-service patches can be installed on unlocked nodes.

Applied

A patch in the *Applied* state has been installed on all hosts that require it.

You can use the **sw-patch query-hosts** command to see which hosts are fully patched (**Patch Current**). This also shows which hosts require reboot, either because they are not fully patched, or because they are fully patched but not yet rebooted.

```
~(keystone_admin)$ sudo sw-patch query-hosts

  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.12       Yes             Yes         15.12   idle
controller-0  192.168.204.3        No              Yes         15.12   idle
controller-1  192.168.204.4        Yes             Yes         15.12   idle
```

Installing Patches Before Initial Commissioning

When initially installing the Titanium Server software, it is recommended that you install the latest available patches on **controller-0** before running the **config_controller** script, and before installing the software on other hosts. This ensures that:

- The software on **controller-0**, and all other hosts, is up to date when the cluster comes alive.
- You reduce installation time by avoiding patching the system right after an out-of-date software installation is complete.

This exercise assumes that the patches to install are available on a USB flash drive, or from a server reachable by **controller-0**.

Procedure

1. Initialize **controller-0**.

   Use the Titanium Server bootable ISO image to initialize **controller-0**. See the *Titanium Server Software Installation Guide* for details.

   This step takes you to the point where you use the console port to log in to **controller-0** as user **wrsroot**.

2. Populate the patch storage area.

   Upload the patches from the USB flash drive using the command **sw-patch upload** or **sw-patch upload-dir** as described in Populating the Patch Storage Area on page 20.

3. Apply the patches.

   Apply the patches using the command **sw-patch apply --all**.

   The patches are now in the software updates repository, ready to be installed.

4. Install the patches on the controller.

   ```
   $ sudo sw-patch install-local
   Patch installation is complete.
   Please reboot before continuing with configuration.
   ```

   This command installs all applied patches on **controller-0**.

5. Reboot **controller-0**.

You must reboot the controller to ensure that it is running with the software fully patched.

```
$ sudo reboot
```

6. Continue the installation.

   Log in again as **wrsroot** and continue with the software installation by running the **config_controller** script. See the *Titanium Server Software Installation Guide* for further details and directions.

Once all hosts in the cluster are initialized, they are all running fully patched software. The Titanium Server cluster is up to date.

Installing Reboot-Required Patches Using the Web Administration Interface

You can use the web administration interface to upload, delete, apply, and remove patches.

This section presents an example patching workflow using a single patch. The main steps of the procedure are:

- Upload the patch(es).
- Lock the host(s).
- Install patches; any unlocked nodes will reject the request.
- Unlock the host(s). Unlocking the host(s) automatically triggers a reboot.

Procedure

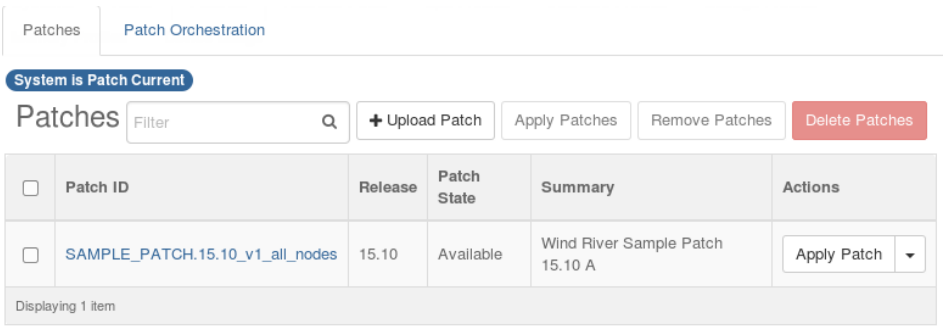
1. Log in to the web administration interface as the **admin** user.
2. In the Titanium Server Web administration interface, open the Software Management on page.

   The Software Management page is available from **Admin** > **Platform** > **Software Management** in the left-hand pane.
3. Select the Patches page to see the current patch status.

   The Patches page shows the current status of all patches uploaded to the system. If there are no patches, an empty Patch Table is displayed.
4. Upload the patch file to the patch storage area.

   Click the **Upload Patch** button to display an upload window from which you can browse your workstation's file system to select the patch file. Click the **Upload Patch** button once the selection is done.

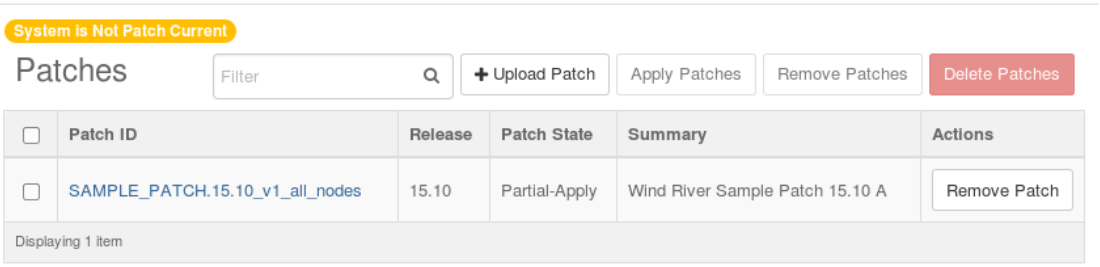
   The patch file is transferred to the Active Controller and is copied to the patch storage area, but it has yet to be applied to the cluster. This is reflected in the Patches page as illustrated below.



5. Apply the patch.

   Click the **Apply Patch** button associated with the patch. Alternatively, select the patch first using the selection boxes on the left, and then click the **Apply Patches** button at the top. You can use this selection process to apply all patches, or a selected subset, in a single operation.

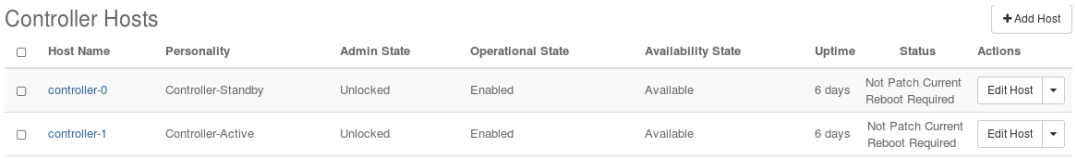
   The Patches page is updated to report the patch to be in the *Partial-Apply* state.



