Dell | Apache Hadoop Solution

Crowbar User Guide

Notes, Cautions, and Warnings

✎ **NOTE:** A NOTE indicates important information that helps you make better use of your computer

△ **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.**

Information in this document is subject to change without notice.

© 2011 Dell Inc. All rights reserved.

Reproduction of these materials is allowed under the Apache 2 license.

Information in this document is subject to change without notice.

© 2011 Dell Inc. All rights reserved.

Reproduction of these materials in any manner whatsoever without the written permission of Dell Inc. is strictly forbidden.

Trademarks used in this text: Dell™, the DELL logo, CloudEra™, Nagios™, Ganglia™, Opscode Chef™, OpenStack™, Canonical Ubuntu™ , VmWare™, Dell Precision™, OptiPlex™, Latitude™, PowerEdge™, PowerVault™, PowerConnect™, OpenManage™, EqualLogic™, KACE™, FlexAddress™ and Vostro™ are trademarks of Dell Inc. Intel®, Pentium®, Xeon®, Core™ and Celeron® are registered trademarks of Intel Corporation in the U.S. and other countries. AMD® is a registered trademark and AMD Opteron™, AMD Phenom™, and AMD Sempron™ are trademarks of Advanced Micro Devices, Inc. Microsoft®, Windows®, Windows Server®, MS-DOS® and Windows Vista® are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Red Hat Enterprise Linux® and Enterprise Linux® are registered trademarks of Red Hat, Inc. in the United States and/or other countries. Novell® is a registered trademark and SUSE ™ is a trademark of Novell Inc. in the United States and other countries. Oracle® is a registered trademark of Oracle Corporation and/or its affiliates. Citrix®, Xen®, XenServer® and XenMotion® are either registered trademarks or trademarks of Citrix Systems, Inc. in the United States and/or other countries. VMware®, Virtual SMP®, vMotion®, vCenter®, and vSphere® are registered trademarks or trademarks of VMWare, Inc. in the United States or other countries.

Other trademarks and trade names may be used in this publication to refer to either the entities claiming the marks and names or their products. Dell Inc. disclaims any proprietary interest in trademarks and trade names other than its own.

October 2011

Contents

Introduction

This document provides instructions you to use when deploying **Hadoop** components.

Concepts

The purpose of this guide is to explain the user interface of Crowbar. Use the Crowbar User Guide for assistance with installing Crowbar and configuring the target system.

Note: Concepts beyond the scope of this guide are introduced as needed in notes and references to other documentation.

Cloudera Apache Distribution

The focus of this guide is the use of Crowbar, *not* Hadoop. While Crowbar includes substantial components to assist in the deployment of Hadoop, its operational aspects are completely independent of Hadoop. For more information about Hadoop, please visit http://hadoop.apache.org/.

cloudera This guide provides this additional information about Cloudera as notes flagged with the Cloudera logo. For detailed operational support for Hadoop, we suggest visiting the Coudera documentation web site at http://www.cloudera.com.

Opscode Chef Server

Crowbar makes extensive use of Opscode Chef Server**,** http://opscode.com. To explain Crowbar actions, you should 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 is included.

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

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 are identified.

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

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

The Crowbar Solution

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 operating system.
- 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 your existing environments, Crowbar allows customization of its components. You can disable the default monitoring tools (Nagios and Ganglia) if you prefer to use your 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, you create 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; for example: upstream DNS resolvers.
- List of machines in the deployment that fulfill the different roles in the function.
- Internal system information.

When provisioning a function, you 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. You can edit these proposals before installing these services on the admin node.

For Hadoop and eco-system components (Pig, Hive, Sqoop, and Zookeeper), employ Crowbar tools to construct a starting proposal and then edit it to fit the specific requirements for your environment. Once the proposal is ready, you commit the proposal.

When a proposal is committed, Crowbar configures the Chef server and other components in the systems (TFTP, DHCP, and so on) 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 makes sure that machines are hooked into the key infrastructure services. Critical services (for example 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.

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 describe more advanced customization options.

Node Provisioning

When a new node is added to the system, set it up to allow PXE booting. Once a machine is powered on, Crowbar uses 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.

- 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, used by the rest of Crowbar. Addresses can be allocated from different pools, meant for different usages; for example: 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 makes sure that when a new machine is added to the deployment, it has a default entry added to the DNS zone. The default host name is the machine's MAC address prefixed by the letter, 'd' (for example: d00-a4-b3-c2-d1-e0.yourdomain.com).

Nagios™

Nagios monitors provisioned services for availability. Each cluster instance is represented as a host group, letting you quickly identify the health of a given instance.

Install the Nagios server on the admin node. It is configured to monitor all the nodes in the system. As new nodes are brought online, Crowbar dynamically updates Nagios to include them.

Ganglia™

Ganglia monitors the installed cluster for capacity and performance information, letting you 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.

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.

Managing Growth

The Dell | Cloudera Reference Architecture is organized into three components, for sizing as the Hadoop environment grows. From smallest to largest, they are:

- Rack
- Pod
- Cluster

Each has specific characteristics and sizing considerations. You can scale the environment by adding additional capacity as needed, without the need to replace any existing components.

Rack

A rack is the smallest component in a Hadoop environment, and consists of all of the power, network cabling, and two Ethernet switches required to support up to 20 data nodes. These nodes should utilize their own power connectivity and data center space – separate from other racks – and be treated as a fault zone.

Pod

A pod is a single set of stacked Ethernet switches. For the Dell | Cloudera Reference Architecture, both the maximum and minimum are six. A pod consists of the administration and operation infrastructure to support three racks.

Cluster

A cluster is a set of greater than one pod, up to a maximum of 12 pods. A cluster is a set of Hadoop nodes that share the same Network Node and management tools for operating the Hadoop environment.

Note: Please see the Dell | Cloudera Solution Reference Architecture Guide for more information.

Default Networks

The default networks are presented in the following table.

Table 1-1: Default Networks

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 the entire network.	100	Not tagged
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
Edge 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

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

Layout

Due to the nature of Crowbar's network layout, addresses are assigned to a whole network based upon interface, Network Type (Production, Management, and External) and teaming type.

Table 1-2: Master/Secondary (Admin) Name Nodes Network Connections

Interface	Network Type	Teaming Type
BMC	Management LAN	Single
LOM1	Production LAN	Teamed
LOM2	Production LAN	Teamed
Eth1	Production LAN	Teamed
Eth2	Management LAN	Single

Table 1-3: Edge Nodes Network Connections

Interface	Network Type	Teaming Type
BMC	Management LAN	Single
LOM1	Production LAN	Teamed 1
LOM2	Production LAN	Teamed 1
Eth1	External LAN	Teamed 2
Eth2	External LAN	Teamed 2

Table 1-4: Slave Nodes Network Connections

Interface	Network Type	Teaming Type
BMC	Management LAN	Single
LOM1	Production LAN	Teamed 1
LOM2	Production LAN	Teamed 1

IP Addressing

The IP address can be assigned in this fashion, using large subnets to support many machines on the production network. The Management network is a Class C network with 254 IP addresses; the Production network is what is known as a /23 with 512 IP addresses. In each network, the first 10 IP addresses are reserved for switches, routers, and firewalls.

Note: Each network's ".1" address is reserved for the network gateway.

Table 1-5: IP Addressing Schema

LAN	Network	Subnet	Gateway	Reserved
Management LAN	172.16.0.0	255.255.255.0	172.16.0.1	0.1 – 0.10
Production LAN	172.16.2.0	255.255.254.0	172.16.2.1	2.1-2.20
Name Nodes	DHCP Allocated			

LAN	Network	Subnet	Gateway	Reserved
Slave Nodes	DHCP Allocated			
External LAN	TBD by Customer			

Rack Awareness

With the network set up using Top of Rack (ToR) switches, Rack Awareness can be programmed by using the Chef information about which switch the LOM1 is plugged into. A simple script has been added to the Hadoop configuration to pull the information out of Chef, and then use it for Rack Awareness.

Table 1-6: Pod 1 IP Addressing Layout

Network: 172.16.0.0 Netmask: 255.255.252.0

Multicast: 172.16.0.0 Broadcast 172.16.3.255

Pod	Rack Number	Network	Server Type	IP Range	Subnet Mask	Gateway
1	1	Production	Slave	172.16.0.1-42	255.255.252.0	172.16.0.1
1	2	Production	Slave	172.16.1. 1-42	255.255.252.0	172.16.0.1
1	3	Production	Slave	172.16.2. 1-42	255.255.252.0	172.16.0.1
1		Production	Master Name	172.16.3.1-19	255.255.252.0	172.16.0.1
1		Production	Secondary Name	172.16.3.20-30	255.255.252.0	172.16.0.1
1		Production	Edge Node	172.16.3.41-50	255.255.252.0	172.16.0.1
1	1	BMC	Slave	172.16.0.200-242	255.255.252.0	172.16.0.1
1	2	BMC	Slave	172.16.1.200-242	255.255.252.0	172.16.0.1
1	3	BMC	Slave	172.16.2.200.242	255.255.252.0	172.16.0.1
1		BMC	Master Name	172.16.3.201-219	255.255.252.0	172.16.0.1
1		BMC	Secondary Name	172.16.3.220-230	255.255.252.0	172.16.0.1
1		BMC	Edge Node	172.16.3.231-250	255.255.252.0	172.16.0.1

Table 1-7: Pod 2 IP Addressing Layout

Network: 172.16.0.0 Netmask: 255.255.252.0

Multicast: 172.16.0.0 Broadcast 172.16.3.255

Pod	Rack Number	Network	Server Type	IP Range	Subnet Mask	Gateway

2	1	Production	Slave	172.16.4.1-42	255.255.252.0	172.16.4.1
2	2	Production	Slave	172.16.5. 1-42	255.255.252.0	172.16.4.1
2	3	Production	Slave	172.16.6. 1-42	255.255.252.0	172.16.4.1
2		Production	Master Name	172.16.7.1-19	255.255.252.0	172.16.4.1
2		Production	Secondary Name	172.16.7.20-30	255.255.252.0	172.16.4.1
2		Production	Edge Node	172.16.7.41-50	255.255.252.0	172.16.4.1
2	1	BMC	Slave	172.16.4.200-242	255.255.252.0	172.16.4.1
2	2	BMC	Slave	172.16.5.200-242	255.255.252.0	172.16.4.1
2	3	BMC	Slave	172.16.6.200.242	255.255.252.0	172.16.4.1
2		BMC	Master Name	172.16.7.201-219	255.255.252.0	172.16.4.1
2		BMC	Secondary Name	172.16.7.220-230	255.255.252.0	172.16.4.1
2		BMC	Edge Node	172.16.7.231-250	255.255.252.0	172.16.4.1
2		External	Edge Node	TBD by Customer	TBD	TBD

For more information about Hadoop, please visit http://hadoop.apache.org/.

User Interface

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

Using Crowbar

Crowbar is delivered as a Web application available on the admin node using 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.

Note: 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.

Table 2-1: Service URLs

Service	URL	Credentials
SSH	root@192.168.124.10	crowbar
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
Hadoop Jobtracker UI (master name node)	http://192.168.124.81:50070	

Note: 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.

Crowbar has two primary concepts for users Nodes and Barclamps. Before talking about the UI, it's important to understand how they are used by Crowbar.

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.

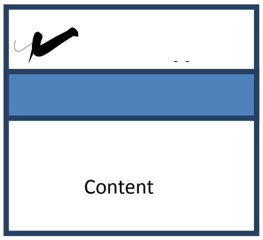
Barclamps

- 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 is reviewed, you must activate it before it becomes an Active 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 is discussed in more detail as we explore the user interface. Information about using, creating, and extending barclamps is included in the Supplemental Material section.

