unix-agent Documentation

Release 1.0.0

Dell Software Group

February 16, 2016

The dcm-agent is an open source python project for use with the Dell Cloud Manager (DCM). When installed inside of a virtual machine that is launched using DCM it gives DCM system level control over the VM instance and thus allows for the automated creation, monitoring, and control of sophisticated cloud applications. Some of the features that it provides are:

- Server health monitoring
- Automated software installation/configuration
- Adding/removing users
- Disk volume management

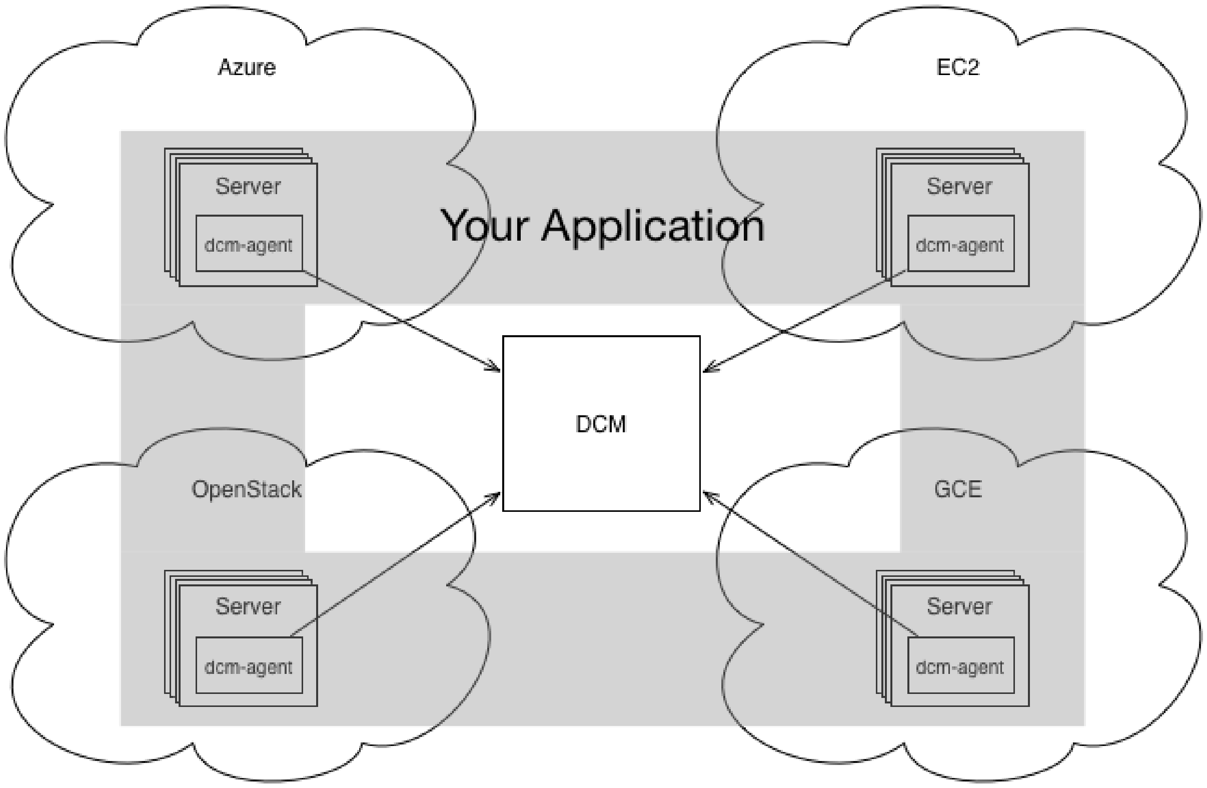
Contents:

CHAPTER 1

Agent Introduction

The dcm-agent is a lightweight service that runs inside of your VM. When the VM is booted and the dcm-agent starts it forms a secure communication channel back to the Dell Cloud Manager. Commands are sent to the dcm-agent along this channel. From simple tasks like adding and removing users on a single VM to the coordination of multiple VMs in a distributed cloud application, the dcm-agent can greatly add to the power of the Dell Cloud Manager.

In this architecture the Dell Cloud Manager becomes the conductor and every VM with a configured dcm-agent provides power for whatever you need orchestrated. With infrastructure cloud computing each VM can be quite powerful and the total number of VMs is pragmatically limitless. The hard part is coordinating all of these pieces to perform in cadence. The Dell Cloud Manager and dcm-agent provide the instrumentation to make this possible.



