Dell Cloud Solution for OpenStack™ Solutions

Crowbar Users Guide

Notes, Cautions, and Warnings

NOTE: A NOTE indicates important information that helps you make better use of your system.

CAUTION: A CAUTION indicates potential damage to hardware or loss of data if instructions are not followed.

WARNING: A WARNING indicates a potential for property damage, personal injury, or death.

Contents

1 Introduction 5
1.1 Concepts 5
1.2 OpenStack 5
1.3 Opscode Chef Server 5
1.4 Dell Specific Options 5
2 The Crowbar Solution 6
2.1 Architecture 6
2.2 System end state 7
• Node provisioning 7
• Networking administration 7
• NTP 8
• DNS 8
• Nagios 8
• Ganglia 8
• Logging 8
2.3 Network setup 8
3 User Interface 9
3.1 Using Crowbar 9
3.2 Nodes 10
3.3 Barclamps 10
3.4 General Layout 10
3.5 Dashboard 10
3.6 Node Switch Groups 10
3.7 Node Details 11
4 Overview 12
5 Deployment Functions 13
5.1 Life Cycle 13
5.2 PXE State Machine 14
5.3 Barclamps 16
5.4 Crowbar Barclamp 17

5.5 Deployer Barclamp 18
5.6 Provisioner Barclamp 19
5.7 Network Barclamp 20
5.8 RAID Barclamp 21
5.9 BIOS Barclamp 22
5.10 IPMI Barclamp 22
5.11 NTP Barclamp 23
5.12 Logging Barclamp 23
5.13 Nova Barclamp 24
5.14 Swift Barclamp 27
5.15 Glance Barclamp 28
5.16 Nagios Barclamp 29
5.17 Ganglia Barclamp 29
5.18 Test Barclamp 30
5.19 Proposals 30
5.20 Active Roles 31
6 Supplemental Material 32
6.1 System Verification 32
6.2 Deactivating Barclamps 32
6.3 Creating Barclamps 32
6.4 Gathering Log Information 32

1 Introduction

This document provides instructions for users of **Crowbar** to assist in deploying **OpenStack** components including **Nova, Glance** and **Swift**.

Other suggested materials:

- OpenStack Reference Architecture (RA, July 2011)
- Crowbar Getting Started Guide (July 2011)
- Bootstrapping OpenStack Clouds (Dell Tech White Paper , Feb 2011)

1.1 Concepts

The purpose of this guide is to explain the user interface of Crowbar. Please consult the getting started guide for assistance with installing Crowbar and configuring the target system.

Concepts beyond the scope of this guide will be introduced as needed in notes and references to other documentation..

1.2 OpenStack

The focus of this guide is the use of Crowbar, *not* OpenStack. While Crowbar includes substantial components to assist in the deployment of OpenStack, its operational aspects are completely independent of OpenStack.

For detailed operational support for OpenStack, we suggest visiting the OpenStack documentation web site at http://docs.openstack.org/.

This guide will provide this additional information about OpenStack as notes flagged with the OpenStack logo.

1.3 Opscode Chef Server

Crowbar makes extensive use of Opscode Chef Server**,** http://opscode.com. To explain Crowbar actions, it is helpful (but not required) to understand the underlying Chef implementation.

To use Crowbar, it is **not** necessary to log into the Chef Server; consequently, use of the Chef UI is not covered in this guide. Supplemental information about Chef will be including using this formation.

This guide will provide this additional Chef information as notes flagged with the Opscode logo.

1.4 Dell Specific Options

The Dell EULA version of Crowbar provides additional functionality and color pallets than the open source version. When divergences are relevant, they will be identified.

To perform some configuration options and provide some integrations, we use libraries that cannot be distributed via open source.

Crowbar is **not** limited to managing Dell servers and components. Due to driver requirements, some barclamps, e.g. BIOS & RAID, must be targeted to specific hardware; however, those barclamps are not required for system configuration.

2 The Crowbar Solution

2.1 Architecture

Crowbar provides a modular platform containing the building blocks to provision, monitor and operate a large scale cloud deployment. Starting with bare metal installation, Crowbar automates all the required installation and configuration tasks. The core capabilities provided are:

- Hardware configuration – updating and configuring BIOS and BMC boards
- Deployment of base OS
- Deployment of cloud components
- Providing core network infrastructure services (NTP, DNS, DHCP)
- Monitoring availability and performance of all deployed components

To allow easy integration into customers' existing environments, Crowbar allows customization of its components. Customers can disable the default monitoring tools (Nagios and Ganglia) if they prefer to use their own existing monitoring tools, and internal cloud services can be connected to extant services; for example, the cloud's NTP service can be configured to synchronize with existing servers.

Each function in the cloud is controlled by a Crowbar component called a Barclamp. There are Barclamps for Nagios, Ganglia, NTP, and a variety of other basic services. Each Barclamp is responsible for all the aspects of the underlying technology required to make it usable. To control the operation of a Barclamp, a user creates a proposal for the Barclamp (or may edit one already in place). A proposal comprises several parts:

- Parameters to customize the operation of the function; e.g. upstream DNS resolvers to use
- List of machines in the deployment that fulfill the different roles in the function; e.g. for Swift, which machine will play the proxy-server role and which ones will be storage nodes
- Internal system information

When provisioning a function, a user will generally start with a proposal generated by Crowbar. Each core service running on the admin server has a default proposal included as part of the Crowbar installation. A user may edit these proposals before installing these services on the admin node.

For OpenStack (i.e. Swift, Nova and Glance) components, a user can employ Crowbar tools to construct a starting proposal and then edit it to fit the specific requirements for their environment. Once the proposal is ready, the user commits the proposal.

When a proposal is committed, Crowbar configures the Chef server and other components in the systems (TFTP, DHCP, etc.) to build the setup described in the proposal. Machines in the deployment affected by the proposals have their configuration updated using Chef client-side components. At the end of the process, the function described by the proposal is ready for use.

Finally, a cloud deployment is dynamic. Machines come and go, break down, or get repurposed. Crowbar's operational model ensures that machines are "hooked" into the key infrastructure services. Critical services (e.g. Nagios monitoring) are installed automatically on newly provisioned machines by default, and those machines may be easily allocated for use with any additional Crowbar services desired.

2.2 System end state

The sections that follow describe the services and capabilities available assuming the system is installed with defaults. As mentioned above, the Crowbar framework allows for many customizations. This section focuses on the primary use cases for crowbar, namely integrating all the functions into an existing network environment. Later sections will describe more advanced customization options.

• Node provisioning

When a new node is added to the system, it must be set up to allow PXE booting. Once a machine is powered on, Crowbar utilizes the PXE boot protocol to manage its provisioning process.