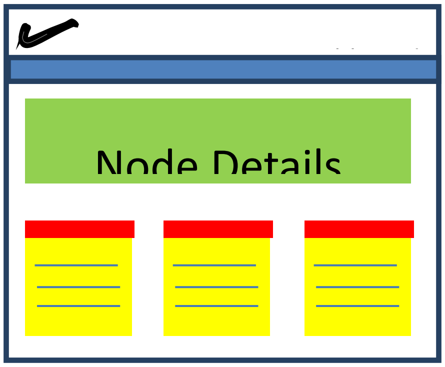
General Layout



The menu for Crowbar is displayed on the upper right side of the page. Clicking on one of the menu sections causes related content to display 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 you should not have to refresh the page to get the most current information.

Nodes (system dashboard)



The Dashboard shows all the nodes in the system and lets you manipulate their power and configuration states.

Node Switch Groups

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

Note: If you use a consistent pattern for connecting nodes to switches then the Crowbar display matches 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.

Table 2-2: Deployment States

Status	Icon	Comment	User Action
Ready	🟢	Requested services provisioned.	
Pending	🟡	Holding state after discovery	Node waiting to be activated
Not Ready	🔴 blinking	Crowbar and 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.

Node Details

Clicking on a node's name displays 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 expands 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 Barclamps

The Barclamps page lets you create, edit, review and deploy proposals for Barclamps. These activities are the way that Crowbar decides which nodes to deploy and how to configure them.

Note: Previous versions of Crowbar split the Barclamp lifecycle into multiple pages. All three phases of the lifecycle (barclamp, proposal, active role) are now represented together on the Barclamps page.

Barclamp List

The Barclamps page shows a list of all available barclamps (see "Included Barclamps" table). The barclamps are represented as the blue lines in the figure to the right. Expanding a barclamp (by clicking ▷) displays the associated proposals for the selected barclamp (red box in the figure). You jump directly to the relevant proposal by clicking on its name under its barclamp.

A barclamp will show the status of the proposals that are attached to it using a status light (see table below). If multiple proposals are assigned, then multiple light are displayed. If there are no proposals, a diamond is displayed. Hovering over the light will show you the name and status of the matching proposal. The proposals status updates automatically without a refresh.

Table 4-59: Proposal Status

Status	Icon	Next Step	Comment	User Interaction
None	◇	Create	No proposal has assigned to the barclamp	Create a proposal for the barclamp if desired
User Input	●	Apply	Proposal waiting for user input and activation	Edit the proposal. Apply proposal after review.
Ready	🟢	Deactivate	Proposal has been deployed.	Ready for use.
Pending	🟡	Dequeue	Queued for deployment.	Needs additional resources and/or configuration.
Not Ready	🔴 blinking	None	Proposal is being configured.	Normal part of application process.

From the Barclamp list, you may take actions on the proposals based on their state as shown in the table above. Please review the *Life Cycle* section for more information about the different proposal states.

All core barclamps automatically create "*default*" proposals and do not allow users to create additional proposal. Other barclamps allow the creation of multiple proposals. These additional proposals can be used to manage deployment configurations or control which parts of the system are active in which barclamps.

To create a new proposal, expand the barclamp row to expose the create form for new proposal. You must supply a name for the proposal but descriptions are optional. Clicking create will take you to the proposal editor (*details below*). The create form will not be shown for barclamps that only allow a single proposal.

Note: Naming for proposals is limited to lowercase letters and numbers only (not "spaces). Capitalization is automatic for formatting only.

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

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.

Proposal View/Edit

Selecting a proposal from on the list navigates to the proposal details page.

Note: If a proposal is active, you will be initially taken to a read only view of the proposal. It is acceptable to edit an active proposal and re-apply. To access the proposal editor from the read only view, click the edit button.

Selecting the name of the proposal on the list opens 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 lets you 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.

Note: While most barclamps coordinate with Chef to perform node deployments, Crowbar includes some special function barclamps that can be used to change how Crowbar operates.

If the barclamp does not have customer editor or your browser does not support the editor, Crowbar automatically uses 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 opens the JSON editor for that section. This option lets you 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 retains your configuration settings. Apply save and then applies your proposal so that Crowbar to begin deploying the barclamp on the selected nodes. Deleting a proposal removes it from the system and you lose your configuration.

Note: If you attempt to apply a proposal that does not have sufficient nodes then Crowbar shows 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 then puts them into the run list of the selected nodes.

Crowbar uses a naming pattern for Roles that let you 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)

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

Note: 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 cause the node status to be unknown. These buttons are only available if the system is able to successfully configure the BMC on the target system.

Table 2-3: Details Panel Buttons

Button	Action	Useful When
Identify	Causes the identify light to blink for 15 seconds.	Trying to identify a node within a rack.
Power On	Sends a power on signal to the BMC of the selected system.	Remotely powering on a system.
Shutdown	Sends a power off signal to the BMC of the selected system.	Remotely powering on a system.
Reboot	Sends 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) 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:

Table 2-4: Buttons on Bottom of Details Panel

Button	Action	Config Lost?	Reboot?	Useful When
Delete	Completely removes all records of the node from the Crowbar/Chef database. If a node is deleted, it is rediscovered if it reboots.	Yes	No	Removing nodes.
Reset	Removes 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	Reimages 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) lets you make per node decisions about how the node is deployed.

Note: You **must** allocate the node to let Crowbar 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. In addition, use it to simulate white listing.

To Allocate a node, manually allocate the node from the edit page or you may include that node in an applied barclamp proposal.

For more information about Hadoop, please visit http://hadoop.apache.org/

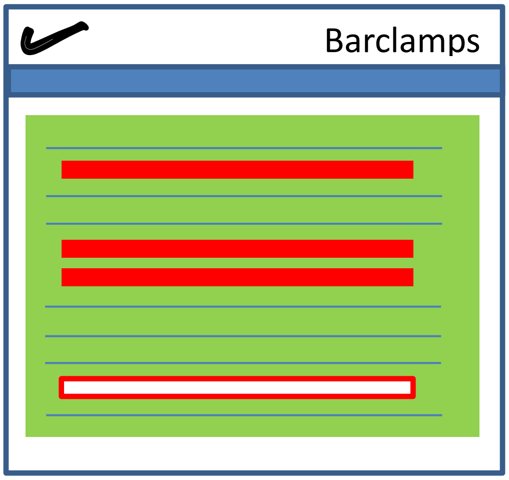
Barclamps

The Barclamps page lets you create, edit, review and deploy proposals for Barclamps. These activities are the way that Crowbar decides which nodes to deploy and how to configure them.

Note: Previous versions of Crowbar split the Barclamp lifecycle into multiple pages. All three phases of the lifecycle (barclamp, proposal, active role) are now represented together on the Barclamps page.

Barclamp List

The Barclamps page shows a list of all available barclamps (see "Included Barclamps" table). The barclamps are represented as the blue lines in the figure to the right. Expanding a barclamp (by clicking ▷) displays the associated proposals for the selected barclamp (red box in the figure). You jump directly to the relevant proposal by clicking on its name under its barclamp.



A barclamp will show the status of the proposals that are attached to it using a status light (see table below). If multiple proposals are assigned, then multiple light are displayed. If there are no proposals, a diamond is displayed. Hovering over the light will show you the name and status of the matching proposal. The proposals status updates automatically without a refresh.

Table 4-59: Proposal Status

Status	Icon	Next Step	Comment	User Interaction
None	◇	Create	No proposal has assigned to the barclamp	Create a proposal for the barclamp if desired
User Input	●	Apply	Proposal waiting for user input and activation	Edit the proposal. Apply proposal after review.
Ready	🟢	Deactivate	Proposal has been deployed.	Ready for use.
Pending	🟡	Dequeue	Queued for deployment.	Needs additional resources and/or configuration.
Not Ready	🔴 blinking	None	Proposal is being configured.	Normal part of application process.

From the Barclamp list, you may take actions on the proposals based on their state as shown in the table above. Please review the *Life Cycle* section for more information about the different proposal states.

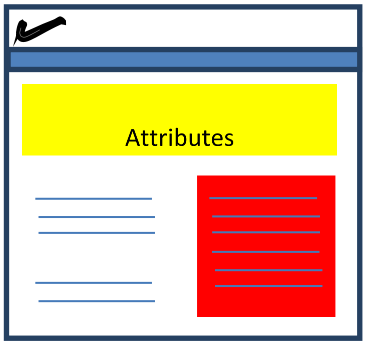
All core barclamps automatically create "*default*" proposals and do not allow users to create additional proposal. Other barclamps allow the creation of multiple proposals. These additional proposals can be used to manage deployment configurations or control which parts of the system are active in which barclamps.

To create a new proposal, expand the barclamp row to expose the create form for new proposal. You must supply a name for the proposal but descriptions are optional. Clicking create will take you to the proposal editor (*details below*). The create form will not be shown for barclamps that only allow a single proposal.

Note: Naming for proposals is limited to lowercase letters and numbers only (not "spaces). Capitalization is automatic for formatting only. This limitation is necessary because activated proposals are created as roles in Chef and follow a prescribed naming convention.

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.

Proposal View/Edit



Selecting a proposal from on the list navigates to the proposal details page.

Note: If a proposal is active, you will be initially taken to a read only view of the proposal. It is acceptable to edit an active proposal and re-apply. To access the proposal editor from the read only view, click the edit button.

Selecting the name of the proposal on the list opens 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 lets you 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.

Note: While most barclamps coordinate with Chef to perform node deployments, Crowbar includes some special function barclamps that can be used to change how Crowbar operates.

If the barclamp does not have customer editor or your browser does not support the editor, Crowbar automatically uses 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 opens the JSON editor for that section. This option lets you 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 retains your configuration settings. Apply save and then applies your proposal so that Crowbar to begin deploying the barclamp on the selected nodes. Deleting a proposal removes it from the system and you lose your configuration.

Note: If you attempt to apply a proposal that does not have sufficient nodes then Crowbar shows 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 then puts them into the run list of the selected nodes.

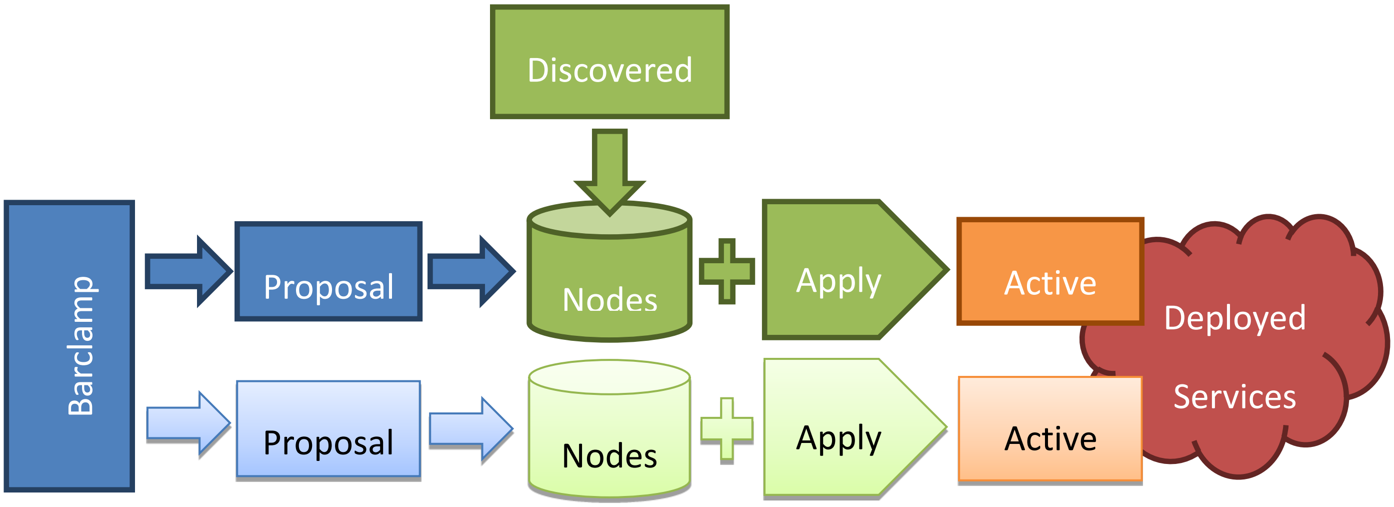
Crowbar uses a naming pattern for Roles that let you quickly figure out which barclamp and proposal is being applied to a node's run list in Chef. The instantiated barclamp naming pattern is [barclamp]-config-[proposal]. Barclamps then use additional roles to control node proposal membership (aka the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

Deployment Functions

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

Life Cycle

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



Crowbar progresses all deployments through a fixed lifecycle. It is important to understand this lifecycle to use Crowbar.