1.1 Installation

The DCM Agent is distributed in either **rpm** or **deb** formation, however to ease in the installation a script is provided that will determine what package your server needs, download it, install it, and configure the system for use with it.

There are 2 ways to install the DCM Agent using this script:

1. **Interactive mode** - you are prompted for a subset of the available installation options. This is a good option for users who are new to the DCM Agent and want a guided installation process. See the interactive install page for an example session.
2. **Batch mode** - you specify the desired options as command line arguments. This is a good option for those that are automating the installation process. Note that non-interactive mode can be used in combination with “user data” or “startup scripts” on some clouds. See the batch mode install page for an example session.

In addition there are optional environment variables that you can set prior to installing the DCM Agent which affect and direct the installation and allow further custom installs.

Note: Installing the DCM Agent requires **root** authority.

1.1.1 Interactive install syntax

Issue the following command to install the DCM Agent in “interactive mode”.

```
bash <( curl -k -L https://linux-stable-agent.enstratius.com/installer.sh
)
```

You will be prompted for:

1. Cloud provider of the launched server where the DCM Agent will be installed.
2. Web socket URL string for the Dell Cloud Manager provisioning server agentManager service.

   Format: wss://**hostname_or_ipaddress/**agentManager
3. Whether or not to start the DCM Agent at system boot (default is No)
4. Whether or not to install the latest Chef client (default is No)

You can see an example of an interative installation *here.*

1.1.2 Non-interactive install syntax

Issue the following command to install the DCM Agent in “non-interactive mode”.

```
curl -k -L https://linux-stable-agent.enstratius.com/installer.sh
| bash -s - [options]
```

```
Options:

    -c, --cloud           cloud Provider
                          ---------------------------------------------------------------
                          Amazon, Azure, CloudStack, CloudStack3,
                          Eucalyptus, Google, Joyent, OpenStack, Other, UNKNOWN

    -u, --url             Web socket URL of the Dell Cloud Manager provisioning server agent
                          Format: wss://hostname_or_ipaddress/agentManager
                          Default: wss://dcm.enstratius.com/agentManager

    -B, --on-boot         Configure the DCM Agent to start at system boot (default is not to

    -m, --mount-point     Mount point where DCM stores data (default /mnt/dcmdata)

    -t, --tmp-path        Path to the temporary directory  (default /tmp)

    -U, --user            Linux user that the DCM Agent will run as (default dcm)
                          If you specify a different user then that user must already ex-st.

    -o, --chef-client     Install the latest Chef client

    -p, --base-path       Base path where to install the agent (default /dcm)

    -r, --reload-conf     Reload the configuration file (used to populate defaults)

    -L, --loglevel        Log level for logging (ERROR, WARN, INFO, DEBUG)

    -l, --logfile         Name of the DCM Agent logfile (default agent.log)

    -v, --verbose         Increase the amount of output produced by the script
```

You can see an example of an Non-interactive installation *here.*

Note: In most cases it is not necessary to specify the **-c** parameter as the DCM Agent can detect the cloud.

Note: By default the DCM Agent is configured to not be started at system boot. If you wish to have the DCM Agent configured to be started at system boot then specify the **-B** or **–on-boot** option.

Warning: The default value for the web socket URL is **wss://dcm.enstratius.com/agentManager** which is the Dell Cloud Manager SaaS provisioning server. This needs to be changed for On-Premise environments.