After a system is fully installed by Crowbar, it has the following characteristics:

- BIOS is updated and configured based on the system's usage
- BMC (Board Management Controller) is configured to allow management and IPMI support
- Administration access to the OS is configured – IP addressing and SSH keys are installed
- Nagios and Ganglia monitoring scripts are installed for the functions deployed on the system
- Chef-client daemon is configured to maintain the system's state in sync
- NTP sync client is configured

Additionally, the system is configured to fulfill the functions that are deployed on it and is added to the appropriate cluster (e.g. a Swift or Nova cluster).

• Networking administration

Crowbar's network Barclamp carries on responsibilities related to L2/L3 management, namely:

- Physical NIC configuration – BMC port allocation (teamed or not)
- VLAN configuration on nodes
- IP address location service, utilized by the rest of crowbar. Addresses can be allocated from different pools, meant for different usages; e.g. Admin network, BMC, Storage and Public

The above functions involve managing information on the server and executing operations on nodes as they are provisioned. On a node, NICs are defined to match VLANs and appropriate addressing information is configured.

- **NTP**

The admin node runs an NTP server to synchronize time on all the machines in the cluster. Optionally, the NTP server can synchronize with upstream servers, in which case nodes are configured to sync their local time to those servers instead.

- **DNS**

The admin node runs a DNS server to allow resolution of internal and (optionally) external names. The DNS server can be configured with the following:

- A list of upstream DNS servers to contact
- A set of static mappings
- The default domain name suffix

Crowbar ensures that when a new machine is added to the deployment, it has a default entry added to the DNS zone. The default host name will be the machine's MAC address prefixed by the letter 'd' (e.g. d00-a4-b3-c2-d1-e0.yourdomain.com).

- **Nagios**

A Nagios server is installed on the admin node and is configured to monitor all the nodes in the system. As new nodes are brought online, Crowbar will dynamically update Nagios to include them.

Nagios monitors provisioned services (Swift, Nova and the networking infrastructure) for availability. Each cluster instance is represented as a "host group", allowing users to quickly identify the health of a given instance.

- **Ganglia**

Ganglia monitors the installed cluster for capacity and performance information, allowing the user to easily gauge the cluster's capacity and check recent activity.

- **Logging**

The admin node serves as a central log repository. Its syslog daemon is configured to accept remote messages and each node is configured to forward all messages there.

2.3 Network setup

The network configuration assumes a flat L2 wiring – all network connections should be accessible at that layer. Where isolation between different logical networks is required, VLANs are used.

The default networks are presented in the following table.

Usage	Description	Default reserved vLAN tag	Tagged
Admin/Internal vLAN	Used for administrative functions such as Crowbar node installation, TFTP booting, DHCP assignments, KVM, system logs, backups, and other monitoring. There is only one vLAN set up for this function and it is spanned across	100	Not tagged

	the entire network.		
BMC vLAN	Used for connecting to the BMC of each node.	100	Not tagged
Storage vLAN	Used by the Swift storage system for replication of data between machines, monitoring of data integrity, and other storage specific functions. (802.1q Tagged)	200	Tagged
External vLANs	Used for connections to devices external to the OpenStack Cloud infrastructure; these include externally visible services such as load balancers and web servers. Use one or many of these networks, dependent on the need to segregate traffic among groups of servers. (802.1q Tagged)	300	Tagged
Nova Floating	Assigned to Nova VMs by the Nova manager.	400	Tagged
Nova Fixed	Assigned to Nova VMs by the Nova manager.	500	Tagged

Note: The admin and BMC networks are expected to be in the same L2 network.

3 User Interface

The Crowbar interface has two primary concepts: nodes and barclamps. All actions are focused on management of these two elements.

3.1 Using Crowbar

Crowbar is delivered as a web application available on the "admin" node via HTTP on port 3000. By default, you can access it using http://192.168.124.10:3000 (see table below). Additionally the default installation contains an implementation of Nagios and Ganglia for status and performance monitoring of the installation.

Nagios, Ganglia and Chef can be accessed directly from a web browser or via selecting one of the links on the Dashboard in the crowbar UI.

Service	URL	Credentials
SSH	openstack@192.168.124.10	openstack
Crowbar UI	http://192.168.124.10:3000/	crowbar / crowbar
Nagios UI	http://192.168.124.10/nagios3	nagiosadmin / password
Ganglia UI	http://192.168.124.10/ganglia	nagiosadmin / password
Chef UI	http://192.168.124.10:4040/	admin / password

Crowbar has been tested on the following browsers: FireFox 3.5+, FireFox 4.0, Internet Explorer 7, and Safari 5. A minimum screen resolution of 1024x768 is recommended.
The IP address (192.168.124.10) is the default address. Replace it with the address assigned to the Admin node.

3.2 Nodes

Nodes represent distinct servers in your environment. A server is a single operating system that can be physical or virtual with multiple NICs and HDDs. Each server is identified uniquely by the MAC address of the NIC on the administrative network.

Crowbar nodes map directly to Chef nodes. In fact, all data used in Crowbar is stored entirely in Chef. Chef is the database for Crowbar. Changing a node's data in Chef changes it in Crowbar.

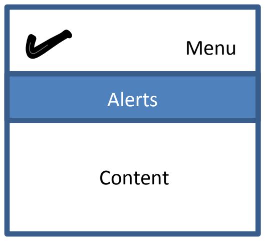
3.3 Barclamps

Barclamps represent modular capabilities that you can install onto none, some, or all of the nodes in your environment. Barclamps are activated by generating a 'Proposal' for that Barclamp. It is possible to generate multiple proposals for a barclamp. Once a proposal has been reviewed, you must activate it before it becomes an 'Active Role' in the system.

In addition to logic for Crowbar, barclamps are decomposed in Chef as multiple components: crowbar data bag entries, cookbooks, recipes, and roles. Our objective is to allow the Chef components used by Barclamps to operate in Chef even without Crowbar.

Barclamps have a specific life cycle that will be discussed in more detail as we explore the user interface. Information about using, creating, and extending barclamps is included in the Supplemental Material section.

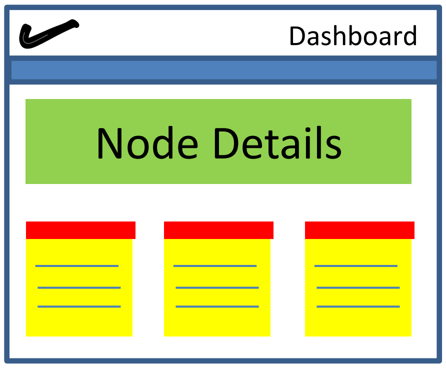
3.4 General Layout



The menu for Crowbar is displayed on the upper right side of the page. Clicking on one of the menu sections will cause related content to be displayed on the lower section of the screen.

Alerts or confirmation messages may be displayed between the menu and the page content. Most Crowbar screens automatically update state information so users should not have to refresh the page to get the most current information.

3.5 Dashboard



The Dashboard shows all the nodes in the system and allows users to manipulate their power and configuration states.

3.6 Node Switch Groups

Nodes are shown organized by the physical switching infrastructure which is discovered automatically during the configuration process. The switch and port of each node's administrative network interface is used to determine groups and order within groups.