Figure 1 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.

Note: Many barclamps, require multiple nodes before they can be deployed. A proposal is not deploy until it has sufficient capacity.

You must edit a barclamp in the Crowbar data bag 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 following table shows options for changes to proposals based on their current state

State	Icon	Forwards	Backwards
None	◇	Create	None

User Input	●	Apply	Delete
Ready	🟢	Apply	Deactivate
Pending	🟡	None: System Working	Dequeue
Not Ready	🔴 blinking	None: System Working	None: System Working

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

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.

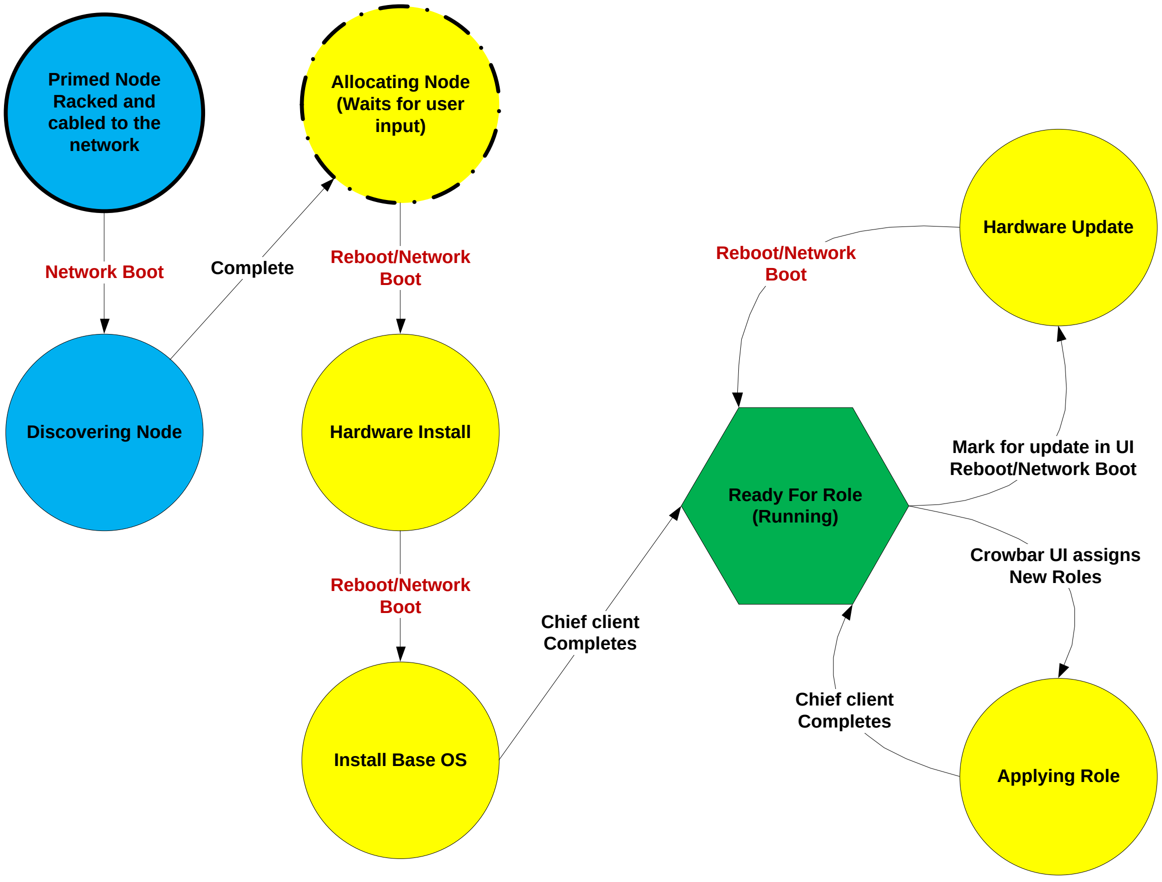


Figure 2 shows the states a system progresses through to fully provision a system

Discovering Node

During the Discovering Node state, the node network boots a CentOS LiveCD image to make sure that the node can run some basic components. Once this is complete, the LiveCD image registers 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 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 is queue until the newly allocated node achieves is 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 reboots forcing another network boot. During the Hardware Install state, the node network boots a CentOS LiveCD image to make sure that the BIOS/BMC/RAID controller are up-to-date. The CentOS LiveCD also update the BIOS and RAID configurations. Once this is complete, the LiveCD image registers 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 reboots 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 gets 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 network boots the local boot image again and return to this state.

Hardware Update

To transition to this state, you mark the node as needing hardware updates in the Crowbar UI and then reboot the node. The node network boots and does the same actions as the Hardware Install state. Once this is complete, the liveCD image registers 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 allows 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.

Included Barclamps

Table 4-1: Barclamps

Barclamp	Function / Comments	Role
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
Deployer	Initial classification system for the Crowbar environment (aka the state machine)	Core
Provisioner	The roles and recipes to set up the provisioning server and a base environment for all nodes	Core
Network	Instantiates network interfaces on the Crowbar managed systems. Also manages the address pool.	Core
RAID	Sets up LSI RAID controllers in a variety of configurations. If missing, you can performed it manually.	Core / Optional
BIOS	Configures BIOS options for Dell PowerEdge C servers. If missing, can be performed manually.	Core / Optional
IPMI	Allows management of the IP Management Interface (IPMI) on servers when the BMC network is enabled.	Extendable
NTP	Common NTP service for the cluster (required for secure access). An NTP server can be specified.	Extendable
DNS	Manages the DNS subsystem for the cluster	Extendable
Logging	Centralized logging system based on syslog	Extendable
Nagios	System monitoring service for the cluster that can be used by other barclamps	Optional
Ganglia	Performance monitoring service for the cluster that can be used by other barclamps	Optional
Hadoop	Common libraries and utilities that provides the basic Hadoop runtime environment (HDFS/Map Reduce). A set of components and interfaces which implements a distributed filesystem and provides general I/O access for the hadoop framework (serialization, Java RPC and persistent data storage).	Optional
Hive	Data warehouse that infrastructure provides SQL based data summarization and ad hoc querying.	Optional
Pig	Platform for analyzing large data sets that consists of a high-level language for expressing data algorithms.	Optional
Sqoop	SQL based command-line tool to assist with HDFS data import/export (SQL-to-Hadoop).	Optional
ZooKeeper	High-performance coordination service for distributed applications. ZooKeeper provides primitives such as distributed locks which can be used for building large scale	Optional

Barclamp	Function / Comments	Role
	distributed processing applications.	
Test	Provides a shell for writing tests against	Internal

Details about individual barclamps are including in the Barclamps section below.

Barclamp Details

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.

Barclamp Parameters

Table 4-2: 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. Unspecified 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:

Table 4-3: User Name Key

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

Deployer Barclamp

The Deployer provides an initial classification system for the Crowbar environment. As nodes are discovered, the Deployer makes sure that discovery tools are run on the node by making sure 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 also builds 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 Deplorer also allocates the admin and BMC addresses from the network barclamp.

In addition, the Deployer 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 waits 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.

Barclamp Parameters

Table 4-4: Deployer Barclamp Parameters

Name	Default	Description
bios_map	A list of default settings 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.

Table 4-5: BIOS Map Entry 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.

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 is handled outside of this barclamp.

Barclamp Parameters

Table 4-6: Provisioner Barclamp Parameters

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 uses in this instance of the barclamp.

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

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 creates the appropriate 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.

Barclamp Parameters

Table 4-7: Network Configuration Options

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.

Table 4-8: Teaming Sub-Parameters

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

Table 4-9: 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.
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 is 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.

Table 4-10: Network 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 apply to the interface. A value of false assumes that the node receives 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. For example: DHCP, admin, BMC, hosts.

Table 4-11: Range Map String Key

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.

RAID Barclamp

RAID = Redundant Array Of independent Disks. This 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 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 makes sure that the configuration on the RAID controller matches that specified using 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 data bags/crowbar-data). The defaults are SingleRaid10 and JBODOnly.

- An attribute on the chef node of the machine which identifies which data bag (described above) should be applied 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.

Note: Hadoop runs a Raid10 disk configuration on the master name node and secondary name node. Hadoop runs a JBOD disk configuration on the slave nodes (Hadoop data nodes)..

When invoked, the recipe uses a LWRP to inspect the current configuration on the system, and compare it to the desired state. If the two 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 configuration.

Barclamp Parameters

BIOS Barclamp

The BIOS barclamp provides the following specific control features:

- Uploading a known BIOS firmware into flash. Setting parameters to defined a set, based on the machine's role.
- IPMI Barclamp
- LAN parameters (IP address, netmask gateway) and User credentials.
- The IPMI Barclamp has a couple of list parameters.

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

Barclamp Parameters

Table 4-12: IPMI Barclamp Parameters

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

Note: The IPMI barclamp ads the following credentials to the BMC user configuration: username/password.

NTP Barclamp

The NTP Barclamp provides a common NTP service for the cluster. You can specify an NTP server or servers and all other nodes are clients of them.

Barclamp Parameters

Table 4-13: NTP Barclamp Parameters

Name	Default	Description

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.

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

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.

Barclamp Parameters

Table 4-14: Logging Barclamp Configuration Parameters

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

Hadoop Barclamp

The Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programmatic driven processing model. Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Rather than rely on hardware to deliver high-availability, the Hadoop library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Hadoop is ideal for organizations with a growing need to store and process massive application datasets. It enables applications to work with thousands of nodes and petabytes of data. This Crowbar barclamp provides the ability to deploy and maintain Hadoop cluster Admin, Master, Slave and Edge nodes. It also provides the capability to configure and deliver Hadoop HDFS and MapReduce components.

Hadoop Overview

- **Hadoop Core**: The common libraries and utilities that provide the basic Hadoop runtime environment. A set of components and interfaces which implement a distributed filesystem and provide general I/O access for the Hadoop framework (serialization, Java RPC and persistent data storage).
- **Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce**: A software framework for distributed processing of large data sets on compute clusters.

HDFS Overview

HDFS is a core component of the Hadoop framework and it is the underlining Hadoop virtual file system.

HDFS has the underlining concepts of three node classes:

- Master name node which is responsible for managing the file system metadata and transactions.
- Secondary name node which is responsible for checkpointing the name node's persistent state.
- Slave data nodes which are responsible for actually storing the file data.

HDFS stores files as a series of blocks, each of which is by default 64MB in size. A block is the unit of storage for data nodes. Data nodes store and retrieve blocks, and have no concept of the files that are composed of these blocks.