6. Install the patch on **controller-0**.

   a) Select the **Hosts** tab.

      The **Hosts** tab on the Host Inventory page reflects the new status of the hosts with respect to the new patch state. As shown below, both controllers are now reported as not patch current and requiring reboot.



   b) Transfer active services to the standby controller by selecting the **Swact Host** option from the **Edit Host** button associated with the active controller host.

      NOTE: Access to the web administration interface may be lost briefly during the active controller transition. You may have to log in again.

   c) Select the **Lock Host** option from the **Edit Host** button associated with **controller-0**.

   d) Next, select the **Install Patches** option from the **Edit Host** button associated with **controller-0** to install the patch.

      A confirmation window is presented giving you a last opportunity to cancel the operation before proceeding.

   e) Select the **Unlock Host** option from the **Edit Host** button associated with controller-0.

7. Repeat the steps above with **controller-1** to install the patch on **controller-1**.

8. Verify the state of the patch.

   Visit the Patches page again. The patch is now in the *Applied* state.

The patch is applied now, and all affected hosts have been updated.

Patches can be removed using the **Remove Patches** button from the Patches page. The workflow is similar to the one presented in this section, with the exception that patches are being removed from each host instead of being applied.

Installing Reboot-Required Patches Using the CLI

You can install reboot-required patches using the CLI.

Procedure

1. Log in as Keystone user **admin** to the active controller.

   Log in as user **wrsroot** to the active controller and source the script **/etc/nova/openrc** to obtain administrative privileges as described in *Linux User Accounts*.

2. Verify that the patches are available using the **sw-patch query** command.

```
~(keystone_admin)$ sudo sw-patch query
       Patch ID             Patch State
========================   ===========
TITANIUM_15.12_PATCH_0001   Available
TITANIUM_15.12_PATCH_0002   Available
TITANIUM_15.12_PATCH_0003   Available
```

3. Apply the patch.

```
~(keystone_admin)$ sudo sw-patch apply TITANIUM_15.12_PATCH_0001
TITANIUM_15.12_PATCH_0001 is now in the repo
```

   The patch is now in the *Partial-Apply* state, ready for installation from the software updates repository on the impacted hosts.

4. Optional: Apply all available patches in a single operation.

```
~(keystone_admin)$ sudo sw-patch apply --all
TITANIUM_15.12_PATCH_0001 is now in the repo
TITANIUM_15.12_PATCH_0002 is now in the repo
TITANIUM_15.12_PATCH_0003 is now in the repo
```

   In this example, there are three patches ready for installation from the software updates repository.

5. Query the patching status of all hosts in the cluster.

   You can query the patching status of all hosts at any time as illustrated below. Note that the reported status is the accumulated result of all applied and removed patches in the software updates repository, and not just the status due to a particular patch.

```
~(keystone_admin)$ sudo sw-patch query-hosts

  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.12       Yes             No          15.12   idle
```

```
controller-0  192.168.204.3        No             Yes         15.12   idle
controller-1  192.168.204.4        No             Yes         15.12   idle
```

For each host in the cluster, the following patch status fields are displayed:

Patch Current

Indicates whether there are patches pending for installation or removal on the host or not. If *Yes*, then all relevant patches in the software updates repository have been installed on, or removed from, the host already. If *No*, then there is at least one patch in either the *Partial-Apply* or *Partial-Remove* state that has not been applied to the host.

Reboot Required

Indicates whether the host must be rebooted or not as a result of one or more patches that have been either applied or removed, or because it is not patch current.

Release

Indicates the running software release version.

State

There are four possible states:

- idle - In a wait state.
- installing - Installing (or removing) patches.
- install-failed - The operation failed, either due to a patch error or something killed the process. Check the **patching.log** on the node in question.
- install-rejected - The node is unlocked, therefore the request to install has been rejected. This state persists until there is another install request, or the node is reset.

Once the state has gone back to idle, the install operation is complete and you can safely unlock the node.

In this example, **compute-0** is up to date, no patches need to be installed and no reboot is required. By contrast, the controllers are not patch current, and therefore a reboot is required to install the patch.

6. Install all pending patches on **controller-0**.
   a) Switch the active controller services.

      ```
      ~(keystone_admin)$ system host-swact controller-0
      ```

      Before patching a controller node, you must transfer any active services running on the host to the other controller. Only then it is safe to lock the host.

   b) Lock the host.

      You must lock the target host, controller, compute, or storage, before installing patches.

      ```
      ~(keystone_admin)$ system host-lock controller-0
      ```

   c) Install the patch.

      ```
      ~(keystone_admin)$ sudo sw-patch host-install controller-0
      ```

NOTE: You can use the **sudo sw-patch host-install-async** *hostname* command if you are launching multiple installs in parallel.

d) Unlock the host.

```
~(keystone_admin)$ system host-unlock controller-0
```

Unlocking the host forces a reset of the host followed by a reboot. This ensures that the host is restarted in a known state.

All patches are now installed on **controller-0**. Querying the current patch status displays the following information:

```
~(keystone_admin)$ sudo sw-patch query-hosts

  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.12       Yes             No          15.12   idle
controller-0  192.168.204.3        No              No          15.12   idle
controller-1  192.168.204.4        No              Yes         15.12   idle
```

7. Install all pending patches on **controller-1**.

   Repeat the previous step targeting **controller-1**.

   All patches are now installed on **controller-1** as well. Querying the current patching status displays the following information:

```
~(keystone_admin)$ sudo sw-patch query-hosts

  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.12       Yes             No          15.12   idle
controller-0  192.168.204.3        Yes             No          15.12   idle
controller-1  192.168.204.4        Yes             No          15.12   idle
```

8. Install any pending patches for the compute or storage hosts.

NOTE: All VMs currently running on a compute host are *live-migrated* to another host. If this is not possible, the lock command fails. The Patch User can determine the reason for the failure and decide how to proceed. For example:

- There may not be enough resources for the VMs on this host to live-migrate to another host, in which case:
  - you can remove any VMs on other hosts that are no longer required, or
  - you can install more hosts.
- There may be some VMs on the host being locked that cannot support live-migration due to their current setup, in which case you you can cold-migrate such VMs

If the **Patch Current** status for a compute or storage host is **No**, apply the pending patches using the following commands:

```
~(keystone_admin)$ system host-lock hostname

~(keystone_admin)$ sudo sw-patch host-install-async hostname

~(keystone_admin)$ system host-unlock hostname
```

where *hostname* is the name of the host (for example, **compute-0**).