If you use a consistent pattern for connecting nodes to switches then the Crowbar display will match your nodes' physical location.

The top of the group box (red in illustration) shows the Switch MAC address and a pie chart of the nodes' status within each group. Nodes (yellow in illustration) connected to the switch display in port order (lowest on top) with their current deployment state shown by the status light.

The possible states are:

Status	Icon	Comment	User Action
Ready	🟢	Requested services provisioned.	
Pending	🟡	Holding state after discovery	Node waiting to be activated
Not Ready	🔴 blinking	Crowbar & Chef actively deploying	
Unknown	●	In between states or not reporting for 20 minutes (powered off).	Restart server if desired

The Admin node is the node that runs the Crowbar, Chef and other core services. It exists in the system before the user interface is available.

3.7 Node Details

Clicking on a node's name will display details about the selected node in the details panel (green in illustration). The detail panel displays important information about the node including its FQDN, uptime, switch connectivity hardware profile, and a detailed list of all active network connections.

Node detail only shows a tiny fraction of the total details that Chef tracks for each node. To see the complete list, examine the Run List and Attributes for each node in Chef.

The Links list is barclamp specific and will expand depending on which barclamps are using the selected node. Links open a new window to view additional information about the node.

The Barclamps and Roles lists indicate what capabilities have been assigned to the node.

The Deployment Functions section has a much more detailed discussion of Barclamps and Roles.

When a role or barclamp is selected in the details panel, the nodes that share the same barclamp or role are highlighted in the group panel. This helps quickly identify groups of nodes that are similarly configured.

The buttons on the top of the details panel (Identify, Power On, Shutdown and Reboot) use the node's IPMI interface to change the node's physical state. These states will cause the node status to be "unknown." These buttons will only be available if the system was able to successfully configure the BMC on the target system.

Button	Action	Useful When
Identify	Will cause the identify light to blink for 15 sec	Trying to identify a node within a rack
Power On	Send a power on signal to the BMC of the selected system	Remotely powering on a system

Shutdown	Send a power off signal to the BMC of the selected system	Remotely powering on a system
Reboot	Send a power cycle signal to the BMC of the selected system	Remotely power cycling a system which has stopped responding

The buttons on the bottom of the details panel (Delete, Reset, Reinstall and Hardware Update) will reset the node's deployment state. These functions are very useful during lab configurations when the system is being continuously reconfigured. The buttons take the following actions:

Button	Action	Config Lost?	Reboot?	Useful When
Delete	Completely remove all records of the node from the Crowbar/Chef database. If a node is deleted, it will be rediscovered if it reboots.	Yes	No	Removing nodes
Reset	Remove all the roles assigned to the node and reimage it back to an unassigned node.	Yes	Yes	Reallocate the node for a new purpose
Reinstall	Reimage the node and then reapply the current deployment profile to the node. This effectively rebuilds the server back to a pristine state.	No	Yes	Tuning the Chef recipes or configuration details
Hardware Update	Keeps the current configuration, but forces the node to reboot.	Yes	Yes	To apply BIOS or RAID updates.

Using the Edit link (after the node name in the top left) allows you to make per node decisions about how the node will be deployed.

You **must** "Allocate" the node to allow Crowbar to complete the deployment group panel. The allocate step acts as a pause state for deployment so that you have time choose a node's role in the system before Crowbar provisions it. It can also be used to simulate white listing.
To "Allocate" a node, you may manually allocate the node from the edit page or you may include that node in an applied barclamp proposal.

4 Overview

This user screen is available in the Dell version.

The Overview page shows a reference taxonomy for a Cactus version OpenStack deployment. Crowbar highlights the sections of the taxonomy that have been enabled in the system. Like most Crowbar pages, this page updates automatically so changes in the system status are automated reflected.

There are no user actions for the Overview page. It is a view only page.

For the API components of the taxonomy (Nova, Glance and Swift), Crowbar will provide the IP address on which these components may be accessed. This is the fastest way to determine which identify which nodes are hosting API services.

5 Deployment Functions

To allow control of the process, Crowbar breaks deployment into phases. You must activate and allocate nodes before Crowbar will implement optional services.

5.1 Life Cycle

Understanding the Barclamp life cycle is essential to understanding Crowbar interface layout.

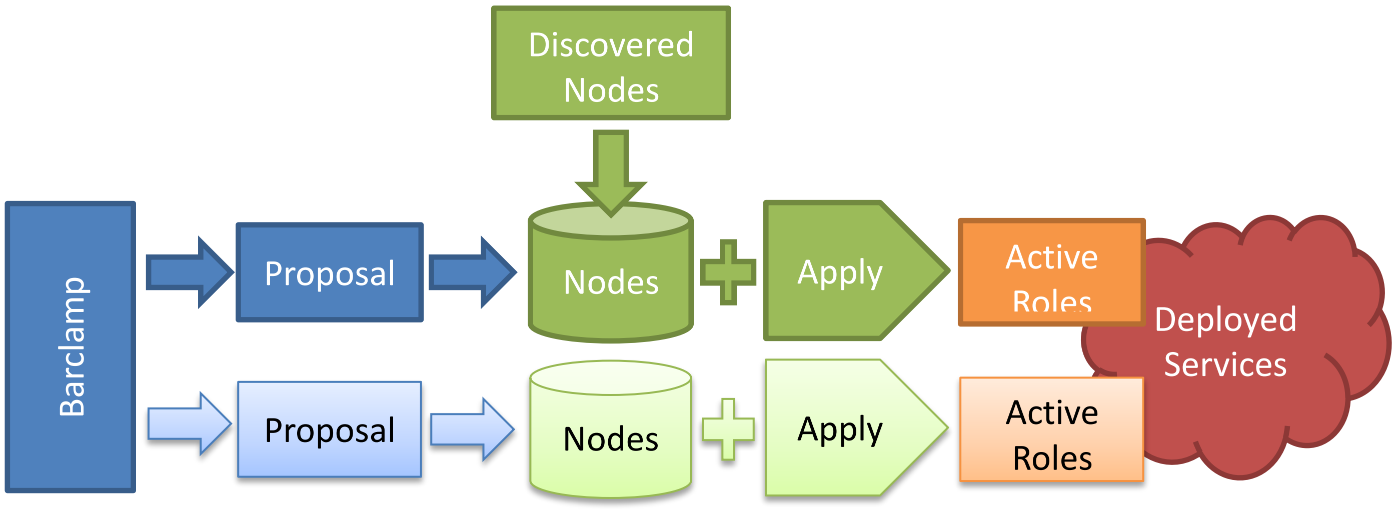


Figure 1 shows the life cycle of a barclamp from concept to proposal and deployment.

Figure 1 shows the entirety of a barclamp within the Crowbar user interface. A Barclamp defines the capability for a service but cannot be deployed. To deploy a barclamp, you must create a Proposal. Once the proposal is created, you must select nodes to operate on. As discussed in the next sections, you may also edit the Proposal's attributes as needed.