1.1.3 Interactive Install Example

Run the following command below with **root** authority to install the DCM Agent in interactive mode.

```
bash <( curl -k -L https://linux-stable-agent.enstratius.com/installer.sh
)
```

```
root@ip-10-29-59-177:~# bash <( curl -k -L https://linux-stable-agent.enstratius.com/installer.s

*** some display output from the install has been omitted in order to reduce clutter ***

Get instance ID called
Instance ID is i-ce752c3f
 0) Amazon
 1) Azure
 2) CloudStack
 3) CloudStack3
 4) DigitalOcean
 5) Eucalyptus
 6) Google
 7) Joyent
 8) Konami
 9) OpenStack
10) Other
11) UNKNOWN
Select your cloud (Amazon): 0
Please enter the contact string of the agent manager (wss://dcm.enstratius.com/agentManager)
wss://66.57.3.53/agentManager
Making the needed directories...
    /dcm
    /dcm/bin
    /dcm/bin
    /dcm/custom
    /dcm/custom/bin
    /dcm/etc
    /dcm/logs
    /dcm/home
    /dcm/cfg
    /tmp
...Done.
Changing ownership to dcm:dcm
Would you like to start the agent on boot? (Y/n)
Y
  Adding system startup for /etc/init.d/dcm-agent ...
    /etc/rc0.d/K20dcm-agent -> ../init.d/dcm-agent
    /etc/rc1.d/K20dcm-agent -> ../init.d/dcm-agent
    /etc/rc6.d/K20dcm-agent -> ../init.d/dcm-agent
    /etc/rc2.d/S20dcm-agent -> ../init.d/dcm-agent
    /etc/rc3.d/S20dcm-agent -> ../init.d/dcm-agent
    /etc/rc4.d/S20dcm-agent -> ../init.d/dcm-agent
    /etc/rc5.d/S20dcm-agent -> ../init.d/dcm-agent
(Optional) Would you like to install chef client? (Y/N) N
To start the agent now please run:
/etc/init.d/dcm-agent start
root@ip-10-29-59-177:~#
```

1.1.4 Non-interactive Install Examples

The following curl command will install the latest DCM Agent and configure it to communicate to this agentManager server: **wss://66.57.3.53/agentManager**. For most DCM Agent installs in On-Premise Dell Cloud Manager server environments that is the only required option. The DCM Agent will also be started after the install finishes.

```
curl -k -L https://linux-stable-agent.enstratius.com/installer.sh |
bash -s - -u wss://66.57.3.53/agentManager && service dcm-agent start
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--     0rhel-6.6-x86_64
100  9933  100  9933    0     0   1939      0  0:00:05  0:00:05 --:--:--  118k
Starting the installation process...
Downloading DCM Agent from https://linux-stable-agent.enstratius.com/dcm-agent-rhel-6.6-x86_64.r
This may take a few minutes.
Downloading https://linux-stable-agent.enstratius.com/dcm-agent-rhel-6.6-x86_64.rpm ...
Installing DCM Agent.

*** The remaining output from the install has been omitted in order to reduce clutter ***
```

```
Click here to view the entire install output
```

The following curl command will install the latest DCM Agent, configure it for the OpenStack cloud, install the latest Chef client, and configure the DCM Agent to communicate to this agentManager server: **wss://66.57.3.53/agentManager**. The DCM Agent will also be started after the install finishes.

```
curl -k -L https://linux-stable-agent.enstratius.com/installer.sh |
bash -s - -c OpenStack --chef-client -u wss://66.57.3.53/agentManager
&& service dcm-agent start
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--     0rhel-6.6-x86_64
100  9933  100  9933    0     0   1952      0  0:00:05  0:00:05 --:--:--  183k
Starting the installation process...
Downloading DCM Agent from https://linux-stable-agent.enstratius.com/dcm-agent-rhel-6.6-x86_64.r
This may take a few minutes.
Downloading https://linux-stable-agent.enstratius.com/dcm-agent-rhel-6.6-x86_64.rpm ...
Installing DCM Agent.

*** The remaining output from the install has been omitted in order to reduce clutter ***
```

```
Click here to view the entire install output
```

1.1.5 Environment Variables

By default the DCM Agent installer detects the Linux distribution of the server where the installer is running on and will download and install the latest appropriate DCM Agent distribution.

These optional environment variables can be set prior to running the DCM Agent installer. They affect some decisions made by the installer and allow for advanced custom installs.

Note: In most cases it is not necessary to use (export) any of these environment variables.

```
AGENT_LOCAL_PACKAGE=<path>         This variable will direct the installer to use the specified file-s
                                   Example: export AGENT_LOCAL_PACKAGE="/tmp/dcm-agent-ubuntu-14.04-an

AGENT_BASE_URL=<url>               This setting will direct the installer to use the specified URL as
                                   By default the agent installer downloads the agent distribution pac

                                   To direct the installer to download the DCM Agent distribution pack
                                   export AGENT_BASE_URL="https://linux-development-agent.enstratius.c

AGENT_UNSTABLE                     If this variable exists then the installer will download and instal
                                   Example: export AGENT_UNSTABLE=YES

AGENT_VERSION=<version>            This setting will direct the installer to download the specified ve
                                   Note: this is the version of the DCM Agent distribution package, no
                                   Example:  export AGENT_VERSION="9.12"

DCM_AGENT_FORCE_DISTRO_VERSION=    This setting will direct the installer to use the specified DCM Age
                                   Example:  export DCM_AGENT_FORCE_DISTRO_VERSION="rhel-6.6-x86_64"
```