- **Master name node** - The master name node, is responsible for managing the filesystem metadata and data node mappings. The master name node holds that mapping from files to blocks, which it stores in memory as well as in a persistent metadata store on disk (e.g image file and edit log). The mapping between blocks and the data nodes they reside on is not stored persistently. Instead, it is stored in the name node's memory, and is built up from the periodic block reports that data nodes send to the name node. This is the primary metadata store for the cluster.
- **Secondary name node** - The secondary name is a checkpointing mechanism which can take over the primary name node's functional aspects for this particular operation. During system operation, the name node maintains two on-disk data structures to represent the filesystem state (image file and an edit log). The image file is a checkpoint of the filesystem metadata at a point in time and the edit log is a transactional redo log of every filesystem metadata mutation since the image file was created. Incoming changes to the filesystem metadata (such as creating a new file) are written to the edit log2. When the name node starts, it reconstructs the current state by replaying the edit log. To ensure that the log doesn't grow without bound, at periodic intervals the edit log is rolled, and a new checkpoint is created by applying the old edit log to the image. This process is performed by the secondary name node daemon, often on a separate machine to the primary since creating a checkpoint has similar memory requirements to the name node itself. A side effect of the checkpointing mechanism is that the secondary holds an out-of-date copy of the primary's persistent state, which, in extreme cases, can be used to recover the filesystem's state. Blocks are stored on the underlying filesystem of the data node, as opposed to the data node managing their own storage, as native kernel level filesystems do.
- **Slave nodes** - Slave nodes are the distributed collection points for data storage. Functioning data nodes send heartbeats to the name node every 3 seconds. This mechanism forms the communication channel between data node and name node. Occasionally, the name node will piggyback a command to a data node on the heartbeat response, for instance, "send a copy of block e to data node b". One of the first things that a data node does upon startup is send a block report to the name node, and this allows the name node to rapidly form a picture of the block distribution across the cluster.

Cluster Deployment Topology

The Crowbar Hadoop barclamp framework has expanded the concept of node deployment beyond HDFS in order to introduce the notion of a cloud edge node. The cloud edge node sits on the cloud boundary and provides the underlining interface between the data/processing capacity within the Hadoop cluster and the data consumer/end user environment. The addition of the cloud edge node serves to off-load external transactional processing requests from the data nodes and provide an additional level of security between the private cloud and the outside world.

- **Master and secondary name nodes** - Runs all the basic services needed to manage the HDFS data storage and MapReduce task distribution and tracking.
- **Slave node** - Runs all the services required to store blocks of data on the local hard drives and execute processing tasks against that data.
- **Edge Node** – Provides the interface between a data and processing capacity available in the Hadoop cluster and a user of that capacity. Most of the Hadoop eco-system sub-components run on the edge node.
- **Admin Node** – Provides cluster deployment/management capabilities and is used to deploy Hadoop to all the nodes in the cluster (The Crowbar administration node).

Note: The Hadoop **secondary name node** runs on the **Crowbar admin node.**

The typical deployment process is:

1. Deploy the core components: HDFS, MapReduce on the NameNodes and DataNodes.
2. Bring up the cluster.

3. Deploy the EdgeNode.
4. Deploy the eco-system sub-components within the cluster ZooKeeper on a slave node or Sqoop, Hive, Pig on the EdgeNode.

There may be cases when the customer may choose to deploy the add-on services on slave nodes or even the name nodes. Also when the cluster grows beyond a certain size the customer may need to run the name node (the HDFS manager) daemon and the JobTracker (the MapReduce manager) on different machines. In that case the customer needs to be able to terminate/uninstall the JobTracker daemon on the original name node and bring it up on the new JobTracker machine.

Eco-system sub-components need to be able to scale independently of the cluster configuration and/or capacity. For example, there may be cases when the data transfer capacity between the Hadoop and data warehouse (i.e. Aster Data) may exceed the max capacity of a single edge node. Adding a second edge node may be a viable alternative.

The design of the Hadoop add-on services need to separate the core Hadoop components (HDFS, MapReduce) from the add-on services and allow the customer to manipulate and deploy the services configuration that makes sense in his environment regardless of the size or topology of the actual Hadoop cluster.

Hadoop Barclamp Parameters

Table 4-21: Hadoop Barclamp configuration Parameters

Name	Description	Required	Default
fs_checkpoint_dir	Determines where on the local filesystem the DFS secondary name node should store the temporary images to merge. If this is a comma-delimited list of directories then the image is replicated in all of the directories for redundancy.	true	${hadoop.tmp.dir}/dfs/namesecondary
fs_checkpoint_edits_dir	Determines where on the local filesystem the DFS secondary name node should store the temporary edits to merge. If this is a comma-delimited list of directories then teh edits is replicated in all of the directories for redundancy. Default value is same as fs.checkpoint.dir.	true	${fs.checkpoint.dir}
fs_checkpoint_period	The number of seconds between two periodic checkpoints.	true	3600
fs_checkpoint_size	The size of the current edit log (in bytes) that triggers a periodic checkpoint even if the fs.checkpoint.period hasn't expired.	true	67108864
fs_default_name	The name of the default filesystem. A URI whose scheme and authority determine the FileSystem implementation. The uri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class. The uri's authority is used to determine the host, port, etc. for a filesystem.	true	file:///

Name	Description	Required	Default
fs_file_impl	The FileSystem for file: uris.	true	org.apache.hadoop.fs.LocalFileSystem
fs_ftp_impl	The FileSystem for ftp: uris.	true	org.apache.hadoop.fs.ftp.FTPFileSystem
fs_har_impl	The FileSystem for Hadoop archives.	true	org.apache.hadoop.fs.HarFileSystem
fs_har_impl_disable_cache	Don't cache 'har' filesystem instances.	true	true
fs_hdfs_impl	The FileSystem for hdfs: uris.	true	org.apache.hadoop.hdfs.DistributedFileSystem
fs_hftp_impl		true	org.apache.hadoop.hdfs.HftpFileSystem
fs_hsftp_impl		true	org.apache.hadoop.hdfs.HsftpFileSystem
fs_kfs_impl	The FileSystem for kfs: uris.	true	org.apache.hadoop.fs.kfs.KosmosFileSystem
fs_ramfs_impl	The FileSystem for ramfs: uris.	true	org.apache.hadoop.fs.InMemoryFileSystem
fs_s3_block_size	Block size to use when writing files to S3.	true	67108864
fs_s3_buffer_dir	Determines where on the local filesystem the S3 filesystem should store files before sending them to S3 (or after retrieving them from S3).	true	${hadoop.tmp.dir}/s3
fs_s3_impl	The FileSystem for s3: uris.	true	org.apache.hadoop.fs.s3.S3FileSystem
fs_s3_maxRetries	The maximum number of retries for reading or writing files to S3, before we signal failure to the application.	true	4
fs_s3_sleepTimeSeconds	The number of seconds to sleep between each S3 retry.	true	10
fs_s3n_impl	The FileSystem for s3n: (Native S3) uris.	true	org.apache.hadoop.fs.s3native.NativeS3FileSystem
fs_trash_interval	Number of minutes between trash checkpoints. If zero, the trash feature is disabled.	false	
hadoop_http_filter_initializers	A comma separated list of class names. Each class in the list must extend org.apache.hadoop.http.FilterInitializer. The corresponding Filter will be initialized. Then, the Filter will be applied to all user facing jsp and servlet web pages. The ordering of the list defines the ordering of the filters.	false	
hadoop_logfile_count	The max number of log files.	true	10
hadoop_logfile_size	The max size of each log file.	true	10000000
hadoop_native_lib	Should native hadoop libraries, if present, be used.	true	true
hadoop_rpc_socket_factory_	SocketFactory to use to connect to a DFS.	false	

Name	Description	Required	Default
class_ClientProtocol	If null or empty, use hadoop.rpc.socket.class.default. This socket factory is also used by DFSClient to create sockets to DataNodes.		
hadoop_rpc_socket_factory_class_default	Default SocketFactory to use. This parameter is expected to be formatted as "package.FactoryClassName".	true	org.apache.hadoop.net.StandardSocketFactory
hadoop_security_authentication	Possible values are simple (no authentication), and kerberos.	true	simple
hadoop_security_authorization	Is service-level authorization enabled?	true	false
hadoop_security_group_mapping	Class for user to group mapping (get groups for a given user).	true	org.apache.hadoop.security.ShellBasedUnixGroupsMapping
hadoop_security_uid_cache_secs	NativeIO maintains a cache from UID to UserName . This is the timeout for an entry in that cache.	true	14400
hadoop_socks_server	Address (host:port) of the SOCKS server to be used by the SocksSocketFactory.	false	
hadoop_tmp_dir	A base for other temporary directories.	true	/tmp/hadoop-${user.name}
hadoop_util_hash_type	The default implementation of Hash. Currently this can take one of the two values: 'murmur' to select MurmurHash and 'jenkins' to select JenkinsHash.	true	murmur
io_bytes_per_checksum	The number of bytes per checksum. Must not be larger than io.file.buffer.size.	true	512
io_compression_codecs	A list of the compression codec classes that can be used for compression/decompression.	true	org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec
io_file_buffer_size	The size of buffer for use in sequence files. The size of this buffer should probably be a multiple of hardware page size (4096 on Intel x86), and it determines how much data is buffered during read and write operations.	true	4096
io_mapfile_bloom_error_rate	The rate of false positives in BloomFilter -s used in BloomMapFile. As this value decreases, the size of BloomFilter -s increases exponentially. This value is the probability of encountering false positives (default is 0.5%).	true	0.005
io_mapfile_bloom_size	The size of BloomFilter -s used in BloomMapFile . Each time this many keys is appended the next BloomFilter will be created (inside a DynamicBloomFilter). Larger values minimize the number of filters, which slightly increases the performance, but may waste too much space if the total number of keys is usually much smaller than this number.	true	1048576

Name	Description	Required	Default
io_seqfile_compress_blocksize	The minimum block size for compression in block compressed SequenceFiles .	true	1000000
io_seqfile_lazydecompress	Should values of block-compressed SequenceFiles be decompressed only when necessary.	true	true
io_seqfile_sorter_recordlimit	The limit on number of records to be kept in memory in a spill in SequenceFiles .Sorter.	true	1000000
io_serializations	A list of serialization classes that can be used for obtaining serializers and deserializers.	true	org.apache.hadoop.io.serializer.WritableSerialization
io_skip_checksum_errors	If true, when a checksum error is encountered while reading a sequence file, entries are skipped, instead of throwing an exception.	true	false
ipc_client_connect_max_retries	Indicates the number of retries a client will make to establish a server connection.	true	10
ipc_client_connection_maxidletime	The maximum time in msec after which a client will bring down the connection to the server.	true	10000
ipc_client_idlethreshold	Defines the threshold number of connections after which connections will be inspected for idleness.	true	4000
ipc_client_kill_max	Defines the maximum number of clients to disconnect in one go.	true	10
ipc_client_tcpnodelay	Turn on/off Nagle's algorithm for the TCP socket connection on the client. Setting to true disables the algorithm and may decrease latency with a cost of more/smaller packets.	true	false
ipc_server_listen_queue_size	Indicates the length of the listen queue for servers accepting client connections.	true	128
ipc_server_tcpnodelay	Turn on/off Nagle's algorithm for the TCP socket connection on the server. Setting to true disables the algorithm and may decrease latency with a cost of more/smaller packets.	true	false
local_cache_size	The limit on the size of cache you want to keep, set by default to 10GB. This will act as a soft limit on the cache directory for out of band data.	true	10737418240
topology_node_switch_mapping_impl	The default implementation of the DNSToSwitchMapping . It invokes a script specified in topology.script.file.name to resolve node names. If the value for topology.script.file.name is not set, the default value of DEFAULT_RACK is returned for all node names.	true	org.apache.hadoop.net.ScriptBasedMapping
topology_script_file_name	The script name that should be invoked to resolve DNS names to NetworkTopology	false	