Applying the Proposal tells Crowbar to deploy the proposal onto the nodes. While deploying, nodes return to the Ready state when deployment is completed. Once a proposal has become an Active Role, you cannot edit it. You must delete the Role and repeat the Apply process.

Several barclamps, noteably OpenStack Nova and Swift, require multiple nodes before they can be deployed. A proposal will not being to deploy until it has sufficient capacity.

You must edit a barclamp in the Crowbar databag in Chef to change the attribute defaults of a barclamp when not using the Crowbar UI.
At the time a proposal is applied, Crowbar updates the Run List of the selected nodes.

The Nova barclamp is *not* currently enabled to allow multiple proposals.

5.2 PXE State Machine

While crowbar brings systems up the systems run through a series of states each of which preforms different actions on the systems. This is controlled by the Provisioner barclamp and the status of any of the systems is indicated by the icon next to the representation of the system on the Dashboard page. You can see a description of the state which a system is in by hovering over the status icon.

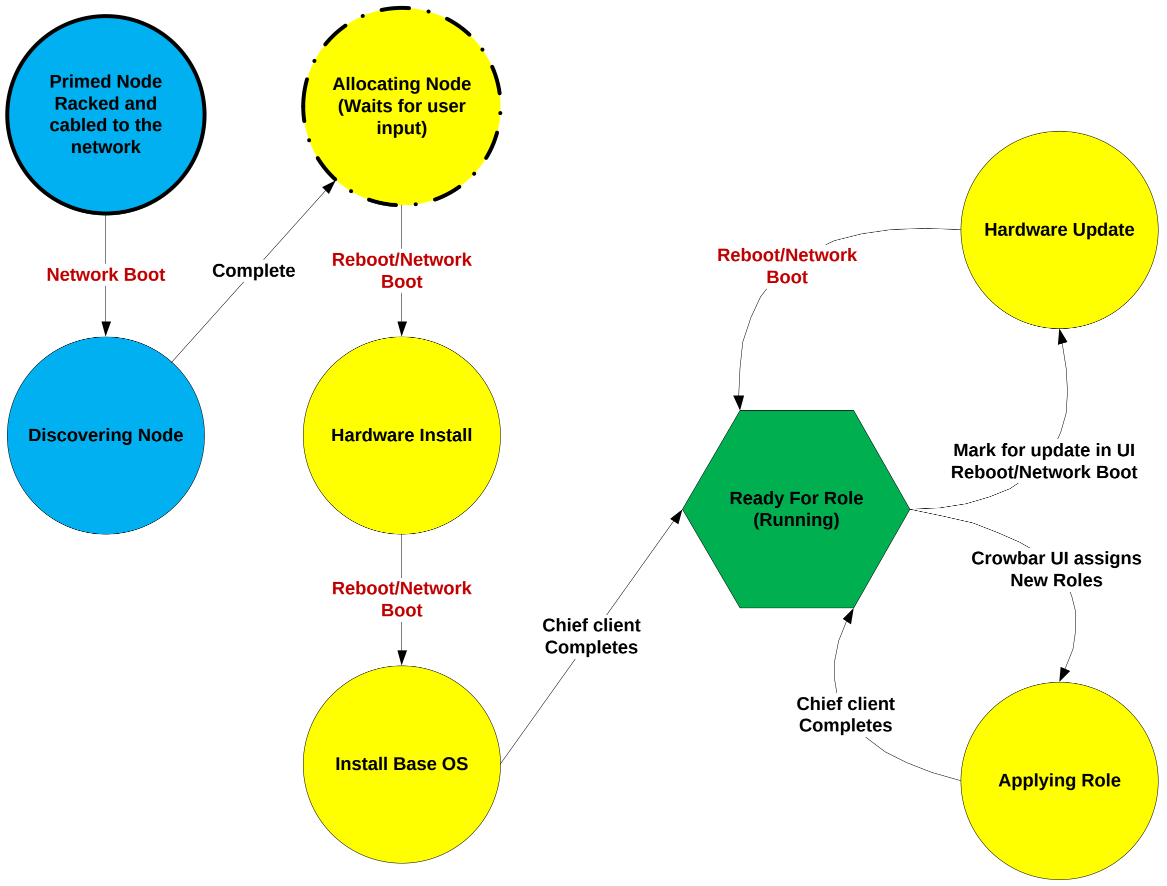


Figure 2 shows the states a system will progress through in order to fully provision a system

Discovering Node

During the Discovering Node state, the node will network boot a CentOS LiveCD image to make sure that the node can run some basic components. Once this is complete, the liveCD image will register with the Admin node to declare its state. The admin node adds the machine to the Chef database, allocates a BMC and Admin network address for the node.

Allocating Node

By default the nodes will pause after the discovered state. The node awaits allocation either manually through the API (CLI or UI) or the being included in a proposal that is committed to the system. Since the node is not in the ready state, the proposal will be queue until the newly allocated node achieves ready. Allocation is required to allow the system to use what the node is going to become to influence the configuration of the bios and raid systems.

Hardware Install

The LiveCD image will reboot forcing another network boot. During the Hardware Install state, the node will network boot a CentOS LiveCD image to ensure that the BIOS/BMC/RAID controller are up-to-date. The CentOS LiveCD will also update the BIOS and RAID configurations. Once this is complete, the liveCD image will register with the Admin node to declare its state. The admin node updates the Chef database and resets the DHCP entry to installation state.

Base OS Install

LiveCD image will reboot forcing another network boot. The node boots into a network installation of Ubuntu. The installation process reboots the node. Upon reboot and completing the one-time setup script, the system transitions to the Ready for Role state. The first time chef-client is run, the node will get a base set of configuration (nagios client, ganglia client, ntp client, dns resolver).

Ready for Role

This is the default state for normal operation. The node runs the chef client periodically to ensure that it is up to date. It is waiting for changes to its configuration. If the node reboots, the node will network boot the local boot image again and return to this state.

Hardware Update

To transition to this state, the admin will mark the node as needing hardware updates in the crowbar UI and then reboot the node. The node will network boot and do the same actions as the Hardware Install state. Once this is complete, the liveCD image will register with the Admin node to declare its state. The admin node updates the Chef database and resets the DHCP entry to installation state. The return state is back to the Ready for Role state. This state is to allow for the update of BIOS without having to re-install the system.

Applying Role

This is a transient state that represents the running chef-client. It is the time that the node is applying new configuration before returning to Ready for Role. The transition to this state happens periodically or can be forced by the admin node to run chef-client.

5.3 Barclamps

The Barclamp page shows a list of all available barclamps (see table below). Selecting a barclamp displays the details for the selected barclamp.

The details page shows the Proposals created and the Active Roles deployed for the barclamp. You can jump directory to the relevant proposal or role by clicking on its name.

From the barclamp details page, you may create a new proposal for the system.

Naming for proposals is limited to letters and numbers only (not spaces). Capitalization is allowed.

