Monitoring

Application Control

Application control

The functioning system is comprised of applications. Each application has a name and a version number. An application may have multiple processes running within and each process has its own state.

Application Status

The command **app status** is used to display the status of the system. When the command is executed, it requests an up-to-date status of every process, and hence may take a few seconds to return.

A typical app status screen from the command line interface:

```
platform@development:~$ app status
voss-deviceapi v1.1.1 (2014-08-20 17:16)
     |-voss-queue running
     |-voss-notifications running
     |-voss-wsgi running
template_creator v0.0.0
template_runner v0.0.0
mongodb v1.1.1 (2014-08-20 17:30)
     |-arbiter running
     |-database running
support v1.1.1 (2014-08-20 17:17)
snmp v1.1.1 (2014-08-20 17:28)
     |-daemon running (completed)
     |-traps running (completed)
cluster v1.1.1 (2014-08-20 17:26)
platform v1.1.1 (2014-08-21 07:40)
nginx v1.1.1 (2014-08-20 17:27)
     |-proxy running
services v1.1.1 (2014-08-20 17:19)
     |-wsgi running
     |-logs running
     |-firewall running
     |-mount running
```

```
|-syslog running (completed)
|-time running (completed)
```

The following states are defined:

- `running` indicates that the process is running correctly.
- `completed` indicates that the process ran to completion successfully.
- `suspended` indicates that the process is suspended while waiting for another process.
- `stopped` indicates that the process is not running. An error message indicates that the process stopped for an unexpected reason.

Starting and Stopping

The system application may be stopped with **app stop** and restarted with **app start**. By default this is a non-blocking command, which means that the console prompt will be available after running this command while processes that are a part of it are running.

It is possible to start or stop individual applications and/or processes by appending the `<application-name>[:<process-name>]`. The list of applications can be seen by using the command **app status**.

For example, to stop the process `voss-queue`:

app start voss-queue

It is possible to perform a blocking start by including `blocking` after start but before the `<application-name>[:<process-name>]`. For example:

app start blocking

app start blocking voss-queue

This will ensure that all background processes that are started by **app start** will be completed before the console prompt is available.

Remote Execution in Clusters

Commands can be run on a remote node and the output displayed locally using:

cluster run <nodename>|<role>|all

For example, **cluster run database app start mongodb** will restart the mongodb service on all database nodes.

cluster run all backup create localbackup will perform a local backup of each node on its local drive.

Sometimes there are long-running processes running on a server. To display such jobs, use the **cluster job list** command. It is also possible to re-attach to those jobs to see the output, using **cluster job reconnect <pid>**.

Note When shutting down nodes in a cluster, this should be done individually on each node and not with the cluster run all command.

Notifications

Warnings and Notifications

On console login, the system displays a health report indicating the status of the system. This health report shows the following:

```
Last login: Tue Sep  3 10:19:07 2013 from 172.29.232.68
host: alan, role: standalone, load: 0.35, USERS: 3
date: 2013-09-03 10:20:02 +00:00, up: 2:05
network: 172.29.89.182, ntp: 172.29.1.15
SECURITY UPDATES: 136 security updates available
database: 8.0Gb
services: ok
```

The values in the report mean the following:

- last console login and IP address source
- the load average of the system
- the number of users currently logged in
- the system uptime
- the status of the system services
- whether security updates are available
- disk, CPU and memory warnings if applicable
- warnings are displayed in upper-case to draw attention

The report can be redisplayed by typing the command:

help

The system can be configured to forward warnings and notifications to a variety of destinations, including:

- local email
- remote email addresses
- remote SNMP destinations

Local email allows the administrator to view a list of warnings, and delete these as necessary.

SNMP queries

The server runs a SNMP server which can be queried remotely.

The following details identify the server:

- **snmp community <community string>**
- **snmp contact <system contact>**
- **snmp name <system name>**

- **snmp location <system location>**

SNMP CPU load notifications are set using:

snmp load <1min load> <5min load> <15min load>

which results in notifications being sent should the threshold be exceeded.

Incoming SNMP queries can be restricted to a single host for security purposes using:

snmp query <ip>

SNMP v2 can be set with:

snmp query snmp://<community string>@<ip>

SNMP v3 username and password can be set with:

snmp query snmp://user:auth:password@<ip>:<port>

Where:

- user: username for the SNMP3 trap server
- auth: the password (MD5 or SHA), with minimum length of 8 characters.
- password: the secondary password that is set for 'priv', with minimum length of 8 characters.

To verify what the version of snmp is currently in use, run **snmp list** and reference the `version` field.

The screen console output below is an example of the use of **snmp community <community string>:**

```
$ snmp list
     community: private7
     load1: 5
     load15: 3
     load5: 4
     query: snmp://private7@172.29.42.249
     syscontact: localhost
     syslocation: None
     sysname: None
     version: v2

$ snmp community private
You are about to restart the SNMP service. Do you wish to continue? y
Please update notify to reflect your lastest changes.

$ snmp list
     community: private
     load1: 5
     load15: 3
     load5: 4
     query: snmp://private@172.29.42.249
     syscontact: localhost
     syslocation: None
     sysname: None
     version: v2
```