Name	Description	Required	Default
	names. Example: the script would take host.foobar as an argument, and return /rack1 as the output.		
topology_script_number_args	The max number of args that the script configured with topology.script.file.name should be run with. Each arg is an IP address.	true	100
webinterface_private_actions	If set to true, the web interfaces of JT and NN may contain actions, such as kill job, delete file, etc., that should not be exposed to public. Enable this option if the interfaces are only reachable by those who have the right authorization.	true	false

Table 4-22. Hadoop Barclamp Environment Parameters

Name	Description	Required	Default
hadoop_datanode_opts	Command line configuration options for the data nodes.	false	
hadoop_heapsize	The maximum amount of heapsize to use, in MB (e.g. 1000MB). This is used to configure the heap size for the hadoop daemon. The default, value is 1000.	true	1000
hadoop_jobtracker_opts	Command line configuration options for the jobtracker.	false	
hadoop_log_dir	The directory where the daemons log files are stored. They are automatically created if they don't already exist.	true	/var/log/hadoop
hadoop_namenode_opts	Command line configuration options for the primary name node.	false	
hadoop_secondarynamenode_opts	Command line configuration options for the secondary name node.	false	
hadoop_tasktracker_opts	Command line configuration options for the tasktracker.	false	

Table 4-23: Hadoop Barclamp HDFS Parameters

Name	Description	Required	Default
dfs_access_time_precision	The access time for HDFS file is precise up to this value. The default value is 1 hour. Setting a value of 0 disables access times for HDFS.	true	3600000

Name	Description	Required	Default
dfs_balance_bandwidthPerSec	Specifies the maximum amount of bandwidth that each DataNode can utilize for the balancing purpose in term of the number of bytes per second.	true	1048576
dfs_block_access_key_update_interval	Interval in minutes at which NameNode updates its access keys.	true	600
dfs_block_access_token_enable	If "true", access tokens are used as capabilities for accessing DataNodes. If "false", no access tokens are checked on accessing DataNodes.	true	false
dfs_block_access_token_lifetime	The lifetime of access tokens in minutes.	true	600
dfs_block_size	The default block size for new files.	true	67108864
dfs_blockreport_initialDelay	Delay for first block report in seconds.	false	
dfs_blockreport_intervalMsec	Determines block reporting interval in milliseconds.	true	3600000
dfs_client_block_write_retries	The number of retries for writing blocks to the data nodes, before we signal failure to the application.	true	3
dfs_data_dir	Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.	true	${hadoop.tmp.dir}/dfs/data
dfs_datanode_address	The address where the DataNode server will listen to. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50010
dfs_datanode_data_dir_perm	Permissions for the directories on on the local filesystem where the DFS data node store its blocks. The permissions can either be octal or symbolic.	true	755
dfs_datanode_dns_interface	The name of the Network Interface from which a data node should report its IP address.	true	default
dfs_datanode_dns_nameserver	The host name or IP address of the name server (DNS) which a DataNode should use to determine the host name used by the NameNode for communication and display purposes.	true	default
dfs_datanode_du_reserved	Reserved space in bytes per volume. Always leave this much space free for non dfs use.	false	
dfs_datanode_failed_volumes_tolerated	The number of volumes that are allowed to fail before a DataNode stops offering service. By default any volume failure will cause a DataNode to shut down.	false	
dfs_datanode_handler_count	The number of server threads for the	true	3

Name	Description	Required	Default
	DataNode.		
dfs_datanode_http_address	The DataNode HTTP server address and port. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50075
dfs_datanode_https_address	The DataNode HTTPS server address and port.	true	0.0.0.0:50475
dfs_datanode_ipc_address	The DataNode IPC server address and port. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50020
dfs_default_chunk_view_size	The number of bytes to view for a file on the browser.	true	32768
dfs_df_interval	Disk usage statistics refresh interval in msec.	true	60000
dfs_heartbeat_interval	Determines DataNode heartbeat interval in seconds.	true	3
dfs_hosts	Names a file that contains a list of hosts that are permitted to connect to the NameNode. The full pathname of the file must be specified. If the value is empty, all hosts are permitted.	false	
dfs_hosts_exclude	Names a file that contains a list of hosts that are not permitted to connect to the NameNode. The full pathname of the file must be specified. If the value is empty, no hosts are excluded.	false	
dfs_http_address	The address and the base port where the dfs NameNode web UI will listen on. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50070
dfs_https_address		true	0.0.0.0:50470
dfs_https_client_keystore_resource	Resource file from which SSL client keystore information will be extracted.	true	ssl-client.xml
dfs_https_enable	Decide if HTTPS (SSL) is supported on HDFS.	true	false
dfs_https_need_client_auth	Whether SSL client certificate authentication is required.	true	false
dfs_https_server_keystore_resource	Resource file from which SSL server keystore information will be extracted.	true	ssl-server.xml
dfs_max_objects	The maximum number of files, directories and blocks DFS supports. A value of zero indicates no limit to the number of objects that DFS supports.	false	
dfs_name_dir	Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.	true	${hadoop.tmp.dir}/dfs/name
dfs_name_edits_dir	Determines where on the local filesystem	true	${dfs.name.dir}

Name	Description	Required	Default
	the DFS name node should store the transaction (edits) file. If this is a comma-delimited list of directories then the transaction file is replicated in all of the directories, for redundancy. Default value is same as dfs.name.dir.		
dfs_namenode_decommission_interval	Namenode periodicity in seconds to check if decommission is complete.	true	30
dfs_namenode_decommission_nodes_per_interval	The number of nodes NameNode checks if decommission is complete in each dfs.namenode.decommission.interval.	true	5
dfs_namenode_delegation_key_update_interval	The update interval for master key for delegation tokens in the NameNode in milliseconds.	true	86400000
dfs_namenode_delegation_token_max_lifetime	The maximum lifetime in milliseconds for which a delegation token is valid.	true	604800000
dfs_namenode_delegation_token_renew_interval	The renewal interval for delegation token in milliseconds.	true	86400000
dfs_namenode_handler_count	The number of server threads for the NameNode.	true	10
dfs_namenode_logging_level	The logging level for dfs namenode. Other values are "dir"(trac e namespace mutations), "block"(trace block under/over replications and block creations/deletions), or "all".	true	info
dfs_permissions	If "true", enable permission checking in HDFS. If "false", permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.	true	true
dfs_permissions_supergroup	The name of the group of super-users.	true	supergroup
dfs_replication	Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.	true	3
dfs_replication_considerLoad	Decide if chooseTarget considers the target's load or not.	true	true
dfs_replication_interval	The periodicity in seconds with which the NameNode computes replication work for DataNodes.	true	3
dfs_replication_max	Maximal block replication.	true	512
dfs_replication_min	Minimal block replication.	true	1
dfs_safemode_extension	Determines extension of safe mode in milliseconds after the threshold level is reached.	true	30000
dfs_safemode_threshold_pct	Specifies the percentage of blocks that	true	0.999f

Name	Description	Required	Default
	should satisfy the minimal replication requirement defined by dfs.replication.min. Values less than or equal to 0 mean not to start in safe mode. Values greater than 1 will make safe mode permanent.		
dfs_secondary_http_address	The secondary NameNode HTTP server address and port. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50090
dfs_support_append	Does HDFS allow appends to files? This is currently set to false because there are bugs in the "append code" and is not supported in any prodction cluster.	true	false
dfs_web_ugi	The user account used by the web interface. Syntax: USERNAME,GROUP1,GROUP2, ...	true	webuser,webgroup

Table 4-24: Hadoop Barclamp Map/Reduce Parameters

Name	Description	Required	Default
hadoop_job_history_location	If job tracker is static the history files are stored in this single well known place. If No value is set here, by default, it is in the local file system at ${hadoop.log.dir}/history.	false	
hadoop_job_history_user_location	User can specify a location to store the history files of a particular job. If nothing is specified, the logs are stored in output directory. The files are stored in "_logs/history/" in the directory. User can stop logging by giving the value "none".	false	
hadoop_rpc_socket_factory_class_JobSubmissionProtocol	SocketFactory to use to connect to a Map/Reduce master (JobTracker). If null or empty, then use hadoop.rpc.socket.class.default.	false	
io_map_index_skip	Number of index entries to skip between each entry. Zero by default. Setting this to values larger than zero can facilitate opening large map files using less memory.	false	
io_sort_factor	The number of streams to merge at once while sorting files. This determines the number of open file handles.	true	10
io_sort_mb	The total amount of buffer memory to use while sorting files, in megabytes. By default, gives each merge stream 1MB, which should minimize seeks.	true	100
io_sort_record_percent	The percentage of io.sort.mb dedicated to tracking record boundaries. Let this value	true	0.05

Name	Description	Required	Default
	be r, io.sort.mb be x. The maximum number of records collected before the collection thread must block is equal to (r * x) / 4.		
io_sort_spill_percent	The soft limit in either the buffer or record collection buffers. Once reached, a thread will begin to spill the contents to disk in the background. Note that this does not imply any chunking of data to the spill. A value less than 0.5 is not recommended.	true	0.80
job_end_retry_attempts	Indicates how many times Hadoop should attempt to contact the notification URL.	false	
job_end_retry_interval	Indicates time in milliseconds between notification URL retry calls.	true	30000
jobclient_output_filter	The filter for controlling the output of the task's userlogs sent to the console of the JobClient . The permissible options are: NONE, KILLED, FAILED, SUCCEEDED, and ALL.	true	FAILED
keep_failed_task_files	Controls whether the files for failed tasks be kept. This should only be used on jobs that are failing, because the storage is never reclaimed. It also prevents the map outputs from being erased from the reduce directory as they are consumed.	true	false
map_sort_class	The default sort class for sorting keys.	true	org.apache.hadoop.util.QuickSort
mapred_acls_enabled	Specifies whether ACLs should be checked for authorization of users for doing various queue and job level operations. ACLs are disabled by default. If enabled, access control checks are made by JobTracker and TaskTracker when requests are made by users for queue operations like submit job to a queue and kill a job in the queue and job operations like viewing the job-details (See mapreduce.job.acl-view-job) or for modifying the job (See mapreduce.job.acl-modify-job) using Map/Reduce APIs, RPCs or via the console and web user interfaces.	true	false
mapred_child_env	User added environment variables for the task tracker child processes. Example: 1) A=foo This will set the env variable A to foo 2) B=$B:c This is inherit tasktracker's B env variable.	false	
mapred_child_java_opts	Java opts for the task tracker child processes. The following symbol, if present, will be interpolated: @taskid@ is replaced by current TaskID . Any other occurrences of '@' will go unchanged. For example, to enable verbose gc logging to a file named for the taskid in /tmp and to set	true	-Xmx200m