This limitation is necessary because activated proposals are created as roles in Chef and follow a prescribed naming convention.

As of July 2011, the following barclamps are included with the Dell EULA version Crowbar. Exceptions for the open source version are noted.

Barclamp	Function / Comments	Role	License
Crowbar	The roles and recipes to set up the barclamp framework. References other barclamps. Modify the default proposal to change the Usernames and passwords for access to the Crowbar UI.	Core	Apache 2
Deployer	Initial classification system for the Crowbar environment (aka the state machine)	Core	Apache 2
Provisioner	The roles and recipes to set up the provisioning server and a base environment for all nodes	Core	Apache 2
Network	Instantiates network interfaces on the crowbar managed systems. Also manages the address pool.	Core	Apache 2
RAID	Sets up LSI RAID controllers in a variety of configurations. If missing, can be performed manually.	Core / Optional	Dell EULA
BIOS	Configures BIOS options for Dell PowerEdge C servers. If missing, can be performed manually.	Core / Optional	Dell EULA
IPMI	Allows management of the IP Management Interface (IPMI) on servers when the BMC network is enabled.	Extendable	Apache 2
NTP	Common NTP service for the cluster (required for secure access). An NTP server can be specified.	Extendable	Apache 2
DNS	Manages the DNS subsystem for the cluster	Extendable	Apache 2
Logging	Centralized logging system based on syslog	Extendable	Apache 2

Barclamp	Function / Comments	Role	License
Nova	Installs and configures the Cactus release of Openstack Nova. It relies upon the network and glance barclamps for normal operation.	OpenStack	Apache 2
Swift	part of Openstack, and provides a distributed blob storage	OpenStack	Apache 2
Glance	Glance service (Nova image management) for the cloud	OpenStack	Apache 2
Nagios	System monitoring service for the cluster that can be used by other barclamps	Optional	GPL 2
Ganglia	Performance monitoring service for the cluster that can be used by other barclamps	Optional	BSD
Test	Provides a shell for writing tests against	Internal	Apache 2

5.4 Crowbar Barclamp

The Crowbar Barclamp provides the roles and recipes to set up the barclamp framework.

The crowbar barclamp initializes the system, creates initial instances of other barclamps defined in its configuration, and creates the users to access the crowbar API and UI. Any number of barclamp instances can be started. By default, the system starts a network, ganglia, nagios, ntp, dns, provisioner, deployer, ipmi, raid, and BIOS barclamp based upon their default configurations. The initialization function of the crowbar barclamp works exactly like the other barclamps. A proposal is created and can be committed during installation.

The main post-installation function is to provide the main transition entry point for the system. All barclamps' transition functions can be called directly, but the crowbar barclamp calls these in an order specified in its configuration which is determined by their priority. The default unspecified priority is 100. The special cases are the provisioner, which is last, and the deployer and network, which are first and second.

Crowbar Barclamp parameters.

Name	Default	Description
barclamps	A list of all barclamps we ship	A list of supported barclamps. This is not used in the product currently.
instances	The starting barclamps using their default configurations	A map of barclamp names that reference a list of json files (default is special to mean to take the defaults) that represent starting barclamp instances to create
run_order	A map of barclamp names to integer priorities	The map is used to define the order that the barclamps are called when handling a transition request. lower is higher priority. Unspecifed items are assumed to have a priority of 100.
users	A map of users - containing crowbar	This map defines the users allowed to access crowbar's UI and REST API.

The users map contains a map. The key is the user name and the rest of the required fields are:

Name	Description
password	Clear text password of the user
description	A description of the user.

5.5 Deployer Barclamp

The Deployer provides an initial classification system for the Crowbar environment. As nodes are discovered, the Deployer ensures that discovery tools are run on the node by ensuring that the deployer-client role is assigned to the node, the results of that discovery are classified, and the node's attributes are updated to reflect its potential usage. The deployer will also build a map of valid and usable disks.

The deployer gives the primary name to the node at the discovered state. The names default to the letter 'd' and the mac address (with dashes instead of colons). The deployer also allocates the admin and bmc addresses from the network barclamp.

The deployer also defines and provides the node's configuration for raid and bios. These values are assigned part of the hardware-installing state transition. The deployer uses a list of role name patterns that define what the raid and bios configurations should be. These are applied as values in the node attributes under crowbar -> hardware. bios_set can in the set of "Virtualization" or "Storage". raid_set can be in the set of JBODOnly and SingleRaid10.

The deployer is also responsible for manipulating the run-list during the hardware-installing and update (or hardware-updating) states. The run list should only include bios, raid, and ipmi operations.

The deployer also controls the allocate flag on the node. The allocate flag is used to pause the node during after discovery. The node will wait for it to be allocated to contain. The deployer has a configuration option to indicate if the allocate flag should be set to false (and cause a pause) or just allocate all nodes.

Deployer Barclamp parameters.

Name	Default	Description
bios_map	A list of default setings for bios and raid for swift and nova	The map defines a list of patterns that would apply a configuration setting for bios and raid.
use_allocate	true	A boolean value - true indicates that a pause should be injected after the discovered state to allow the admin to

	accept and allocate the node.

The bios_map entries are maps with the following keys:

Name	Description
pattern	Regular expression applied to the role names on the node.
bios_set	The bios set of parameters to apply. Values are: Virtualization or Storage
raid_set	The raid set of parameters to apply. Values are: JBODOnly or SingeRaid10

5.6 Provisioner Barclamp

The Provisioner provides the roles and recipes to set up the provisioning server and a base environment for all provisioned nodes. The Provisioner also provides the transition entry point for nodes that need to have DHCP transitions done. The Provisioner assumes that addressing will be handled outside of this barclamp.

The following parameters are defined for the Provisioner barclamp.

Name	Default	Description
default_user	openstack	User to create for external login
default_password	unset	Clear text password to use for external login
default_password_hash	Hash of openstack	MD5 hash of password to use for external login. printf 'password' | mkpassed -s -m md5 will generate the hash.
web_port	8091	The default web port that the repository web server uses
use_local_security	true	This defaults the security updates path in the install to use the admin node instead of the internet.
dhcp	map	This is a map that contains the DHCP parameters (lease-time and state_machine)
lease-time	60	The number of seconds a DHCP lease is valid for the system
state_machine	map	This is the state machine that DHCP server will use in this

		instance of the barclamp

While neither is required, one of default_password or default_password_hash is required.

5.7 Network Barclamp

The Network barclamp provides two functions for the system. The first is a common role to instantiate network interfaces on the crowbar managed systems. The other function is address pool management.

The network interfaces are controlled by the network role that is applied by the barclamp as a node transition to "installed". Based upon assigned addresses, the network recipe will create the apprioriate single, dual, or team mode interface sets.

The network assignment function is handled by the creation of an API extension of the base barclamp. The barclamp adds the allocate_ip REST API call. This function allocates an IP address from a requested network and updates the node's attributes and the network's data bag. The available networks (and their parameters) are defined in the configuration for the barclamp.