Note: The DCM Agent distribution packages have this naming convention: dcm-agent-<distribution>-<version>-<architecture>-<agent-version>.<pkg type>

```
Where:
        <distribution>   is the linux distribution name (e.g. centos, debian, rhel, or ubuntu)
        <version>        is the linux distribution version (e.g. 12.04)
        <architecture>   is the linux distribution architecture (e.g. i386, x86_64 or amd64)
        <version>        is the DCM Agent version (e.g. latest, 0.9.11, 0.9.12, etc.)
        <pkg type>       is the linux distribution package type (e.g. deb or rpm)
```

1.1.6 Manual Installation

Although it is not recommended or officially supported, the dcm-agent can be installed manually into a python 3.4 virtual environment by running the following commands in the src directory:

```
$ pip install .
```

One installed the program *Configuration* must be run in order to properly setup the system for use with the agent.

Configuration

The installer configures the agent initially, but the configuration values can be changed without reinstalling the agent.

One installed the program *dcm-agent-configure* can be run in order to (re)set the values in the configuration file.

You can do this interactively or by explicitly passing arguments. To run the configuration tool interactively pass it the -i option. This will prompt you with questions to answer.

```
$ dcm-agent-configure -i
```

For those looking to automate an installation needed options can be set from the command line instead. The needed options are:

–cloud {Amazon, etc...}, -c {Amazon, etc...} The cloud where this virtual machine will be run. Options: Amazon, Atmos, ATT, Azure, Bluelock, CloudCentral, CloudSigma, CloudStack, CloudStack3, Eucalyptus, GoGrid, Google, IBM, Joyent, OpenStack, Rackspace, ServerExpress, Terremark, VMware, Other

—url URL, -u URL The location of the DCM web socket listener

–base-path BASE_PATH, -p BASE_PATH The directory to which the agent should be installed.

This program will create a directory structure under <BASE_PATH>. In it you will find the file <BASE_PATH>/etc/agent.conf. This file contains information about your agent installation.

Re-configure

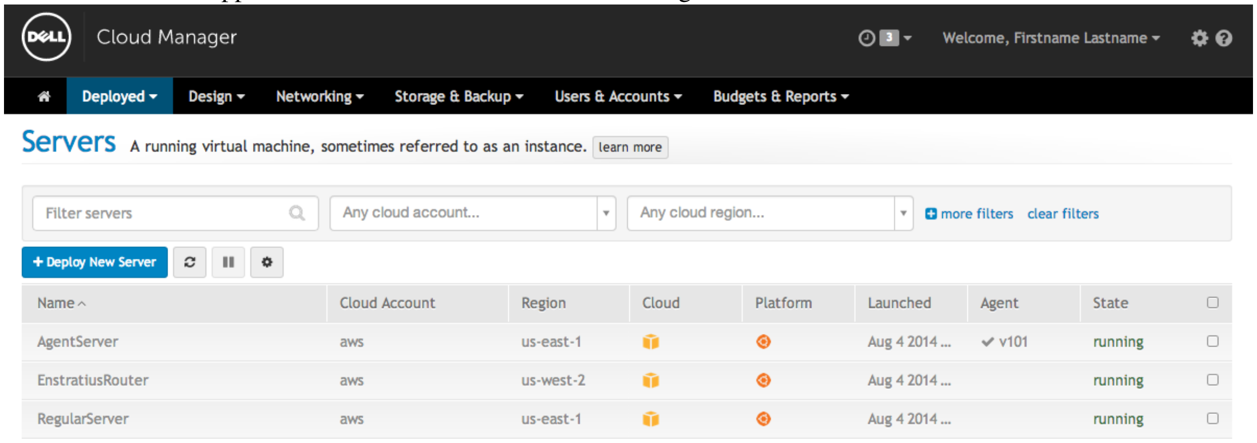
The dcm-agent-configure program can also be used to update an installed agent's configuration. To reconfigure run

```
$ dcm-agent-configure -r <BASE_PATH>/etc/agent.conf
```

This will read in the current configuration and use all of its values as defaults. Any other passed in command line options or answers to interactive questions will override these defaults.