Name	Description	Required	Default
	the heap maximum to be a gigabyte, pass a 'value' of: -Xmx1024m -verbose:gc -Xloggc:/tmp/@taskid@.gc The configuration variable mapred.child.ulimit can be used to control the maximum virtual memory of the child processes.		
mapred_child_tmp	To set the value of tmp directory for map and reduce tasks. If the value is an absolute path, it is directly assigned. Otherwise, it is prepended with task's working directory. The java tasks are executed with option -Djava.io.tmpdir='the absolute path of the tmp dir'. Pipes and streaming are set with environment variable, TMPDIR='the absolute path of the tmp dir'.	true	./tmp
mapred_child_ulimit	The maximum virtual memory, in KB, of a process launched by the Map-Reduce framework. This can be used to control both the Mapper/Reducer tasks and applications using Hadoop Pipes, Hadoop Streaming etc. By default it is left unspecified to let cluster admins control it via limits.conf and other such relevant mechanisms. Note: mapred.child.ulimit must be greater than or equal to the -Xmx passed to JavaVM , else the VM might not start.	false	
mapred_cluster_map_memory_mb	The size, in terms of virtual memory, of a single map slot in the Map-Reduce framework, used by the scheduler. A job can ask for multiple slots for a single map task via mapred.job.map.memory.mb, upto the limit specified by mapred.cluster.max.map.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off.	true	-1
mapred_cluster_max_map_memory_mb	The maximum size, in terms of virtual memory, of a single map task launched by the Map-Reduce framework, used by the scheduler. A job can ask for multiple slots for a single map task via mapred.job.map.memory.mb, upto the limit specified by mapred.cluster.max.map.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off.	true	-1
mapred_cluster_max_reduce_memory_mb	The maximum size, in terms of virtual memory, of a single reduce task launched by the Map-Reduce framework, used by the scheduler. A job can ask for multiple slots for a single reduce task via mapred.job.reduce.memory.mb, upto the	true	-1

Name	Description	Required	Default
	limit specified by mapred.cluster.max.reduce.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off.		
mapred_cluster_reduce_memory_mb	The size, in terms of virtual memory, of a single reduce slot in the Map-Reduce framework, used by the scheduler. A job can ask for multiple slots for a single reduce task via mapred.job.reduce.memory.mb, upto the limit specified by mapred.cluster.max.reduce.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off.	true	-1
mapred_compress_map_output	Controls whether the outputs of the maps be compressed before being sent across the network. Uses SequenceFile compression.	true	false
mapred_healthChecker_interval	Frequency of the node health script to be run, in milliseconds.	true	60000
mapred_healthChecker_script_args	List of arguments which are to be passed to node health script when it is being launched comma separated.	false	
mapred_healthChecker_script_path	Absolute path to the script which is periodically run by the node health monitoring service to determine if the node is healthy or not. If the value of this key is empty or the file does not exist in the location configured here, the node health monitoring service is not started.	false	
mapred_healthChecker_script_timeout	Time after node health script should be killed if unresponsive and considered that the script has failed.	true	600000
mapred_heartbeats_in_second	Expert: Approximate number of heart-beats that could arrive at JobTracker in a second. Assuming each RPC can be processed in 10msec, the default value is made 100 RPCs in a second.	true	100
mapred_hosts	Names a file that contains the list of nodes that may connect to the jobtracker. If the value is empty, all hosts are permitted.	false	
mapred_hosts_exclude	Names a file that contains the list of hosts that should be excluded by the jobtracker. If the value is empty, no hosts are excluded.	false	
mapred_inmem_merge_threshold	The threshold, in terms of the number of files, for the in-memory merge process. When we accumulate threshold number of files we initiate the in-memory merge and spill to disk. A value of 0 or less than 0 indicates we want to DON'T have any	true	1000

Name	Description	Required	Default
	threshold and instead depend only on the ramfs's memory consumption to trigger the merge.		
mapred_job_map_memory_mb	The size, in terms of virtual memory, of a single map task for the job. A job can ask for multiple slots for a single map task, rounded up to the next multiple of mapred.cluster.map.memory.mb and upto the limit specified by mapred.cluster.max.map.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off iff mapred.cluster.map.memory.mb is also turned off (-1).	true	-1
mapred_job_queue_name	Queue to which a job is submitted. This must match one of the queues defined in mapred.queue.names for the system. Also, the ACL setup for the queue must allow the current user to submit a job to the queue. Before specifying a queue, ensure that the system is configured with the queue, and access is allowed for submitting jobs to the queue.	true	default
mapred_job_reduce_input_buffer_percent	The percentage of memory- relative to the maximum heap size- to retain map outputs during the reduce. When the shuffle is concluded, any remaining map outputs in memory must consume less than this threshold before the reduce can begin.	true	0.0
mapred_job_reduce_memory_mb	The size, in terms of virtual memory, of a single reduce task for the job. A job can ask for multiple slots for a single map task, rounded up to the next multiple of mapred.cluster.reduce.memory.mb and upto the limit specified by mapred.cluster.max.reduce.memory.mb, if the scheduler supports the feature. The value of -1 indicates that this feature is turned off iff mapred.cluster.reduce.memory.mb is also turned off (-1).	true	-1
mapred_job_reuse_jvm_num_tasks	How many tasks to run per JVM. If set to -1, there is no limit.	true	1
mapred_job_shuffle_input_buffer_percent	The percentage of memory to be allocated from the maximum heap size to storing map outputs during the shuffle.	true	0.70
mapred_job_shuffle_merge_percent	The usage threshold at which an in-memory merge will be initiated, expressed as a percentage of the total memory allocated to storing in-memory map outputs, as defined by mapred.job.shuffle.input.buffer.percent.	true	0.66

Name	Description	Required	Default
mapred_job_tracker	The host and port that the MapReduce job tracker runs at. If "local", then jobs are run in-process as a single map and reduce task.	true	local
mapred_job_tracker_handler_count	The number of server threads for the JobTracker . This should be roughly 4% of the number of tasktracker nodes.	true	10
mapred_job_tracker_history_completed_location	The completed job history files are stored at this single well known location. If nothing is specified, the files are stored at ${hadoop.job.history.location}/done.	false	
mapred_job_tracker_http_address	The job tracker HTTP server address and port the server will listen on. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50030
mapred_job_tracker_jobhistory_lru_cache_size	The number of job history files loaded in memory. The jobs are loaded when they are first accessed. The cache is cleared based on LRU.	true	5
mapred_job_tracker_persist_jobstatus_active	Indicates if persistency of job status information is active or not.	true	false
mapred_job_tracker_persist_jobstatus_dir	The directory where the job status information is persisted in a file system to be available after it drops of the memory queue and between jobtracker restarts.	true	/jobtracker/jobsInfo
mapred_job_tracker_persist_jobstatus_hours	The number of hours job status information is persisted in DFS. The job status information will be available after it drops of the memory queue and between jobtracker restarts. With a zero value the job status information is not persisted at all in DFS.	false	
mapred_job_tracker_retiredjobs_cache_size	The number of retired job status to keep in the cache.	true	1000
mapred_jobtracker_blacklist_fault_bucket_width	The width (in minutes) of each bucket in the tasktracker fault timeout window. Each bucket is reused in a circular manner after a full timeout-window interval (defined by mapred.jobtracker.blacklist.fault-timeout-window).	true	15
mapred_jobtracker_blacklist_fault_timeout_window	The timeout (in minutes) after which per-job tasktracker faults are forgiven. The window is logically a circular buffer of time-interval buckets whose width is defined by mapred.jobtracker.blacklist.fault-bucket-width; when the "now" pointer moves across a bucket boundary, the previous contents (faults) of the new bucket are cleared. In other words, the timeout's granularity is determined by the bucket width.	true	180
mapred_jobtracker_complete	The maximum number of complete jobs	true	100

Name	Description	Required	Default
userjobs_maximum	per user to keep around before delegating them to the job history.		
mapred_jobtracker_job_history_block_size	The block size of the job history file. Since the job recovery uses job history, it is important to dump job history to disk as soon as possible. Note that this is an expert level parameter. The default value is set to 3 MB.	true	3145728
mapred_jobtracker_maxtasks_per_job	The maximum number of tasks for a single job. A value of -1 indicates that there is no maximum.	true	-1
mapred_jobtracker_restart_recover	"true" to enable (job) recovery upon restart, "false" to start afresh.	true	false
mapred_jobtracker_taskScheduler	The class responsible for scheduling the tasks.	true	org.apache.hadoop.mapred.JobQueueTaskScheduler
mapred_jobtracker_taskScheduler_maxRunningTasksPerJob	The maximum number of running tasks for a job before it gets preempted. No limits if undefined.	false	
mapred_line_input_format_linespermap	Number of lines per split in NLineInputFormat .	true	1
mapred_local_dir	The local directory where MapReduce stores intermediate data files. May be a comma-separated list of directories on different devices in order to spread disk i/o. Directories that do not exist are ignored.	true	${hadoop.tmp.dir}/mapred/local
mapred_local_dir_minspacekill	If the space in mapred.local.dir drops under this, do not ask more tasks until all the current ones have finished and cleaned up. Also, to save the rest of the tasks we have running, kill one of them, to clean up some space. Start with the reduce tasks, then go with the ones that have finished the least. Value in bytes.	false	
mapred_local_dir_minspacestart	If the space in mapred.local.dir drops under this, do not ask for more tasks. Value in bytes.	false	
mapred_map_max_attempts	Expert: The maximum number of attempts per map task. In other words, framework will try to execute a map task these many number of times before giving up on it.	true	4
mapred_map_output_compression_codec	If the map outputs are compressed, how should they be compressed?.	true	org.apache.hadoop.io.compress.DefaultCodec
mapred_map_tasks	The default number of map tasks per job. Ignored when mapred.job.tracker is "local".	true	2
mapred_map_tasks_speculative_execution	If true, then multiple instances of some map tasks may be executed in parallel.	true	true
mapred_max_tracker_blacklists	The number of blacklists for a tasktracker by various jobs after which the tasktracker	true	4

Name	Description	Required	Default
	will be marked as potentially faulty and is a candidate for graylisting across all jobs. (Unlike blacklisting, this is advisory; the tracker remains active. However, it is reported as graylisted in the web UI, with the expectation that chronically graylisted trackers will be manually decommissioned.) This value is tied to mapred.jobtracker.blacklist.fault-timeout-window; faults older than the window width are forgiven, so the tracker will recover from transient problems. It will also become healthy after a restart.		
mapred_max_tracker_failures	The number of task-failures on a tasktracker of a given job after which new tasks of that job aren't assigned to it.	true	4
mapred_merge_recordsBeforeProgress	The number of records to process during a merge before sending a progress notification to the TaskTracker .	true	10000
mapred_min_split_size	The minimum size chunk that map input should be split into. Note that some file formats may have minimum split sizes that take priority over this setting.	false	
mapred_output_compress	Controls whether the job outputs will be compressed.	true	false
mapred_output_compression_codec	If the job outputs are compressed, how should they be compressed?	true	org.apache.hadoop.io.compress.DefaultCodec
mapred_output_compression_type	If the job outputs are to compressed as SequenceFiles, how should they be compressed? Should be one of NONE, RECORD or BLOCK.	true	RECORD
mapred_queue_default_state	This values defines the state, default queue is in. the values can be either "STOPPED" or "RUNNING" This value can be changed at runtime.	true	RUNNING
mapred_queue_names	Comma separated list of queues configured for this jobtracker. Jobs are added to queues and schedulers can configure different scheduling properties for the various queues. To configure a property for a queue, the name of the queue must match the name specified in this value. Queue properties that are common to all schedulers are configured here with the naming convention, mapred.queue.$QUEUE-NAME.$PROPERTY-NAME, for e.g. mapred.queue.default.submit-job-acl. The number of queues configured in this parameter could depend on the type of scheduler being used, as specified in mapred.jobtracker.taskScheduler. For example, the JobQueueTaskScheduler	true	default