Modification of the following parameters should only be done at install time.

Network configuration options are:

Name	Default	Description
mode	single	A string value of either single, dual, or team. This specifies the default network interface construction model.
teaming	map	A map of values specific to teaming
networks	map	A map of networks that this barclamp should manage

The teaming sub-parameters are:

Name	Default	Description
mode	6	The default teaming algorithm to use for the bonding driver in Linux

The system provides the following default networks.

Name	Usage	Notes
admin	Private network for node to node communication	A router, if wanted, is external to the system. This network must be owned by the crowbar system to run DHCP on.
bmc	Private network for bmc communication	This can be the same as the admin network by using the ranges to limit what IP goes where. A router, if wanted, is external to the system.
bmc_vlan	Private network for admin nodes on the bmc network	This must be the same as the bmc network and have the same vlan. This will be used to generate a vlan tagged interface on the admin nodes that can access the bmc lan.

storage	Private network for storage traffic	A router, if wanted, is external to the system
public	Public network for crowbar and other components	A router, if wanted, is external to the system.
nova_fixed	Public network for nova Virtual Machines	The nova-network node acts as a router. This must be completely owned by the nova system.
nova_floating	Broken	deprecated - most likely to be replaced by nova config.

Each network has the following parameters:

Name	Default	Description
vlan	Integer	The vlan to use on the switch and interfaces for this network
use_vlan	true	A value of true indicates that the vlan should applied to the interface. A value of false assumes that the node will receive untagged traffic for this network.
add_bridge	false	indicates if the network should have a bridge built on top of it. The bridge will be br. This is mostly for Nova compute.
subnet	IP Address	The subnet for this network
netmask	Netmask	The netmask for this network
router	IP Address	The default router for this network
broadcast	IP Address	The default broadcast address for this network
ranges	map	This contains a map of strings to start and stop values for network. This allows for sub-ranges with the network for specific uses. e.g. dhcp, admin, bmc, hosts.

The range map has a string key that is the name and map defining the range.

Name	Type	Description
start	IP Address	First address in the range, inclusive
end	IP Address	Last address in the range, inclusive

⚠ Settings in the Network barclamp should not be changed after the installation of the Admin Node.

5.8 RAID Barclamp

RAID = Redundant Array Of independent Disks. this basically means that a RAID controller makes (or can) multiple disks look like 1 big/smart/safe disk.

The RAID controller can support up to 2 separate RAID volumes of either RAID0, RAID1, RAID1E or RAID10. Any disk not included in a RAID volume, are directly exposed to the Operating System (also known as JBOD = just a bunch of disks).

The crowbar code ensures that the configuration on the RAID controller matches that specified via the crowbar configuration.

The parts that determine the configuration for a node are:

- A set of chef data bags which contain the RAID configuration (in databags/crowbar-data). The defaults are "Storage" and "SingleRaid10"
- An attribute on the chef node of the machine which identifies which databag (described above) should be appled to this node (the attribute is node[:crowbar][:hardware][:raid_set], and it should include the name of a data bag).
- Crowbar (the deployer barclamp) sets the above property when a node is allocated to a proposal.

When invoked, the recipe uses a LWRP to inspect the current configuration on the system, and compare it to the desired state. If the 2 diverge, the code will:

- delete any raid-sets that are not required any more
- allocate available disks among the desired raid sets, according to the "order" attribute.
- issue commands to apply the config.

5.9 BIOS Barclamp

The BIOS barclamp provides the following specific control features:

BIOS:

- Uploading a known BIOS firmware into flash.
- Setting parameters to defined a set, based on the machine's role

5.10 IPMI Barclamp

The IPMI barclamp configures IPMI access on platforms that support it. Specifically it configures:

LAN parameters (ip address, netmask gateway)

User credentials.

The IPMI Barclamp has a couple of list parameters.

Name	Description
bmc_enable	Controls if the barclamp attempts to do anything
debug	turns on more verbose output

5.11 NTP Barclamp

The NTP Barclamp provides a common NTP service for the cluster. An NTP server or servers can be specified and all other nodes will be clients of them.

The NTP Barclamp has a couple of parameters.

Name	Default	Description
external_servers	empty list	A list of IP addresses or hostnames that should be used as external ntp servers. Hostname can be used if the DNS barclamp is configured to have access to an external resolver.
admin_ip_eval	"Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").address"	The ruby eval expression that returns the admin IP address of a node.

If you are setting up an external server it can take up to 5 min for the nodes to sync with the server. Systems should not be rebooted during this process. If systems are rebooted they will pause as the sync occurs.

5.12 Logging Barclamp

The Logging Barclamp provides a centralized logging system based on syslog. The barclamp enables a centralized log server that can then forward information to external syslog servers. The crowbar install process sends logs to the admin node by default, but the configuration from the logging barclamp overrides this initial configuration.

The Barclamp has some configuration parameters

Name	Default	Description
external_servers	Empty list	A list of IP addresses for the logging-server to forward logs to

5.13 Nova Barclamp

The Nova Barclamp installs and configures the Openstack Nova component. It relies upon the network and glance barclamps for normal operation.

The Nova Proposal requires that you have previously deployed a Glance proposal. You must have at LEAST 3 NODES before you can apply the Nova proposal.

Nova Barclamp parameters. A slash indicates a hash level.

Name	Default	Description
use_barclamp	true	A placeholder value for indicating that we are in a barclamp. Lame, but hey. Will probably go away.
libvirt_type	kvm	The type of hypervisor to use. qemu is the one to use in virtualization mode. Other options are available but untested.
network_type	flat	The networking model to use for nova. Options are `flat`, `flatdhcp`, and `dhcpvlan`.
flat_network/flat_network_bridge	br500	The bridge to use for the main network in flat mode
flat_network/flat_injected	true	Should the system attempt to inject networking information into the image?
flat_dhcp_network/flat_network_bridge	br500	The bridge to use for the main network in flatdhcp mode
flat_dhcp_network/flat_network_dhcp_start	10	The starting address of the DHCP range. This is used to reserve some addresses out of the pool.
flat_dhcp_network/flat_interface	eth0.500	The interface that the flat network is really on. This may or may not be needed.
dhcp_vlan_network/vlan_interface	eth0	The interface that vlans should be created upon
dhcp_vlan_network/vlan_start	1000	The starting vlan ID
dhcp_vlan_network/vpn_start	10000	The starting VPN port number for accessing the project's vlan.
num_networks	1	The number of networks to cut the fixed range up into. Really only used for dhcpvlan, i think
network_size	256	Power-of-2 number that is the size of each network to give to a project. Really only used for dhcpvlan, I think
user	nova	User to run the nova services as

user_group	nova	Group gather things together under
project	admin	The default initial admin project
access_key	auto-generated	EC2 and other access key for the initial admin user
secret_key	auto-generated	EC2 and other access key for the initial admin user
user_dir	/var/lib/nova	The default user directory for the nova user.
images	two amis	A list of URLs to get images. The defaults are the ones on the admin node.
rabbit/user	nova	The user to access the rabbit MQ server
rabbit/password	auto-generated	The password to access the rabbit MQ server
rabbit/vhost	/nova	The exchange for nova messages
rabbit/port	unset	The port to use for the rabbitmq server. Unset to allow the system to choose its default.
db/user	nova	The user to access the database server
db/password	auto-generated	The password to access the database server
db/database	nova	The name of the nova database.