1.2 Console Interactions

With the Dell Cloud Manager you can launch servers on any configured cloud via the console. How to do this is described <a href="/docs/compute">here</a>. Once the agent is installed and started in a server that the console recognizes that agent will perform a handshake with the Dell Cloud Manager. Servers that have successfully completed the handshake will appear in the list of servers with their dcm-agent version number as shown below.



Name	Cloud Account	Region	Cloud	Platform	Launched	Agent	State
AgentServer	aws	us-east-1			Aug 4 2014 ...	✓ v101	running
EnstratiusRouter	aws	us-west-2			Aug 4 2014 ...		running
RegularServer	aws	us-east-1			Aug 4 2014 ...		running

Note the difference between *AgentServer*, a server with a connected dcm-agent, and *RegularServer* a server with no dcm-agent. The first has the *v101* in the Agent column. This agent version will vary with the latest DCM agent.

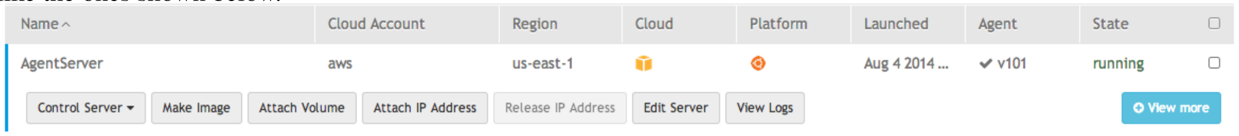
Next Topic Add Users

1.3 Add a UNIX user

When a dcm-agent is connected to the Dell Cloud Manager it provides many additional features. One feature is the ability to add system users to the running server. Once added new users can ssh into the running virtual machine.

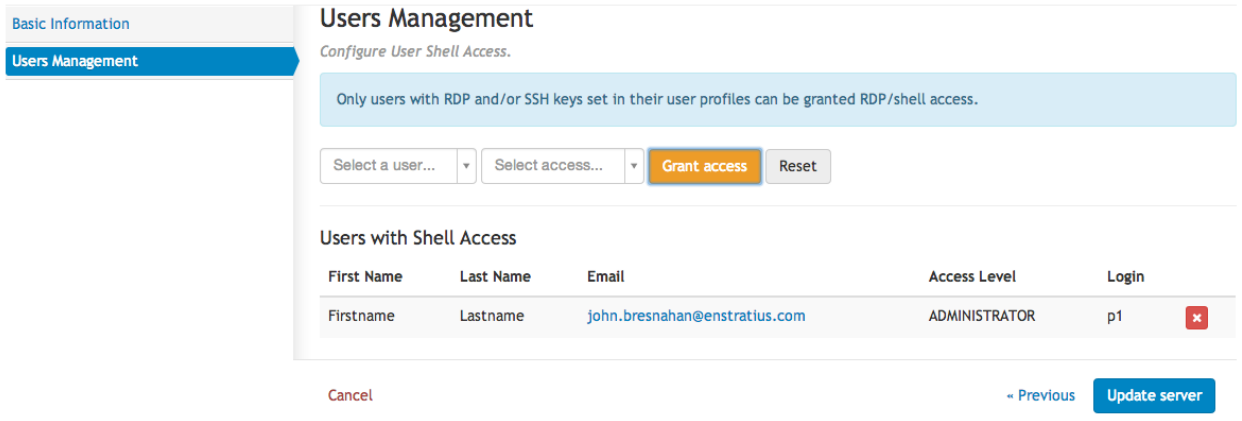
1.3.1 To add a new user

Find your server in the list of deployed servers and click on its row. This will cause a new set of buttons to be displayed like the ones shown below.



Name	Cloud Account	Region	Cloud	Platform	Launched	Agent	State	☐
AgentServer	aws	us-east-1			Aug 4 2014 ...	✓ v101	running	☐

Click on *Edit Server*. This will bring you to a screen similar to the one below:



Users Management

Configure User Shell Access.

Only users with RDP and/or SSH keys set in their user profiles can be granted RDP/shell access.

Users with Shell Access

First Name	Last Name	Email	Access Level	Login
Firstname	Lastname	john.bresnahan@enstratius.com	ADMINISTRATOR	p1

Select the user name that you want to add and the level of access and click *Grant Access*. This will add the user name to the list below. Take note of the login name. In the above example the login name is *p1*. Now click on *Update server*. In a short period of time you will be able to ssh into that server under the user name of *p001*.

1.4 Agent Troubleshooting

At times users may experience problems with the agent. Here we will explain some troubleshooting steps that may help diagnose and fix common problems.