NOTE: Patch installations can be triggered in parallel.

The **sw-patch host-install-async** command (**install patches** on the Web administration interface) can be run on all locked nodes, without waiting for one node to complete the install before triggering the install on the next. If you can lock the nodes at the same time, you can patch them at the same time. For maximum efficiency, you can migrate VMs to lock as many compute nodes at the same time as resources allow and patch at once, thereby reducing the overall time required to install a patch through the whole system.

Likewise, you can install a patch to the standby controller and a compute node at the same time. The only restrictions are those of the lock: You cannot lock both controllers, and you cannot lock a compute node if you do not have enough free resources to migrate the VMs off it. Also, in a CEPH configuration (with storage nodes), you cannot lock more than one of controller-0/controller-1/storage-0 at the same time, as these nodes are running CEPH monitors and you must have at least two in service at all times.

9. Confirm that all patches are installed and the Titanium Server cluster is up-to-date.

   Use the **sw-patch query** command to verify that all patches are **Applied**.

   ```
   ~(keystone_admin)$ sudo sw-patch query
          Patch ID             Patch State
   ========================    ===========
   TITANIUM_15.12_PATCH_0001   Applied
   ```

   If the **Patch State** for any patch is still shown as **Available** or **Partial-Apply**, use the **sw-patch query-hosts** command to identify which hosts are not **Patch Current**, and then apply patches to them as described in the preceding steps.

The Titanium cluster is up to date now. All patches are installed.

Installing In-Service Patches Using the Web Administration Interface

The procedure for applying an in-service patch is similar to that as a reboot-required patch, except that the host does not need to be locked and unlocked as part of applying the patch.

Procedure

1. Log in to the web administration interface as the **admin** user.
2. In the Titanium Server Web administration interface, open the Software Management page.

   The Software Management page is available from **Admin** > **Platform** > **Software Management** in the left-hand pane.

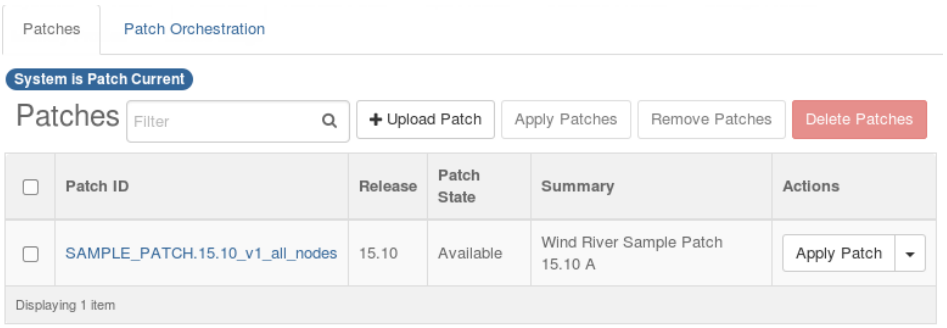
3. Select the Patches page to see the current patch status.

   The Patches page shows the current status of all patches uploaded to the system. If there are no patches, an Patch Table is displayed.

4. Upload the patch file to the patch storage area.

   Click the **Upload Patch** button to display an upload window from which you can browse your workstation's file system to select the patch file. Click the **Upload Patch** button once the selection is done.

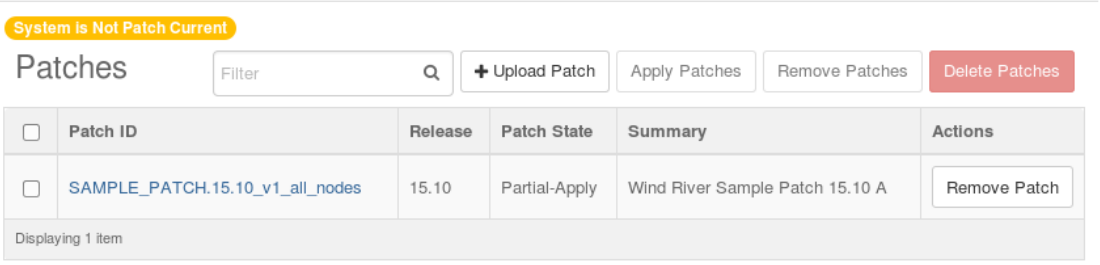
   The patch file is transferred to the Active Controller and is copied to the patch storage area, but it has yet to be applied to the cluster. This is reflected in the Patches page as illustrated below.



5. Apply the patch.

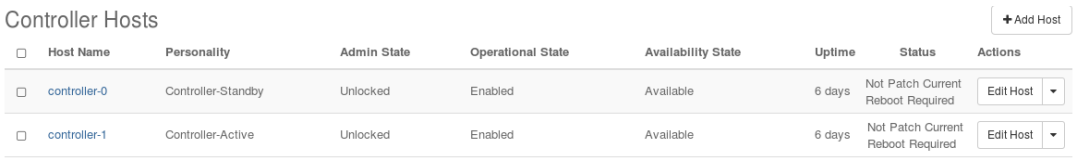
   Click the **Apply Patch** button associated with the patch. Alternatively, select the patch first using the selection boxes on the left, and then click the **Apply Patches** button at the top. You can use this selection process to apply all patches, or a selected subset, in a single operation.

   The Patches page is updated to report the patch to be in the *Partial-Apply* state.



6. Install the patch on **controller-0**.
   a) Select the **Hosts** tab.

      The **Hosts** tab on the Host Inventory page reflects the new status of the hosts with respect to the new patch state. In this example, the patch only applies to controller software, as can be seen by the compute host's status field being empty, indicating that it is patch current.



b) Next, select the **Install Patches** option from the **Edit Host** button associated with **controller-0** to install the patch.

A confirmation window is presented giving you a last opportunity to cancel the operation before proceeding.

7. Repeat the steps above with **controller-1** to install the patch on **controller-1**.

8. Verify the state of the patch.

Visit the Patches page again. The patch is now in the *Applied* state.