There are additional parameters specified in the nova attributes file that haven't been replicated in the barclamp configuration file.

- Nova Networking

This section is called out separately because of its complexity and scope.

Nova has 3 networking modes available. They are integrated with the networking barclamp modes and 10gb networking. Initially, the nova modes will be described and then the integration with the networking barclamp will be described. While the three modes are different, they use a consistent underlying networking mode.

The Nova Barclamp assumes that the Networking Barclamp is running and handling the networks. It will use the information about the topology from the networking barclamp. Nova assumes that three networks are available, admin, public, and nova_fixed_network. Currently, a nova_floating_network is defined, but not used or required. That is evolving in the community. The admin network is for service communication. Public is used for outward facing public services of Nova. Nova_fixed_network is used for the VMs. It is assumed that nova_fixed_network is a completely owned subnet. public network may be partially presented. In all cases, the nova-network node will act as the router between public and nova_fixed_network.

- Flat Network

In this mode, the nova-compute gets an address from nova-network and injects that address into the VMs image (linux-only). In our setup, this image then pulls from nova-api to get its custom configuration files (keys and stuff).

To make this work nicer, the nova-network node acts as the router between the public facing networks. This is NOT part of normal Nova Flat Network. It is part of Nova for the other modes.

The network parameters should be to chop the nova_fixed_network into a single network with all addresses available. This is specified in the num_networks and network_size parameters. The DHCP start parameter of the nova_fixed_network acts a reservation section of addresses for that range. This allows to remove shared network pieces.

This mode will use the interfaces specified by the network barclamp. By default, it will use eth0.500 for fixed network, eth0.300 for public, and eth0 for admin. Bridges will be created as appropriate. If the network mode is changed in the network barclamp, it will switch to using the teamed network or dual nic for the fixed network and public. If these are on 10g, those interfaces will be pulled as appropriate.

- Flat DHCP Network

In this mode, the nova-compute doesn't modify the VM image or allocates an address. The VM is assumed to run DHCP to get its address and then talk to nova-api for custom configuration. The nova-network node runs dnsmasq to provide DHCP to the nova_fixed_network.

The network parameters should be to chop the nova_fixed_network into a single network with all addresses available. This is specified in the num_networks and network_size parameters. The DHCP start parameter of the nova_fixed_network acts the DHCP starting address for the nova-network agent.

This mode will use the interfaces specified by the network barclamp. By default, it will use eth0.500 for fixed network, eth0.300 for public, and eth0 for admin. Bridges will be created as appropriate. If the network mode is changed in the network barclamp, it will switch to using the teamed network or dual nic for the fixed network and public. If these are on 10g, those interfaces will be pulled as appropriate.

- VLAN DHCP Network

In this mode, the nova-compute doesn't modify the VM image and uses dnsmasq to hand out addresses. There are two big differences from Flat DHCP. First, a custom VLAN is allocated for each project. The project gets the next free vlan after the nova_fixed_network vlan and each project gets a subset of the nova_fixed_network vlan defined by the num_networks and network_size. So, if nova_fixed_network is a class B and num_networks is 1024 and network_size is 64, this will support 1024 projects. The external assumption is that the networking barclamp has setup the single, dual or teamed network and that the reserved vlans are already trunk by the switch. Default switch configs already trunk all vlans to all ports.

This mode will use the interfaces specified by the network barclamp. By default, it will use eth0.500 for fixed network, eth0.300 for public, and eth0 for admin. Bridges will be created as appropriate. If the network mode is changed in the network barclamp, it will switch to using the teamed network or dual nic for the fixed network and public. If these are on 10g, those interfaces will be pulled as appropriate. Additional VLANs will start at 501 and continue upwards.

The second big difference is the introduction of a VPN VM that is managed by nova-network to provide access to the network. The nova-network node acts as a router/firewall for the network and routes to the VPN to allow access to the VMs. The VM is controlled and managed by nova-network. It is often called cloud-pipe. The cloud-pipe image needs to be in glance and set-up in a way that openvpn configuration can be injected into it.

⚠ **THIS MODE IS STILL UNDER DEVELOPMENT**

5.14 Swift Barclamp

Swift is part of Openstack, and provides a distributed blob storage. This barclamp installs swift.

Swift includes the following components:

- Proxy node- provides the API to the cluster, including authentication.
- Storage nodes - provide storage for cluster.

Swift Barclamp Parameters

Name	Default	Description
cluster_hash	random generated	cluster hash is shared among all nodes in a swift cluster.can be generated using od -t x8 -N 8 -A n </dev/random
cluster_admin_pw	swauth	super user password - used for managing users.
replicas	1	how many replicas should be made for each object
zones	2	how many zones are in this cluster (should be >= # of replicas)
min_part_hours	1	minimum amount of time a partition should stay put, in hours
partitions	18	number of bits to represent the partitions count
user	swift	the uid to be used for swift processes
group	swift	the gid to be used for swift processes
admin_ip_expr	`"Chef::Recipe::Barclamp::Inventory .get_network_by_type(node, \"admin\").address"`	where to find IP for admin use

storage_ip_expr	`"Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").address"`	where to find IP for admin use
disk_enum_expr	`"node[\"crowbar\"][\"disks\"]"`	expression to find a hash of possible disks to be used.
disk_test_expr	`"v[\"usage\"] == 'Storage'"`	expression accepting a k,v pair for evaluation. if expression returns true, then the disk will be used. by default, use any sdX or hdX that is not the first one (which will hold the OS).
disk_zone_assign_expr	`'$DISK_CNT | =0; $DISK_CNT= $DISK_CNT+1 ;[ $DISK_CNT % node[:swift][:zones] , 99]'`	see below

Note that the _expr parameters are meant to allow the recipes supporting the swift barclamp to be customized in their logic. Customers can embed alternative logic to e.g. disk usage decisions by the system. The same parameters allow the recipes to be utilized in a non-crowbar environment.

- Disk zone assignment expression

An expression to classify disks into zone's and assign them a weight. The expression will return either:

- nil: the disk is not included in the ring
- otherwise an array of [zone, weight]. Zone is an integer representing the zone # for the disk is expected, weight is the weight of the disk

The default expression below just assigns disks in a round robin fashion.

The expression is evaluated with the following context:

- node - the Chef node hash
- params - a hash with the following keys:
  - :ring=> one of "object", "account" or "container"
  - :disk=> disk partition information as created in disks.rb,contains: :name (e.g sdb) :size either :remaining (= all the disk) or an actual byte count.