- Gather agent information. In order to receive help from the DCM support team it is helpful to provide information about your installed agent. The DCM Agent is bundled with a tool that will gather the needed information put it into a tarball. To create this tarball run the following:

```
% sudo /opt/dcm-agent/embedded/agentve/bin/dcm-agent --report
***********************************************************************
To get all log and configuration files copy /tmp/agent_info.tar.gz to
your local machine
***********************************************************************
```

  When reporting a problem please include this tarball.

- Inspect the logs. Useful information about the dcm-agent's execution is written to the file /dcm/log/agent.log. This file can be inspected for error messages and other useful information.

- Increase the log level. By default the dcm-agent logs at the "INFO" level. At times it may be useful to increase the level to DEBUG. This will result in very verbose logging and may provide the information needed. To increase the log level do the following:
  - Edit /dcm/etc/logging.yaml
  - Change occurrences of INFO to DEBUG
  - restart the agent:

```
% sudo /etc/init.d/dcm-agent restart
```

  - Verify the agent manager URL. During normal operation the dcm-agent forms a TCP connection with DCM. If this connection cannot be made the dcm-agent will not work. The contact point is set in the file */dcm/etc/agent.conf* under the setting *agentmanager_url=<URL>*. Verify that the value of URL is correct. In most cases it should start with *wss://* (note the difference between that and *ws://*). Even in cases when the URL is correct there may be a firewall blocking the connection from happening. To verify that a connection can be made from your VM to DCM run the following:

```
% telnet <URL host> <URL port>
Trying 1.2.3.4...
Connected to example.com.
Escape character is '^]'.
^]quit
```

1.5 Agent Architecture

While the full internal workings of the dcm-agent and its interactions with Dell Cloud Manager is outside of the scope of this document, some details are provided here to assist the user in understanding how to run it.

The dcm-agent is a multi-threaded python program that runs inside of a VM which was launched by DCM. A connection is formed from the dcm agent to DCM over which instructions from DCM flow. The instructions cause the agent to run system changing commands which allow DCM to coordinate and control cloud applications.

1.5.1 Web Sockets

The agent connects to DCM using web sockets. The connection direction is always out, so no in-bound ports need to be opened up on your server and it is NAT-friendly. When the agent starts it creates a thread of execution which periodically attempts to form a connection with DCM.

Once the connection is formed, commands can be received from DCM and executed by the agent. The agent can also respond to completed commands and can also log information back to DCM.

If a web socket connection is lost, the agent will periodically attempt to re-connect. This is a normal part of the agent's execution. Web socket connections will time out from being idle, or DCM may close them, or a network partition may happen. The agent is tolerant of all such situations.

CHAPTER 2

Supported Linux Distributions

The following Linux distributions are currently supported by the Linux DCM Agent:

- Ubuntu 14.04 (amd64, i386)
- Ubuntu 12.04 (amd64, i386)
- Ubuntu 10.04 (amd64, i386)
- Debian 7.8 (amd64, i386)
- Debian 6.0 (amd64, i386)
- CentOS 7.0 (x86_64)
- CentOS 6.5 (x86_64, i386)
- CentOS 6.4 (i386)
- RHEL 7.1 (x86_64)
- RHEL 7.0 (x86_64)
- RHEL 6.5 (x86_64)
- RHEL 6.4 (x86_64)

For the above distributions omnibus packages targeted at the specific distribution are built and published. The omnibus packages contain the full stack of dependencies including python 3.

Linux Agent Upgrade Tool

The purpose of this tool is to facilitate the upgrade of the agent so as to require as little as possible from the user running the upgrade.

The tool itself preserves the agent configuration values the user had previously defined in order to use them with the newer, upgraded agent.

The tool is located within the agent repo itself as of version 0.11.3. Go to `dcm/agent/tests/autoupgrade` to find the tools. The tools will also be available in the online repos where the agent packages themselves are found.

There you will find the actual tool called upgrade.py and a small shell wrapper for your convenience called upgrade.sh.

The wrapper script can accept a small number of args, only one of which is required. Here is a description of the arguments you can pass:

```
Usage: ./upgrade.sh version [ base_dir [ package_url [ allow_unkown_certs ]]]

        --version [VERSION]        Required and sets AGENT_VERSION env var.
        --base_dir [DIR]           Optional and sets AGENT_BASE_DIR, default is /dcm.
        --package_url [URL]        Optional and sets AGENT_BASE_URL, default is http://linux-stable-a
        --allow_unknown_certs/-Z   Optional flag to disable cert validation
```

`--version` is the only required parameter and is set to the version you wish to upgrade to.