The patch is applied now, and all affected hosts have been updated.

Installing In-Service Patches Using the CLI

The procedure for applying an in-service patch is similar to that as a reboot-required patch, except that the host does not need to be locked and unlocked as part of applying the patch.

Procedure

1. Upload the patch.

```
$ sudo sw-patch upload INSVC_HORIZON_SYSINV.patch

INSVC_HORIZON_SYSINV is now available
```

2. Confirm that the patch is available.

```
$ sudo sw-patch query
      Patch ID        RR  Release  Patch State
====================  ==  =======  ===========
INSVC_HORIZON_SYSINV  N    16.00    Available
```

3. Check the status of the hosts.

```
$ sudo sw-patch query-hosts
  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.24       Yes             No          16.00   idle
controller-0  192.168.204.3        Yes             No          16.00   idle
controller-1  192.168.204.4        Yes             No          16.00   idle
```

4. Apply the patch

```
$ sudo sw-patch apply INSVC_HORIZON_SYSINV
INSVC_HORIZON_SYSINV is now in the repo
```

The patch state transitions to Partial-Apply:

```
$ sudo sw-patch query
      Patch ID        RR  Release   Patch State
```

```
====================  ==  =======  =============
INSVC_HORIZON_SYSINV  N    16.00   Partial-Apply
```

Because it's an in-service patch, the hosts report that they are not patch current, but they do not require a reboot.

```
$ sudo sw-patch query-hosts
  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.24       No              No          16.00   idle
controller-0  192.168.204.3        No              No          16.00   idle
controller-1  192.168.204.4        No              No          16.00   idle
```

5. Install the patch on controller-1.

```
$ sudo sw-patch host-install controller-1
.............
Installation was successful.
```

6. Query the hosts to to check status.

```
$ sudo sw-patch query-hosts
  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.24       No              No          16.00   idle
controller-0  192.168.204.3        No              No          16.00   idle
controller-1  192.168.204.4        Yes             No          16.00   idle
```

The controller-1 host reports it is now patch current and does not require a reboot, without having been locked or rebooted

7. Install the patch on compute-0.

```
$ sudo sw-patch host-install compute-0
....
Installation was successful.
```

8. Install the patch on controller-0.

```
~(keystone_admin)]$ sudo sw-patch host-install controller-0
............
Installation was successful.
```

You can query the hosts to confirm that all nodes are now patch current, and that the patch has transitioned to the Applied state.

```
$ sudo sw-patch query-hosts
  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.24       Yes             No          16.00   idle
controller-0  192.168.204.3        Yes             No          16.00   idle
controller-1  192.168.204.4        Yes             No          16.00   idle

$ sudo sw-patch query
      Patch ID        RR  Release  Patch State
====================  ==  =======  ===========
INSVC_HORIZON_SYSINV  N    16.00     Applied
```

Removing Reboot-Required Patches

Patches in the *Applied* or *Partial-Apply* states can be removed if necessary, for example, when they trigger undesired or unplanned effects on the Titanium cluster.

Rolling back patches is conceptually identical to installing patches. A roll-back operation can be commanded for a patch in either the *Applied* or the *Partial-Apply* states. As the patch is removed, it goes through the following state transitions:

Applied or *Partial-Apply* to *Partial-Remove*

A patch in the *Partial-Remove* state indicates that it has been removed from zero or more, but not from all, the applicable hosts.

Use the command **sw-patch remove** to trigger this transition.

Partial-Remove to *Available*

Use the command **sudo sw-patch host-install-async** *hostname* repeatedly targeting each time one of the applicable hosts in the cluster. The transition to the *Available* state is complete when the patch is removed from all target hosts. The patch remains in the patch storage area as if it had just been uploaded.

The following example is basically a copy of the workflow used in Patch Status and Lifecycle on page 21, but this time removing instead of installing the patch. The steps are therefore presented in simplified form. Removing patches can be done using the Web Administration Interface, also, as discussed in Installing Reboot-Required Patches Using the Web Administration Interface on page 23.

Procedure

1. Log in as Keystone user **admin** to the active controller.
2. Verify the state of the patch.

```
~(keystone_admin)$ sudo sw-patch query
       Patch ID             Patch State
========================    ===========
TITANIUM_15.12_PATCH_0001   Applied
```

   In this example the patch is listed in the *Applied* state, but it could be in the *Partial-Apply* state as well.

3. Remove the patch.

```
~(keystone_admin)$ sudo sw-patch remove TITANIUM_15.12_PATCH_0001
TITANIUM_15.12_PATCH_0001 has been removed from the repo
```

   The patch is now in the *Partial-Remove* state, ready to be removed from the impacted hosts where it was already installed.

4. Query the patching status of all hosts in the cluster.

```
~(keystone_admin)$ sudo sw-patch query-hosts

  Hostname      IP Address    Patch Current  Reboot Required  Release  State
============  ==============  =============  ===============  =======  =====
compute-0     192.168.204.12       Yes             No          15.12   idle
controller-0  192.168.204.3        No              Yes         15.12   idle
controller-1  192.168.204.4        No              Yes         15.12   idle
```

In this example, the controllers have patches ready to be removed, and therefore must be rebooted.

5. Remove all pending-for-removal patches from **controller-0**.
   a) Swact controller services.
   b) Run the patching sequence.

```
~(keystone_admin)$ sudo sw-patch host-install-async controller-0
```

6. Remove all pending patches from **controller-1**.

The Titanium cluster is up to date now. All patches have been removed, and the patch TITANIUM_15.12_PATCH_0001 can be deleted from the patch storage area if necessary.

It is important to note the role of the command **sudo sw-patch host-install-async** *hostname* as a tool that both installs and removes patches as necessary.

Patch Orchestration Overview

Titanium Server supports patch orchestration, which allows an entire Titanium Server system to be patched with a single operation.

Patch orchestration is configured and run through the CLI, the Horizon GUI, or the VIM REST API.

Patch orchestration will automatically iterate through all nodes of the system and install the applied patch(es) to each node: first the Controller Nodes, then the Storage Nodes and finally, the Compute Nodes. During the patching of Compute Nodes, the migration of VMs off of Compute Nodes being patched is managed automatically by Patch Orchestration. The Controller Nodes are always patched in serial; however, the Storage Nodes and Compute Nodes can configurably be patched in parallel in order to reduce the overall time of installing the patch.