Parameters should not be changed after committing the proposal. Addition or removal of devices from the proposal will be dynamicaly reconfigured in the swift configuration after the initial proposal has been committed.

5.15 Glance Barclamp

The glance barclamp provides the Glance service for the cloud

The Barclamp has a couple of list parameters.

Name	Default	Description
backup_type	"file"	The type of backing store to use. Options are file, s3, swift. Only supported

	currently is file.	

5.16 Nagios Barclamp

The Nagios Barclamp provides a common Nagios service for the cluster. A Nagios server or servers can be specified and all other nodes will be clients of them. The barclamp will attempt to direct all traffic over the admin network.

The Barclamp has a couple of parameters.

Name	Default	Description
admin_interface_eval	Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").interface	The ruby eval expressiong that returns the admin interface of the node.
admin_ip_eval	"Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").address"	The ruby eval expression that returns the admin IP address of a node.

5.17 Ganglia Barclamp

The Ganglia Barclamp provides a common Ganglia service for the cluster. An Ganglia server or servers can be specified and all other nodes will be clients of them.

The Barclamp has the following parameters.

Name	Default	Description
interface_eval	Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").interface	The ruby evaluation string that gets the interface of the admin interface.

5.18 Test Barclamp

The Test barclamp provides a shell for writing tests against. It allows for failures can be injected and other barclamps can be validated against it.

The Barclamp has a couple of list parameters.

Name	Description
barclamps	A list of supported barclamps that are used as the return value for the barclamp list API call
instances	A map of barclamp names that reference a list of json files (default is special to mean to take the defaults) that represent starting barclamp instances to create

5.19 Proposals

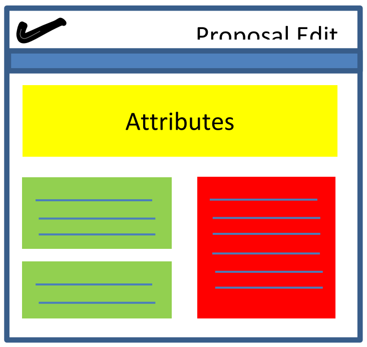
The Proposals page allows you to review and deploy a proposal.

The list page shows you all proposals in the system. If a barclamp has multiple proposals then the barclamp is listed twice but with different proposals in the name column. The proposals created automatically by Crowbar will be named "default."

Crowbar stores barclamps in the Chef under the Crowbar data bag using *bc-template-[barclamp]* as the naming pattern. When a proposal is created, the instance copy is also stored in the Crowbar data bag. Only active proposals have roles created for them.

The status of each proposal is indicated by a status light. A red blinking status light indicates that user attention is required. The proposals status updates automatically without a refresh.

Status	Icon	Comment	User Action
Ready	🟢	Proposal has been deployed	Ready for use.
Pending	🟡	Queued for deployment	Needs additional resources and/or configuration.
Not Ready	🔴 blinking	Proposal is being configured	Normal part of application process



Selecting the barclamp on the list will navigate to the barclamp details page.

Selecting the name of the proposal on the list will open the Edit Proposal page. All proposals have two primary edit areas: Attributes (yellow in figure) and Deployment. Attributes are configurable data that is used by the Chef recipes. Deployment is the Chef roles and nodes assigned to those roles.

Since each barclamp has unique attributes and roles you should consult the documentation for each barclamp if you plan to change its defaults.

Each barclamp may provide a custom editor for its attributes and node deployment information. The typical custom editor allows you to set attribute values using a form and drag and drop nodes from the available list (red on figure) into the roles associated with the barclamp (green on figure). Each barclamp may have specific logic that requires minimum or maximums for node assignments.

While most barclamps coordinate with Chef to perform node deployements, Crowbar includes some special function barclamps that can be used to change how Crowbar operates.
These barclamps include: crowbar, deployer, and provisioner.
Editing these barclamps is beyond the scope of this document.

If the barclamp does not have customer editor, Crowbar will automatically use a raw JSON editor. You can also use this view if you want to see the entire configuration details. Selecting the Raw view option on the right side of the Attributes or Deployment panel will open the JSON editor for that section. This option allows you to directly edit the JSON configuration details for the proposal. This option is typically used when developing new barclamps or for advanced users only.

When you have finished editing the proposal, you may save or apply it. Save will retain your configuration settings. Apply commits your proposal and instructs Crowbar to begin deploying your proposal. Apply does not automatically save your proposal. Deleting a proposal will remove it from the system and you will lose your configuration.

If you attempt to apply a proposal that does not have sufficient nodes then Crowbar will show that proposal as queued. This is likely to happen if you select nodes that have not been allocated.

When you apply a proposal, Crowbar creates Chef roles and puts them in the run list of the selected nodes.

5.20 Active Roles

The Active Roles page shows barclamp proposals that have been applied in the system.

Selecting a barclamp from the Active Roles list will take you to the associated Barclamp details page.

Selecting a proposal from the Active Roles list will display a view page of details for the proposal. The details page shows the attributes specific to the proposal and the nodes which have Chef roles related to the proposal. Selecting a node will navigate you directly to the node's detail page.

Reminder: You cannot edit an active role, you must edit its proposal and reapply.

During a typical deployment, Crowbar will create and apply the core barclamps for the system; consequently, even a newly installed system will show up to 10 “default” applied proposals.

The Active Roles detail page can show you which nodes are running key infrastructure components. For example, on the Nova barclamp, the node(s) running the *nova-multi-controller* role have the API and mangement components of Nova.

Crowbar uses a naming pattern for Roles that allows you to quickly figure out which barclamp and proposal is being applied to a node’s run list in Chef.
The instantiated barclamp naming pattern is *[barclamp]-config-[proposal]*.
Barclamps then use additional roles to control node proposal membership (aka the Run List)

6 Supplemental Material

6.1 System Verification

As a final step, it’s important to be able to verify that your deployment has succeeded. Crowbar does *not* provide specific feedback or updates to confirm that the Chef recipes were successfully deployed.

You should consult the getting started guide and barclamps specific to your system for details on verification of deployment.

6.2 Deactivating Barclamps

As of July 2011, there are no methods and few controls to remove barclamps. The best practice is to remove them before completing the Admin node installation. If you must remove them after installation:

1. In Crowbar, delete all proposals associated with the barclamp
2. In Chef, delete the barclamp in the Crowbar data bag
3. Provide an offering of iron enriched carrots to the bunny

6.3 Creating Barclamps

Information on how to develop your own barclamp is available in the Developers guide. If you are interested in creating, extending or contributing barclamps, please contact our team at openstack@dell.com.

6.4 Gathering Log Information

In order to help facilitate troubleshooting of the environment a utility to gather logs has been provided. Browse to http://<Admin_ip>/support/logs. This will create a tar archive of the relevant logs and ask the user for a location to save the resulting archive.

Depending on the size of the logs to be gathered this utility may take a while to run