Name	Description	Required	Default
	supports only a single queue, which is the default configured here. Before adding more queues, ensure that the scheduler you've configured supports multiple queues.		
mapred_reduce_copy_backoff	The maximum amount of time (in seconds) a reducer spends on fetching one map output before declaring it as failed.	true	300
mapred_reduce_max_attempts	Expert: The maximum number of attempts per reduce task. In other words, framework will try to execute a reduce task these many number of times before giving up on it.	true	4
mapred_reduce_parallel_copies	The default number of parallel transfers run by reduce during the copy(shuffle) phase.	true	5
mapred_reduce_slowstart_completed_maps	Fraction of the number of maps in the job which should be complete before reduces are scheduled for the job.	true	0.05
mapred_reduce_tasks	The default number of reduce tasks per job. Typically set to 99% of the cluster's reduce capacity, so that if a node fails the reduces can still be executed in a single wave. Ignored when mapred.job.tracker is "local".	true	1
mapred_reduce_tasks_speculative_execution	If true, then multiple instances of some reduce tasks may be executed in parallel.	true	true
mapred_skip_attempts_to_start_skipping	The number of Task attempts AFTER which skip mode will be kicked off. When skip mode is kicked off, the tasks reports the range of records which it will process next, to the TaskTracker . So that on failures, TT knows which ones are possibly the bad records. On further executions, those are skipped.	true	2
mapred_skip_map_auto_incr_proc_count	The flag which if set to true, SkipBadRecords .COUNTER_MAP_PROCESSED_RECORDS is incremented by MapRunner after invoking the map function. This value must be set to false for applications which process the records asynchronously or buffer the input records. For example streaming. In such cases applications should increment this counter on their own.	true	true
mapred_skip_map_max_skip_records	The number of acceptable skip records surrounding the bad record PER bad record in mapper. The number includes the bad record as well. To turn the feature of detection/skipping of bad records off, set the value to 0. The framework tries to narrow down the skipped range by retrying	false	

Name	Description	Required	Default
	until this threshold is met OR all attempts get exhausted for this task. Set the value to Long.MAX_VALUE to indicate that framework need not try to narrow down. Whatever records (depends on application) get skipped are acceptable.		
mapred_skip_out_dir	If no value is specified here, the skipped records are written to the output directory at _logs/skip. User can stop writing skipped records by giving the value "none".	false	
mapred_skip_reduce_auto_incr_proc_count	The flag which if set to true, SkipBadRecords .COUNTER_REDUCE_PROCESSED_GROUPS is incremented by framework after invoking the reduce function. This value must be set to false for applications which process the records asynchronously or buffer the input records. For example streaming. In such cases applications should increment this counter on their own.	true	true
mapred_skip_reduce_max_skip_groups	The number of acceptable skip groups surrounding the bad group PER bad group in reducer. The number includes the bad group as well. To turn the feature of detection/skipping of bad groups off, set the value to 0. The framework tries to narrow down the skipped range by retrying until this threshold is met OR all attempts get exhausted for this task. Set the value to Long.MAX_VALUE to indicate that framework need not try to narrow down. Whatever groups(depends on application) get skipped are acceptable.	false	
mapred_submit_replication	The replication level for submitted job files. This should be around the square root of the number of nodes.	true	10
mapred_system_dir	The directory where MapReduce stores control files.	true	${hadoop.tmp.dir}/mapred/system
mapred_task_cache_levels	This is the max level of the task cache. For example, if the level is 2, the tasks cached are at the host level and at the rack level.	true	2
mapred_task_profile	To set whether the system should collect profiler information for some of the tasks in this job? The information is stored in the user log directory. The value is "true" if task profiling is enabled.	true	false
mapred_task_profile_maps	To set the ranges of map tasks to profile. mapred.task.profile has to be set to true for the value to be accounted.	true	0-2
mapred_task_profile_reduces	To set the ranges of reduce tasks to profile. mapred.task.profile has to be set to true for	true	0-2

Name	Description	Required	Default
	the value to be accounted.		
mapred_task_timeout	The number of milliseconds before a task will be terminated, if it neither reads an input; writes an output; nor updates its status string.	true	600000
mapred_task_tracker_http_address	The task tracker HTTP server address and port. If the port is 0 then the server will start on a free port.	true	0.0.0.0:50060
mapred_task_tracker_report_address	The interface and port that task tracker server listens on. Since it is only connected to by the tasks, it uses the local interface. EXPERT ONLY. Should only be changed if your host does not have the loopback interface.	true	127.0.0.1:0
mapred_task_tracker_task_controller	TaskController which is used to launch and manage task execution.	true	org.apache.hadoop.mapred.DefaultTaskController
mapred_tasktracker_dns_interface	The name of the Network Interface from which a task tracker should report its IP address.	true	default
mapred_tasktracker_dns_nameserver	The host name or IP address of the name server (DNS) which a TaskTracker should use to determine the host name used by the JobTracker for communication and display purposes.	true	default
mapred_tasktracker_expiry_interval	Expert: The time-interval, in milliseconds, after which a tasktracker is declared 'lost' if it doesn't send heartbeats.	true	600000
mapred_tasktracker_indexcache_mb	The maximum memory that a task tracker allows for the index cache that is used when serving map outputs to reducers.	true	10
mapred_tasktracker_map_tasks_maximum	The maximum number of map tasks that will be run simultaneously by a task tracker.	true	2
mapred_tasktracker_memory_calculator_plugin	Name of the class whose instance will be used to query memory information on the tasktracker. The class must be an instance of org.apache.hadoop.util.MemoryCalculatorPlugin. If the value is null, the tasktracker attempts to use a class appropriate to the platform. Currently, the only platform supported is Linux.	false	
mapred_tasktracker_reduce_tasks_maximum	The maximum number of reduce tasks that will be run simultaneously by a task tracker.	true	2
mapred_tasktracker_taskmemorymanager_monitoring_interval	The interval, in milliseconds, for which the tasktracker waits between two cycles of monitoring its tasks' memory usage. Used only if tasks' memory management is enabled via mapred.tasktracker.tasks.maxmemory.	true	5000

Name	Description	Required	Default
mapred_tasktracker_tasks_sleeptime_before_sigkill	The time, in milliseconds, the tasktracker waits for sending a SIGKILL to a process, after it has been sent a SIGTERM.	true	5000
mapred_temp_dir	A shared directory for temporary files.	true	${hadoop.tmp.dir}/mapred/temp
mapred_user_jobconf_limit	The maximum allowed size of the user jobconf. The default is set to 5 MB.	true	5242880
mapred_userlog_limit_kb	The maximum size of user-logs of each task in KB. 0 disables the cap.	false	
mapred_userlog_retain_hours	The maximum time, in hours, for which the user-logs are to be retained after the job completion.	true	24
mapreduce_job_acl_modify_job	Job specific access-control list for 'modifying' the job. It is only used if authorization is enabled in Map/Reduce by setting the configuration property mapred.acls.enabled to true. This specifies the list of users and/or groups who can do modification operations on the job. For specifying a list of users and groups the format to use is "user1,user2 group1,group". If set to '*', it allows all users/groups to modify this job. If set to ' '(i.e. space), it allows none. This configuration is used to guard all the modifications with respect to this job and takes care of all the following operations: o killing this job o killing a task of this job, failing a task of this job o setting the priority of this job Each of these operations are also protected by the per-queue level ACL "acl-administer-jobs" configured via mapred-queues.xml. So a caller should have the authorization to satisfy either the queue-level ACL or the job-level ACL. Irrespective of this ACL configuration, job-owner, the user who started the cluster, cluster administrators configured via mapreduce.cluster.administrators and queue administrators of the queue to which this job is submitted to configured via mapred.queue.queue-name.acl-administer-jobs in mapred-queue-acls.xml can do all the modification operations on a job. By default, nobody else besides job-owner, the user who started the cluster, cluster administrators and queue administrators can perform modification operations on a job.	false	
mapreduce_job_acl_view_job	Job specific access-control list for 'viewing' the job. It is only used if authorization is enabled in Map/Reduce by setting the configuration property mapred.acls.enabled to true. This specifies	false	

Name	Description	Required	Default
	the list of users and/or groups who can view private details about the job. For specifying a list of users and groups the format to use is "user1,user2 group1,group". If set to '*', it allows all users/groups to modify this job. If set to ' '(i.e. space), it allows none. This configuration is used to guard some of the job-views and at present only protects APIs that can return possibly sensitive information of the job-owner like o job-level counters o task-level counters o tasks' diagnostic information o task-logs displayed on the TaskTracker web-UI and o job.xml showed by the JobTracker 's web-UI Every other piece of information of jobs is still accessible by any other user, for e.g., JobStatus , JobProfile , list of jobs in the queue, etc. Irrespective of this ACL configuration, job-owner, the user who started the cluster, cluster administrators configured via mapreduce.cluster.administrators and queue administrators of the queue to which this job is submitted to configured via mapred.queue.queue-name.acl-administer-jobs in mapred-queue-acls.xml can do all the view operations on a job. By default, nobody else besides job-owner, the user who started the cluster, cluster administrators and queue administrators can perform view operations on a job.		
mapreduce_job_complete_cancel_delegation_tokens	If false - do not unregister/cancel delegation tokens from renewal, because same tokens may be used by spawned jobs.	true	true
mapreduce_job_counters_limit	Limit on the number of counters allowed per job.	true	120
mapreduce_job_split_metainfo_maxsize	The maximum permissible size of the split metainfo file. The JobTracker won't attempt to read split metainfo files bigger than the configured value. No limits if set to -1.	true	10000000
mapreduce_jobtracker_staging_root_dir	The root of the staging area for users' job files In practice, this should be the directory where users' home directories are located (usually /user).	true	${hadoop.tmp.dir}/mapred/staging
mapreduce_reduce_input_limit	The limit on the input size of the reduce. If the estimated input size of the reduce is greater than this value, job is failed. A value of -1 means that there is no limit set.	true	-1
mapreduce_reduce_shuffle_connect_timeout	Expert: The maximum amount of time (in milli seconds) a reduce task spends in trying to connect to a tasktracker for	true	180000

Name	Description	Required	Default
	getting map output.		
mapreduce_reduce_shuffle_read_timeout	Expert: The maximum amount of time (in milliseconds) a reduce task waits for map output data to be available for reading after obtaining connection.	true	180000
mapreduce_tasktracker_group	Expert: Group to which TaskTracker belongs. If LinuxTaskController is configured via mapreduce.tasktracker.taskcontroller, the group owner of the task-controller binary should be same as this group.	false	
mapreduce_tasktracker_outofband_heartbeat	Expert: Set this to true to let the tasktracker send an out-of-band heartbeat on task-completion for better latency.	true	false
tasktracker_http_threads	The number of worker threads that for the HTTP server. This is used for map output fetching.	true	40

Table 4-27: Hadoop Barclamp Fair Scheduler Parameters

Name	Description	Required	Default
default_min_share_preemption_timeout	Sets the default minimum share preemption timeout for any pools where it is not specified.	true	600
default_pool_scheduling_mode	Sets the default scheduling mode (fair or fifo) for pools whose mode is not specified.	true	fair
fair_share_preemption_timeout	Sets the preemption timeout used when jobs are below half their fair share.	true	600
pool_max_jobs_default	Sets the default running job limit for any pools whose limit is not specified.	true	20
user_max_jobs_default	Sets the default running job limit for any users whose limit is not specified.	true	10

Pig Barclamp

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

Pig's infrastructure layer consists of a compiler that produces sequences of MapReduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

- Ease of programming. It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.

- Optimization opportunities. The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- Extensibility. Users can create their own functions to do special-purpose processing.