Patch orchestration can install one or more applied patches at the same time, and can install Reboot-Required Patches and/or In-Service Patches at the same time. Patch orchestration will only lock/unlock (that is, reboot) nodes to install a patch if at least one Reboot-Required patch has been applied.

The user first creates a patch orchestration strategy, or plan, for the automated patching procedure which contains a number of parameters for customizing the particular behavior of the patching orchestration. This includes parameters to specify which Node types should be patched, serial or parallel patching of nodes, and default action for moving VMs off of the Compute Nodes being patched. Based on these parameters, the state of Nodes and Server Group definitions, patch orchestration creates a number of stages for the overall patch strategy; where each stage generally consists of migrating VMs, locking node(s), installing patch(es) and unlocking node(s) for some subset of the overall nodes.

After creating the patch orchestration strategy, the user can either apply the entire strategy to install the patch across the entire system automatically, or apply an individual stage at a time to control and monitor its progress manually.

For reboot-required patching, there are two modes for moving VMs off of the Compute Nodes being patched. First, start-stop, where instances are stopped before a host is patched; typically,

used for VMs that do not support migration or migration takes too long a time. In order to ensure this does not impact the high level service being provided by the VM, the VM should be protected and grouped into an anti-affinity server group with its standby VM. Second, migrate, where instances are either live migrated or cold migrated off a host before the host is patched. Currently, only start-stop mode supports parallel Compute Node patch installation.

Patch orchestration can only be done on a system that meets the following conditions:

- The system is clear of alarms (with the exception of alarms for locked hosts, stopped instances, and patch applications in progress).
- All hosts must be unlocked-enabled-available.
- All instances must either be enabled or shut down. No instances that are members of a server group can be shut down.
- Two controller nodes must be available.
- All storage nodes must be available.
- When installing reboot required patches with VM behavior of **migrate**, there must be spare compute capacity to migrate VMs off of the compute node(s) being patched.

Configuring Patch Orchestration

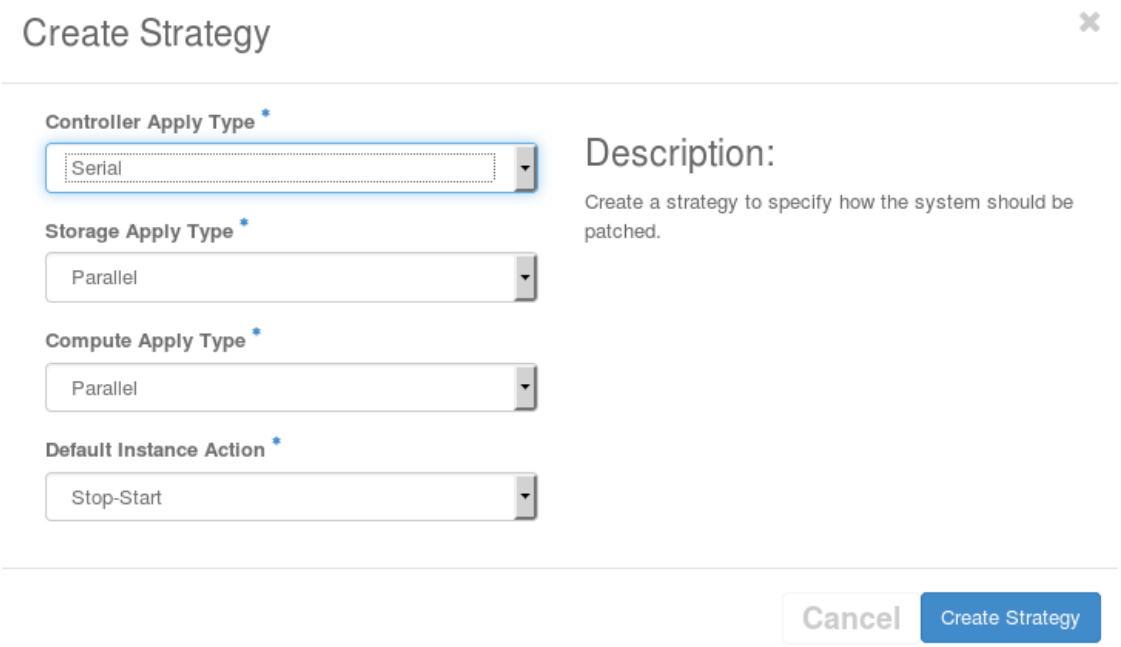
You can configure patch orchestration using the web administration interface.

The patch orchestration interface is found in the web administration interface on the Patch Orchestration tab, available from **Admin** > **Platform** > **Software Management** in the left-hand pane.

Procedure

1. Upload and apply your patches as described in Managing Software Patches on page 17 (do not use **host-install** to install the patches on any hosts).
2. Select **Platform** > **Software Management**, then select the **Patch Orchestration** tab.
3. Click the **Create Strategy** button.

   The Create Strategy dialog appears.



4. Create a patch strategy by specifying settings for the parameters in the Create Strategy dialog box.
   - **controller-apply-type**:
     - serial (default): controllers will be patched one at a time (standby controller first)
     - ignore: controllers will not be patched
   - **storage-apply-type**:
     - serial (default): storage hosts will be patched one at a time
     - parallel: storage hosts will be patched in parallel, ensuring that only one storage node in each redundancy group is patched at a time.
     - ignore: storage hosts will not be patched
   - **compute-apply-type**:
     - serial (default): compute hosts will be patched one at a time
     - parallel: compute hosts will be patched in parallel, ensuring that:
       - At most half of the hosts in a host aggregate will be patched at the same time.
       - For instances in server groups with anti-affinity policies (including best effort), only a single instance in the server group will be impacted at a time.
       - At most, 10 compute hosts will be patched at the same time.

NOTE: Warning: Using the **stop-start** option will result in an outage for each instance, as it is stopped while the compute host is locked/patched/unlocked. In order to ensure this does not impact service, instances MUST be grouped into anti-affinity (or anti-affinity best effort) server groups, which will ensure that only a single instance in each server group is stopped at a time.