`--base_dir` tells the upgrade tool where your dcm installation is located. The default is /dcm.

`--package_url` tells the upgrade tool where to download the agent installer and package from. The default is http://linux-stable-agent.enstratius.com.

`--allow_unknown_certs/-Z` tells the upgrade tool to specify to the agent to ignore cert checking. This is the one option which will not be automatically picked up by the tool. Starting in version 0.9.19 this was available as an option. If you are upgrading from an agent older than that, you may need to specify this flag if you have an on-premise install of DCM and/or if your DCM is operating without a valid certificate.

The agent report option is invoked when the tool is started to aggregate logs and meta info about the install into a gzip and again after the upgrade. In addition the upgrade tool logs to a single file while it is running. These items will be copied to a temp directory for your consumption if the need arises. You will see a message at the finish of the upgrade telling you the location.

CHAPTER 4

Plugin Development

The DCM Agent provides a well defined abstraction to plugins that can be authored by third parties. These plugins allow for the set of agent commands to be extended. For more information see the links below.

4.1 DCM Agent Commands

The dcm-agent's main purpose is to provide a remote set of commands to DCM. The dcm-agent forms a websocket connection with DCM and advertises the set of commands which DCM can invoke. There is a set of built in commands that can be found in the module dcm.agent.plugins.builtin.

4.1.1 Extension Plugins

Via a well defined interface to the *command plugins* this set can be expanded to support user defined functions. When implementing a plugin a set of utilities are defined for use by the author in the dcm.agent.plugin.api module.

Extension plugins are made by creating a python package that has one or more modules in it which are properly constructed plugins. The module is then installed into the same python environment as the agent. At that point the tool *dcm-agent-add-plugins* can be used to locate all of the new plugins in the newly installed package and add them to the existing DCM Agent installation.

```
$ dcm-agent-add-plugins <package name>
```

This program will search every submodule of the given package name for dcm-agent plugins. It will then modify the file /dcm/etc/plugins.conf to make these plugins available to the agent.

4.1.2 Plugin Modules

Each plugin must be in its own python module that is conventionally named after the command_name. For example, the add_user command is implemented in file called add_user.py.

4.1.3 DCM Agent Plugin API

4.1.4 Plugin Base Class

4.1.5 Plugin Reply Class

4.1.6 Plugin Exceptions

4.1.7 Plugin Utility Functions

A set of utility functions are provided to plugin authors which allow for interaction with the dcm-agent.

Parameter Validators

When defining a plugin's accepted parameters the author must define a type function. Parameters come over the wire from DCM as strings. They are converted to specific types by calling these conversion functions. If the string is invalid these functions can throw a TypeError. Functions like str, int, bool, and float which are built into python can be used for primitive types. Plugin authors can write there own as well. Below are a set provided to authors.

Logging

Plugin authors can make use of the standard python logging module in the following way.

```
import logging

_g_logger = logging.getLogger(__name__)
_g_logger.debug("A log message")
```

Additionally plugins are allowed to log messages back to DCM. Care should be taken when choosing what to log back to DCM because this results in network traffic and additional load on both the dcm-agent and DCM. To log a message to the DCM console the following function is used

Utilities

The following functions are available to plugin authors. Plugin authors should not make any other calls into dcm.agent modules or class methods.

An example dcm-agent plugin can be found here.

4.2 Plugin Development Modules

4.2.1 dcm.agent.plugins.api

4.2.2 dcm.agent.plugins.api.base

4.2.3 dcm.agent.plugins.api.utils

4.2.4 dcm.agent.plugins.api.exceptions

CHAPTER 5

Authenticating and Authorizing Connections

The agent is a process that runs with root access inside of a server. It connects to DCM and takes instructions from DCM that run system changing commands. These commands do things like add additional users that also have root level access. Because of this it is crucial that this connection is safely formed and that both sides of the connection are assured they are in communication with the trusted and expected part.

When the agent connects to DCM it need to be assured that there is no bad actor spoofing the expected DCM server and that there is no man in the middle. How this is achieved is discussed here agent_trust.

DCM must also decide if the agent that connected is safe in the following ways:

- It is running on the server it claims to be. This is discussed here: handshake.
- It is talking to a trusted user on that server. This is discussed here token.