Barclamp Parameters

Table 4-34: Pig Barclamp Parameters

Name	Description	Required	Default
java_home	JAVA_HOME environment variable.	true	/usr/java/jdk1.6.0_27/jre
log4jconf	log4jconf log4j configuration file.	true	./conf/log4j.properties
brief	brief logging - no timestamps.	true	false
cluster	Clustername, name of the hadoop jobtracker. If no port is defined port 50020 will be used.	false	
debug_level	Debug level, INFO is default.	true	INFO
file	A file that contains pig script.	false	
jar	Load jarfile, colon separated.	false	
verbose	Verbose print all log messages to screen (default to print only INFO and above to screen).	true	false
exectype	Exectype local or mapreduce - mapreduce is default.	true	mapreduce
ssh_gateway	HOD gateway property.	false	
hod_expect_root	HOD expect root property.	false	
hod_expect_uselatest	HOD use latest root property.	false	
hod_command	HOD command root property.	false	
hod_config_dir	HOD config directory property.	false	
hod_param	HOD param property.	false	
pig_spill_size_threshold	Do not spill temp files smaller than this size (bytes).	true	5000000
pig_spill_gc_activation_size	EXPERIMENT: Activate garbage collection when spilling a file bigger than this size (bytes). This should help reduce the number of files being spilled.	true	40000000
log_file	Log file location.	false	

Hive Barclamp

Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL . This language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Barclamp Parameters

Table 4-41: Hive Barclamp Parameters

Name	Description	Required	Default
hive_exec_scratchdir	Scratch space for Hive jobs.	true	/tmp/hive-${user.name}
hive_metastore_local	Controls whether to connect to remove metastore server or open a new metastore server in Hive Client JVM.	true	true
javax_jdo_option_ConnectionURL	JDBC connect string for a JDBC metastore.	true	jdbc:derby:;databaseName=metastore_db;create=true
javax_jdo_option_ConnectionDriverName	Driver class name for a JDBC metastore.	true	org.apache.derby.jdbc.EmbeddedDriver
hive_metastore_metadb_dir	The location of filestore metadata base dir.	true	file:///var/metastore/metadb/
hive_metastore_uris	Comma separated list of URIs of metastore servers. The first server that can be connected to will be used.	true	file:///var/metastore/metadb/
hive_metastore_warehouse_dir	The location of the default database for the warehouse.	true	/user/hive/warehouse
hive_metastore_connect_retries	Number of retries while opening a connection to metastore.	true	5
hive_metastore_rawstore_impl	Name of the class that implements org.apache.hadoop.hive.metastore.rawstore interface. This class is used to store and retrieval of raw metadata objects such as table, database.	true	org.apache.hadoop.hive.metastore.ObjectStore
hive_default_fileformat	Default file format for CREATE TABLE statement. Options are TextFile and SequenceFile.	true	TextFile
hive_map_aggr	Whether to use map-side aggregation in Hive Group By queries.	true	false
hive_join_emit_interval	How many rows in the right-most join operand Hive should buffer before emitting the join result.	true	1000
hive_exec_script_maxerrsize	Maximum number of bytes a script is allowed to emit to standard error (per map-reduce task). This prevents runaway scripts from filling logs partitions to capacity .	true	100000
hive_exec_compress_output	Controls whether the final outputs of a query (to a local/hdfs file or a hive table) is compressed. The compression codec and other options are determined from hadoop config variables mapred.output.compress.	true	false
hive_exec_compress_intermediate	Controls whether intermediate files produced by hive between multiple map-reduce jobs are compressed. The compression codec and other options are determined from hadoop config variables mapred.output.compress.	true	false

Sqoop Barclamp

SQL based command-line tool to assist with HDFS data import/export (SQL-to-Hadoop). Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce , and then export the data back into an RDBMS.

Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

Barclamp Parameters

Table 4-48: Sqoop Barclamp Parameters

Name	Description	Required	Default
sqoop_connection_factories	A comma-delimited list of ManagerFactory implementations which are consulted, in order, to instantiate ConnManager instances used to drive connections to databases.	false	
sqoop_tool_plugins	A comma-delimited list of ToolPlugin implementations which are consulted, in order, to register SqoopTool instances which allow third-party tools to be used.	false	
sqoop_metastore_client_enable_autoconnect	If true, Sqoop will connect to a local metastore for job management when no other metastore arguments are provided.	true	false
sqoop_metastore_client_autoconnect_url	The connect string to use when connecting to a job-management metastore. If unspecified, uses ~/.sqoop/. You can specify a different path here.	false	
sqoop_metastore_client_autoconnect_username	The username to bind to the metastore.	false	
sqoop_metastore_client_autoconnect_password	The password to bind to the metastore.	false	
sqoop_metastore_client_record_password	If true, allow saved passwords in the metastore.	false	
sqoop_metastore_server_location	Path to the shared metastore database files. If this is not set, it will be placed in ~/.sqoop/.	false	
sqoop_metastore_server_port	Port that this metastore should listen on.	false	

ZooKeeper Barclamp

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them, which makes them brittle in the presence of change, and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

ZooKeeper aims to distill the essence of these different services into a very simple interface to a centralized coordination service. The service itself is distributed and highly reliable. Consensus, group management, and presence protocols will be implemented by the service so that the applications do not need to implement them on their own. Application specific uses of these will consist of a mixture of specific components of ZooKeeper and application specific conventions. ZooKeeper recipes show how this simple service can be used to build much more powerful abstractions.

Barclamp Parameters

Table 4-55: ZooKeeper Barclamp Parameters

Name	Description	Required	Default
cluster_name	Provides the ability separate zookeeper services in logical groups.	true	default
tick_time	The number of milliseconds of each tick.	true	2000
init_limit	The number of ticks that the initial synchronization phase can take.	true	10
sync_limit	The number of ticks that can pass between sending a request and getting an acknowledgement.	true	5
client_port	Port at which the clients will connect.	true	2181
peer_port	Server peer port.	true	2888
leader_port	Server leader port.	true	3888
data_dir	Directory where the Zookeeper snapshot is stored.	true	/var/zookeeper
jvm_flags	Increase the heapsize of the ZooKeeper -Server instance to 4GB.	true	-Dzookeeper.log.threshold=INFO -Xmx4G
data_log_dir	Directory where the data log is stored.	false	/var/log/zookeeper

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 are clients of them. The barclamp attempts to direct all traffic over the admin network.

Barclamp Parameters

Table 4-56. Nagios Barclamp Parameters

Name	Default	Description
admin_interface_eval	Chef::Recipe::Barclamp::Inventory.get_network_by_type(node, \"admin\").interface	The ruby eval expression 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.

Ganglia Barclamp

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

Barclamp Parameters

Table 4-57: Ganglia Barclamp 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.

Test Barclamp

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

Barclamp Parameters

Table 4-58: Test Barclamp 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

Supplemental Material

System Verification

As a final step, it is important 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.

Deactivating Barclamps

As of October 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:

- In Crowbar, delete all proposals associated with the barclamp.
- In Chef, delete the barclamp in the Crowbar data bag.

Creating Barclamps

This section briefly describes barclamps, and how to create and import barclamps. It also briefly describes the layout of a barclamp, and the Crowbar.yml file.

Note: For the latest information about creating barclamps, please visit https://github.com/dellcloudedge/crowbar/wiki/Barclamp:-create-&-install-steps

Introduction

A barclamp is a deployment module that is imported from its own code repository into the Crowbar framework. A barclamp cannot operate without Crowbar, but you do not have to create a unique build of Crowbar in order to create a barclamp.

Note: You must install Crowbar before creating or importing barclamps.

When creating barclamps, most of the work is in building your Chef cookbooks. If you don’t have a cookbook that deploys your application, then stop here and create one.

In addition, there are many barclamps that you can study for examples, such as:

- **Nagios** if you have a service that needs to be integrated into every node.
- **Hadoop**, **Hive**, **Sqoop**, **Pig**, or **Zookeeper** if you have a complex multi-component system.
- **Provisioner** if you want to impact core Crowbar functionality.

Barclamp Creation Procedure

The following steps use the barclamp_model, included under `/dell/opt`.

1. Figure out the name of your barclamp. I’m naming our example “`foo barclamp`”
   a. Barclamps must have unique names.
   b. Do not use spaces or hyphens.
2. From the Crowbar server, become the super admin: `sudo -i`
3. Create a directory for your barclamp: `mkdir /barclamps`
4. Run the barclamp create script: `/opt/dell/bin/barclamp_create.rb foo “Zehicle” /barclamps`
   a. “`foo`” is our barclamp name [required]
   b. “`Zehicle`” is my company name for the copyright information [default is Dell]
   c. “`/barclamps`” is the path where we are putting the barclamp [default is `/opt/dell/barclamps`]
   d. Result will be a populated barclamp. In this example: `/barclamps/foo`.

If you want to plan ahead, you could use an initialized `git` repository as the target.

Reminder: Before building your barclamp, you’ll need to learn about Chef, how Crowbar extends cookbooks, and how barclamps interact. That is beyond the scope of this document.

Importing a Barclamp

Once you created a barclamp, you can import the barclamp into Crowbar & Chef. Assuming that you already created the foo barclamp in /barclamps (see Creating a Barclamp), proceed as follows:

1. From the Crowbar server, become the super admin: sudo –i
2. Run the barclamp install script: /opt/dell/bin/barclamp_install /barclamps/foo
    a. "/barclamps/foo" is the path to your barclamp. It could be anything.
    b. The core barclamps are in /opt/dell/barclamps.
    c. In a vm, you could mount a shared folder to access the barclamp (e.g., /mnt/hgfs/barclamps)

Your barclamp should now appear in the Crowbar UI. You can also see it in Chef under the Crowbar data bag.

While barclamps are generally safe to install multiple times, you can uninstall a barclamp using "barclamp_uninstall.rb /path/to/barclamp".

Barclamp Layout

A barclamp has the following core components:

- crowbar.yml configuration file (documented below).
- README.txt file (optional, recommended).
- Chef directory containing;
    - Cookbooks directory with Chef cookbooks.
    - Data_bags directory with Crowbar configuration files.
    - Roles directory with Chef roles used by the cookbooks and data_bags.
- crowbar_framework directory containing;
    - app directory with Crowbar model, controller, and view code.
    - other optional directories to add components needed by the UI, such as images.

The barclamp_model has a functional layout that covers most configuration requirements. The string ==BC-MODEL== indicates places where the name of the barclamp must be substituted.

Important: The name of the barclamp is embedded into the barclamp path and file names. This is required to avoid file collisions when the barclamp is imported.

Crowbar.yml

The crowbar.yml file is a required configuration file that gives direction to the installer. The file has the following components:

```
barclamp:
  name: name of your barclamp (required, do not use space or hyphens)
  display: pretty name of your barclamp [optional for now]
  description: information about your barclamp your barclamp [optional for now]
  version: what you want to consider for versioning [optiona]
crowbar:
```

layout: 1 (use the # one. This is required because it tells the installer what to expect inside your barclamp)

order: 1000 (if installing multiple barclamps in one pass, order tells the installer in which order to install them)

nav: (remove the nav section, advanced users only)

locale_additions: (you must add UI localizations here if you have any custom UI components)

en: (entries in this file map directly to entries in the config/locales/e

More Information

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 use one of the following contact methods.

Support

Cloudera Support

To obtain support for Hadoop:

- Open a reqeust at Cloudera's support portal. http://www.cloudera.com/hadoop-support/

Crowbar Support

To obtain support for Crowbar:

- Gathering Log Information
- Email hadoop@dell.com

To help facilitate troubleshooting of the environment a utility to gather logs has been provided. Browse to http://<Admin_ip>:3000/support/logs. This creates a tar archive of the relevant logs and asks the user for a location to save the resulting archive.

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

Printed in USA

www.dell.com| support.dell.com