  - ignore: compute hosts will not be patched

- **default-instance-action**:
  - stop-start (default): instances will be stopped before a host is patched (applies to reboot patching only)

NOTE: Warning: Using the **stop-start** option will result in an outage for each instance, as it is stopped while the compute host is locked/patched/unlocked. In order to ensure this does not impact service, instances MUST be grouped into anti-affinity (or anti-affinity best effort) server groups, which will ensure that only a single instance in each server group is stopped at a time.

  - migrate: instances will be migrated off a host before it is patched (applies to reboot patching only).

NOTE: This option can only be used with the serial **compute-apply-type**.

5. Click **Create Strategy** to save the patch orchestration strategy.

   Examine the patch strategy. Pay careful attention to:

   - The sets of hosts that will be patched together in each stage.
   - The sets of instances that will be impacted in each stage.

   The patch strategy has one or more stages, with each stage consisting of one or more hosts to be patched at the same time. Each stage is split into steps (for example, **query-alarms**, **lock-hosts**, **sw-patch-hosts**). Note the following about stages:

   - Controller hosts are patched first, followed by storage hosts and then compute hosts.
   - Compute hosts with no instances are patched before compute hosts with instances.
   - The final step in each stage is "system-stabilize," which waits for a period of time (up to several minutes) and ensures that the system is free of alarms. This ensures that the patch orchestrator does not continue to patch more hosts if the patch application has caused an issue resulting in an alarm.

6. Click the **Apply Strategy** button to apply the patch-strategy. You can optionally apply a single stage at a time by clicking the **Apply Stage** button.

   When applying a single stage, you can only apply the next stage; you cannot skip stages.

7. Optional: To abort the patch, click the **Abort Strategy** button.
   - While a patch-strategy is being applied, it can be aborted. This results in:
     - The current step will be allowed to complete.
     - If necessary an abort phase will be created and applied, which will attempt to:
       - unlock any hosts that were locked
       - start any instances that were stopped

NOTE: If a patch-strategy is aborted after hosts were locked, but before they were patched, the hosts will not be unlocked, as this would result in the patches being installed. You must either install the patches on the hosts or remove the patches before unlocking the hosts and restarting any instances that had been stopped by patch orchestration.

8. Delete the patch-strategy.

   After a patch strategy has been applied (or aborted) it must be deleted before another patch strategy can be created. If a patch strategy application fails, you must address the issue that caused the failure, then delete and re-create the strategy before attempting to apply it again.

Patch Orchestration CLI

The patch orchestration CLI is **sw-manager**.

The commands and options map directly to the parameter descriptions in the web interface dialog, described in Configuring Patch Orchestration on page 34.

There is help available for the overall command and also for each sub-command.

```
# sw-manager patch-strategy --help
usage: sw-manager patch-strategy [-h]  ...

optional arguments:
  -h, --help  show this help message and exit
```

Patch orchestration commands include:

- **create** - Create a strategy
- **delete** - Delete a strategy
- **apply**- Apply a strategy
- **abort**- Abort a strategy
- **show**- Show a strategy

4

Upgrades

Overview of Titanium Server Upgrades

Titanium Server upgrades enable you to move Titanium software from one release of Titanium Server to the next release of Titanium Server.

Titanium Server software upgrade is a manual multi-step rolling-upgrade process, where Titanium Server hosts are upgraded one at time while continuing to provide its hosting services to its hosted applications.

Before starting the upgrades process, the system must be “patch current”, there must be no alarms present on the system, the new software load must be imported and a valid license file for the upgrade must be installed.

The upgrades process starts by upgrading the controllers. The standby controller is upgraded first and involves loading the standby controller with the new release of software and migrating all the Controller Services’ databases for the new release of software. Activity is switched to the upgraded controller, running in a ‘compatibility’ mode where all inter-node messages are using message formats from the old release of software. Before upgrading the second controller is the point-of-no-return for an in-service abort of the upgrades process, see the following section. The second controller is loaded with the new release of software and becomes the new Standby controller.

If present, Storage Nodes are locked, upgraded and unlocked one at a time in order to respect the 1:1 redundancy model of Titanium Server Storage Nodes.

Then Compute Nodes are locked, upgraded and unlocked one at a time in order to respect any Service Group rules that may exist for VMs hosted by the Compute Nodes. When locking a

Compute Node, VMs are either live migrated, if possible, to another Compute Node or cold migrated to another Compute Node. When upgrading the Compute Node, the Compute Node network boots/installs the new software from the Active Controller. After unlocking the Compute Node, the Compute Services are running in a 'compatibility' mode where all inter-node messages are using message formats from the old release of software.

The final steps of the upgrades process is to activate and complete the upgrade process. This basically involves disabling 'compatibility' modes on all hosts and clearing the Upgrade Alarm.

Rolling Back/Aborting an Upgrade

There are two cases for aborting an upgrade:

- Before controller-0 has been upgraded (that is, only controller-1 has been upgraded): In this case the upgrade can be aborted and the system will remain in service during the abort. For more information, see Rolling Back a Software Upgrade Before the Second Controller Upgrade on page 46.
- After controller-0 has been upgraded (that is, both controllers have been upgraded): In this case the upgrade can only be aborted with a complete outage and a re-install of all hosts. This would only be done as a last resort, if there was absolutely no other way to recover the system. For more information, see Rolling Back a Software Upgrade After the Second Controller Upgrade on page 46.

On Titanium Server CPE Systems, after the upgraded controller-1 is active, but before controller-0 is upgraded, an inservice abort is possible, however, VMs that have been already migrated to the upgraded controller-1 will be lost, including their disks. These VMs are only recoverable if you have external backups for their data.

We do not support upgrade abort on CPE after controller-0 has been upgraded.

Preparing for Upgrade

You should complete these preparations before starting an upgrade.

- Apply patches to the currently running release to make sure that the system is patch current.
- Perform a full backup to allow recovery.
- Transfer the new release software load to controller-0 (or onto a USB stick); controller-0 must be active.
- Transfer the new release software license file to controller-0, (or onto a USB stick).
- Unlock all hosts.
  - All nodes must be unlocked. The upgrade cannot be started when there are locked nodes (the health check prevents it).
- Resolve all Active Alarms on the system.
  - Check the system health to determine if there are any alarms. Any alarms should be cleared.

```
$ system health-query controller-0
```

Clear any alarms in the system before beginning the upgrade.