CHAPTER 6

Agent Image Scrubber

DCM allows customers to create images from their current running image. This is a very helpful features that lets a customer configure a virtual machine with the software and services that they need and then take a snapshot of the instance for future use. This powerful feature can also be dangerous. Any private keys or other files which include secrets that were on the system at the time the image was made will be on the child image. Anyone who then boots the child image will have full root level access to all of the data on it and thus access to any secrets left on that image. Thus any such files need to be removed before the image is created. Further there can be data caches that can confuse the image on reboot (dhcp leases and cloud-init startup scripts for example). These files should be removed as well.

The agent distribution comes with the program dcm-agent-scrubber. This program helps clean virtual machines in preparation for making an image from them. For a detailed list of options run:

```
dcm-agent-scrubber --help
```

It is important to note that this program will delete files from the system on which it is run. Some of these files are needed for the system to function properly (eg: /dcm/secure/token). Because of this the scrubber includes a way to safely create a recovery file.

The recovery file is a tarball that contains every file that was removed from the system by the scrubber. The system can be restored with this file by untarring it in the root directory.

Of course if this recovery file is left on the created image the same problem remains, a bad actor could read secret information from it. To solve this problem the scrubber will encrypt the recovery file with the instance owners RSA public key making it so only the original owner of the instance can access the data in the recovery file.

note: It is still recommended that the recovery file be removed from the system before making an image from it.

6.1 Scrubbing The Server

Before making an image from a server that is already running the dcm agent we recommend running the scrubber in the following way:

```
dcm-agent-scrubber -X -k -A -H -t -r /tmp/dcm_scrubber_backup.tar.gz -e <path to your ssh public
```

Then copy the file */tmp/dcm_scrubber_backup.tar.gz.<ssh key name>* off of the server and remove it from the server.

6.2 Recovery

The recovery file is a gziped tarball with the following format:

- recovery.sh A script which aids in the recovery process

- data.enc The data removed from the system by the scrubber in a tarball. If encryption was used to create this it will be an encrypted file.
- public_key The public key used to encrypt the symmetric key used to encrypt data.enc. If encryption was not used this file will not exist.
- key The encrypted symmetric key used to encrypt data.enc. This file must be decrypted by the private key which matches the public key used to create it. The decrypted result can then be used to decrypt the data.enc and thereby recover the data. If encryption was not used this file will not exist.

To recover the scrubbed data untar the recovery file and run the recovery.sh file. This will do all of the decryption work needed and will result in an unencrypted tarball. That tarball can then be copied to the server and untarred in the root directory.

To recover all the deleted files run the following command from the system that contains your private key:

CHAPTER 7

Testing DCM Agent For Linux

This document covers how to run tests locally for development purposes.

Local development tests are supported by Vagrant. You will find a multi machine Vagrantfile located in `src/dcm/agent/tests`. Running the test suite on a particular distribution is as simple as going to that directory and running `vagrant up <machine name>` . You can get a list of available machine names by running `vagrant status`.

Vagrant up provisions the image with an agent and runs the tests suite with coverage. You will see the output in your terminal.

The following environment variables can be set to customize the test environment.

```
TEST_AGENT_STORAGE_CREDS
TEST_AGENT_VERSION
TEST_AGENT_BASE_URL
TEST_AGENT_LOCAL
TEST_AGENT_DOCKER_INSTALL
```

1. **`TEST_AGENT_STORAGE_CREDS` is optional and should point to a local file with entries listed like so**
   and enables a subset of tests dealing with mount/unmount commands to run.:

   1 <aws_access_key> <aws_secret_key> us_west_oregon

2. `TEST_AGENT_VERSION` is optional and lets you explicitly set the version to test.
3. `TEST_AGENT_BASE_URL` is optional and lets you explicitly set where to get the agent package to test.
4. `TESTS_AGENT_LOCAL` is optional and runs the test suite against the local source code.
5. `TEST_AGENT_DOCKER_INSTALL` is optional and provisions the VM with the *agent_docker* script.

CHAPTER 8

Running Tests Manually

You can alternatively run the tests manually in the VM after the VM is in a running state.

Simply `vagrant ssh <machinename>` and do

```
$ sudo -i
$ source /opt/dcm-agent/embedded/agentve/bin/activate
$ nose2 -v dcm.agent.tests
```

CHAPTER 9

Using a Remote Debugger

Coming Soon!

CHAPTER 10

Indices And Tables

- genindex
- modindex
- search