Upgrading Titanium Server Software

You can upgrade a Titanium Server configuration with a new release of Titanium Server platform software.

Prerequisites

This procedure assumes that controller-0 is the active controller.

While the upgrade is in progress, Nova CLI commands must specify micro version 2.24; for example, **nova --os-compute-api-version 2.24**.

The microversion must be specified after the controllers have been upgraded, that is, when the controllers are running 16.10 but any of the computes are running 15.12. This is necessary if you decide to live migrate via the CLI.

NOTE: Live-migrate of instances from 16.10 to 15.12 compute nodes is not supported.

Procedure

1. Install the license file for the release you are upgrading to, for example, 16.00.

```
$ sudo license-install full_path_to_license_file
```

2. Run the **load-import** command on controller-0 to import the new release.

```
$ system load-import ~/bootimage.iso
+--------------------+-----------+
| Property           | Value     |
+--------------------+-----------+
| id                 | 2         |
| state              | importing |
| software_version   | 16.00     |
| compatible_version | 15.12     |
| required_patches   |           |
+--------------------+-----------+
```

   The **load-import** must be done on controller-0.

   You must provide the the full path to the ISO image.

3. Check to ensure the load was successfully imported.

```
$ system load-list

+----+----------+------------------+
| id | state    | software_version |
+----+----------+------------------+
| 1  | active   | 15.12            |
| 2  | imported | 16.00            |
+----+----------+------------------+
```

4. Apply any required patches.

   Typically, there may be a patch that is required to be applied prior to a software upgrade.

For more information, see Managing Software Patches on page 17.

5. Check the system health status for the upgrade to see if there are any system alarms.

```
$ system health-query-upgrade

System Health:
All hosts are unlocked/enabled: [OK]
All hosts have current configurations: [OK]
All hosts are patch current: [OK]
No alarms: [Fail]
[2] alarms found
Required patches are applied: [OK]
License valid for upgrade: [OK]
```

By default, the upgrade process cannot be run and is not recommended to be run with Active Alarms present. However, if the Alarms are of severity Minor or Warning, a **--force** option can be used on the **system upgrade-start** command to force the upgrade process to start, if you feel these alarms will not be an issue over the upgrade process. This cannot be done for Alarms of severity Major or Critical.

NOTE: It is strongly recommended that you clear your system of any and all alarms before doing an upgrade. While the **-force** option is available to run the upgrade, it is a best practice to clear any alarms.

6. Run the **upgrade-start** command on controller-0.

```
$ system upgrade-start
```

NOTE: You must ensure that controller-0 has been locked/unlocked at least once (not the active controller) before starting the upgrade.

The **upgrade-start** must be done on controller-0.

As part of the upgrade, the upgrade process checks the health of the system and validates that the system is ready for an upgrade.

The upgrade process checks that no alarms are active before starting an upgrade. (Minor and warning alarms can be ignored with the command **system upgrade-start -force**).

NOTE: Use the command **system upgrade-start --force** to force the upgrades process to start even if Alarms of severity Minor and/or Warning exist. This should ONLY be done if you feel these alarms will not be an issue over the upgrades process.

On systems with Ceph storage, it also checks that the Ceph cluster is healthy.

For CPE systems, it ensures that no instances are running on controller-1.

After **upgrade-start** is executed and until controller-0 is upgraded, no VM migrations are permitted. CPE upgrades allow VM migration after controller-1 is active.

This will make a copy of the system data to be used in the upgrade. Configuration changes are not allowed after this point, until the upgrade is completed.

7. Run the **host-upgrade-list** command to view the current status of each node.

```
$ system host-upgrade-list

+----+--------------+-------------+-----------------+----------------+
| id | hostname     | personality | running_release | target_release |
```

```
+----+--------------+-------------+-----------------+----------------+
| 1  | controller-0 | controller  | 15.12           | 15.12          |
| 2  | controller-1 | controller  | 15.12           | 15.12          |
| 3  | compute-0    | compute     | 15.12           | 15.12          |
| 4  | compute-1    | compute     | 15.12           | 15.12          |
+----+--------------+-------------+-----------------+----------------+
```

8. Lock the standby controller, controller-1.

```
$ system host-lock controller-1
```

9. Upgrade the standby controller, controller-1.

```
$system host-upgrade controller-1
```

NOTE:

Please note that branding tarballs created for previous releases of Titanium Server are not forward-compatible, and must be recreated to accommodate the latest branding styles present in Horizon.

If you have previously created any customized Horizon branding, then you need to do the following:

- After upgrading controller-1, copy the new branding tarball to **/opt/platform/config/16.10/branding/** on controller-0 and delete the existing tarball in that location.
- Horizon services will not start on controller-1 if there is prior branding applied and this step is skipped.
- For more information about custom branding, see the *Titanium Server Software Development Kit*.

10. Query the hosts' running release and target release status.

```
$ system upgrade-list

+----+--------------+-------------+-----------------+----------------+
| id | hostname     | personality | running_release | target_release |
+----+--------------+-------------+-----------------+----------------+
| 1  | controller-0 | controller  | 16.00           | 16.00          |
| 2  | controller-1 | controller  | 16.00           | 16.00          |
| 3  | compute-0    | compute     | 15.12           | 15.12          |
| 4  | compute-1    | compute     | 15.12           | 15.12          |
+----+--------------+-------------+-----------------+----------------+
```

11. Wait for data-migration-complete and controller-1 to come online.

```
$ system upgrade-show

+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| uuid         | cab7517b-51fc-4fe5-9493-ccc2664de78e |
| state        | data-migration-complete              |
| from_release | 15.12                                |
| to_release   | 16.00                                |
+--------------+--------------------------------------+
```

The upgrading of controller-1 and the data migration can take an hour or more.

12. Unlock standby controller-1.

```
$ system host-unlock controller-1
```

Wait for the **400.001** Services-related alarm to clear.

13. Swact to standby controller-1.

```
$ system host-swact controller-0
```

At this point controller-1 is running the new release, while the rest of the system is running the previous release.

After controller-1 is active on CPE systems, upgrade-abort requires restoring VM instances (complete outage).

14. Lock controller-0.

```
$ system host-lock controller-0
```

15. Upgrade controller-0.

NOTE: Executing this command is the "point of no return" for an inservice abort of the upgrades process. If you abort the upgrade procedure after this point, a full system re-installation will be required.

We do not support upgrade abort on CPE after controller-0 has been upgraded.

```
$ system host-upgrade controller-0
```

16. If using Ceph storage backend, upgrade the storage nodes one at a time.

Wait for all alarms to clear after the unlock before proceeding to the next storage host.

a) Lock storage-0.

```
$ system host-lock storage-0
```

b) Upgrade storage-0.

```
$ system host-upgrade storage-0
```

c) Unlock storage-0.

```
$ system host-unlock storage-0
```

d) Repeat the above steps for each storage host.

17. Upgrade compute hosts, one at a time.

Wait for all alarms to clear after the unlock before proceeding to the next compute host.

a) Lock compute-0.

```
$ system host-lock compute-0
```

b) Upgrade compute-0.

```
$ system host-upgrade compute-0
```

c) Unlock compute-0.

```
$ system host-unlock compute-0
```

d) Repeat the above steps for each compute host.

18. Activate the upgrade.

```
$ system upgrade-activate
```

During the running of the **upgrade-activate** command, new configurations are applied to the controller and compute hosts. **250.001** (*hostname* **Configuration is out-of-date**) alarms are raised for those nodes and are cleared as the configurations are applied. The upgrade state goes from **activating** to **activation-complete** once this is done.

19. Make controller-0 the active controller.

```
$ system host-swact controller-1
```

20. Complete the upgrade.

```
$ system upgrade-complete
```

21. Delete the previous, old release load.

```
$ system load-delete loadID
```

Overview of Upgrade Abort Procedure

You can abort an upgrade procedure if necessary.

There are two cases for aborting an upgrade:

- Before controller-0 has been upgraded (that is, only controller-1 has been upgraded): In this case the upgrade can be aborted and the system will remain in service during the abort.
- After controller-0 has been upgraded (that is, both controllers have been upgraded): In this case the upgrade can only be aborted with a complete outage and a re-install of all hosts. This would only be done as a last resort, if there was absolutely no other way to recover the system.

Rolling Back a Software Upgrade Before the Second Controller Upgrade

You can perform an inservice abort of an upgrade before the second Controller (controller-0 in the examples of this procedure) have been upgraded.

The upgrade is aborted as follows:

Procedure

1. Abort the upgrade with the **upgrade-abort** command.

   ```
   $ system upgrade-abort
   ```

   The upgrade state is set to aborting. Once this is executed, there is no cancelling; the upgrade must be completely aborted.

2. Make controller-0 active.

   ```
   $ system host-swact controller-1
   ```

   If controller-1 was active with the new upgrade release, swacting back to controller-0 will switch back to using the previous release databases, which were frozen at the time of the swact to controller-1. Any changes to the system that were made while controller-1 was active will be lost.

3. Lock and downgrade controller-1.

   ```
   $ system host-lock controller-1
   $ system host-downgrade controller-1
   ```

   The host is re-installed with the previous release load.

4. Unlock controller-1.

   ```
   $ system host-unlock controller-1
   ```

5. Complete the upgrade.

   ```
   $ system upgrade-complete
   ```

   This cleans up configuration, databases, and so forth, related to the aborted upgrade. The upgrade record is deleted since upgrade is no longer in progress.

6. Delete the newer upgrade release that has been aborted.

   ```
   $ system load-delete loadID
   ```

Rolling Back a Software Upgrade After the Second Controller Upgrade

After the second controller is upgraded, you can still roll back a software upgrade, however, the rollback will impact the hosting of applications.

The upgrade is aborted as follows:

Procedure

1. Run the **upgrade-abort** command to abort the upgrade.

   ```
   $ system upgrade-abort
   ```

   Once this is done there is no going back; the upgrade must be completely aborted.

2. Make controller-1 active.

   ```
   $ system host-swact controller-0
   ```

3. Lock controller-0.

   ```
   $ system host-lock controller-0
   ```

4. Wipe the disk and power down all storage (if applicable) and compute hosts.
   a) Execute **wipedisk** from the shell on each storage or compute host.
   b) Power down each host.

5. Lock all storage (if applicable) and compute hosts.

   ```
   $ system host-lock hostID
   ```

6. Downgrade controller-0.

   ```
   $ system host-downgrade controller-0
   ```

   The host is re-installed with the previous release load.

7. Unlock controller-0.

   ```
   $ system host-unlock controller-0
   ```

8. Swact to controller-0.

   ```
   $ system host-swact controller-1
   ```

   Swacting back to controller-0 will switch back to using the previous release databases, which were frozen at the time of the swact to controller-1. This is essentially the same result as a system restore.

9. Lock and downgrade controller-1.

   ```
   $ system host-downgrade controller-1
   ```

   The host is re-installed with the previous release load.

10. Unlock controller-1.

    ```
    $ system host-unlock controller-1
    ```

11. Power up and unlock the storage hosts one at a time (if using ceph storage backend). The hosts are re-installed with the release N load.

12. Restore glance images.

- For a system with controller storage (LVM storage backend), no work is required; the images should be available.
- For a system with dedicated storage (Ceph storage backend), follow the system restore documentation to restore the glance images using the **image-backup** script. If the glance images were not backed up prior to the upgrade, then the images must all be deleted and then re-created.

13. Restore cinder volumes.

- For a system with controller storage (LVM storage backend), no work is required; the images should be available.
- For a system with dedicated storage (Ceph storage backend), follow the system restore documentation to restore the glance images using the **image-backup** script. If the glance images were not backed up prior to the upgrade, then the images must all be deleted and then re-created.

14. Power up and unlock the compute hosts one at a time.

The hosts are re-installed with the previous release load. As each compute host goes online, instances will be automatically recovered by the system.

Note that instances using only Cinder volumes should be recoverable. Instances that were booted from a glance image will be rebuilt from the image; all data on local disks will be lost.

15. Complete the upgrade.

```
$ system upgrade-complete
```

This cleans up the upgrade release, configuration, databases, and so forth.

16. Delete the upgrade release load.

```
$ system load-delete
```