Splunk® Enterprise Managing Indexers and Clusters of Indexers 7.3.1

Upgrade an indexer cluster

Generated: 9/05/2019 10:03 am
Upgrade an indexer cluster

The upgrade process differs considerably depending on the nature of the upgrade. This topic covers these version-based scenarios:

- Upgrading from 6.x or 7.x.
- Upgrading to a new maintenance release (for example, from 6.1.1 to 6.1.2)
- Upgrading from 5.x
- Upgrading from 5.0.1 or earlier

In addition, the topic describes:

- How to upgrade an indexer cluster that integrates with a search head cluster.
- How to perform a site-by-site upgrade of a multisite indexer cluster.

Migrating from single-site to multisite?

To convert a single-site indexer cluster to multisite, perform the upgrade first and then read Migrate an indexer cluster from single-site to multisite.

Upgrading an indexer cluster that integrates with a search head cluster?

If you are upgrading from 6.x or 7.x, you can upgrade each cluster separately, following the steps for tiered upgrades. See Upgrade each tier separately.

Otherwise, you must upgrade both clusters at the same time:

1. Follow the procedure in this topic for the type of indexer cluster upgrade that fits your deployment.
2. Stop all search head cluster members during the step in the indexer cluster upgrade that calls for stopping the search head.
3. Perform the remainder of the search head cluster upgrade steps during the step in the indexer cluster upgrade that calls for upgrading the search head, using the upgrade process described in Perform a non-rolling upgrade in Distributed Search.
Upgrading an indexer cluster that does not have a custom security key?

The security key, also known as the `pass4SymmKey` setting, authenticates communication between the master and the peers and search heads.

Starting in 6.6, a non-default security key is required. If the cluster’s security key was never explicitly set to a custom value, a warning message appears on the master node:

```
pass4SymmKey setting in the clustering or general stanza of server.conf is set to empty or the default value. You must change it to a different value.
```

To remediate this situation, you must set the security key on all the cluster nodes (master, peer nodes, search heads) while the cluster is down. The key must be the same across all cluster nodes.

You set the security key with the `pass4SymmKey` attribute in `server.conf`. See Configure the security key.

To set the key during cluster upgrade, you must upgrade all cluster tiers at once, following the procedure in Upgrade all tiers at once. Set the security key while all nodes are down, so that they all have the same security key when they start up.

**Upgrade a 6.x or 7.x indexer cluster**

When upgrading a 6.x or 7.0 single-site indexer cluster to a later version, you must take down and upgrade all peer nodes as a single operation. You cannot perform a rolling, online upgrade of the peer nodes. If you have a multisite cluster, however, you can upgrade one site at a time. See Site-by-site upgrade for multisite indexer clusters.

You can upgrade all tiers of the cluster (master node, search heads, peer nodes) at once or you can upgrade each tier separately.

The approach of upgrading each tier separately is particularly useful if the search head tier consists of a search head cluster, because it eliminates the need to upgrade the search head cluster simultaneously with the indexer cluster peer nodes.
**Caution:** Even when upgrading each tier separately, it is strongly recommended that you complete the entire upgrade process quickly, to avoid any possibility of incompatibilities between node types running different versions.

To upgrade all cluster tiers at once, see Upgrade all tiers at once.

To upgrade each tier separately, see Upgrade each tier separately.

**Upgrade all tiers at once**

Perform the following steps:

1. Stop the master.
2. Stop all the peers and search heads.
   When bringing down the peers, use the `splunk stop` command, not `splunk offline`.
3. If the cluster does not use a non-default (custom) security key, you must set one now. Starting in 6.6, indexer clusters require a non-default security key. This key must be the same across all nodes in the cluster. See Upgrading an indexer cluster that does not have a custom security key?. On each node (master, peers, and search heads), set the key using the procedure in Configure the security key.
4. Upgrade the master node, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the Installation Manual. **Do not upgrade the peers yet.**
5. Start the master, accepting all prompts, if it is not already running.
6. Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.
7. Upgrade the search heads, followed by the peer nodes. Use the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the Installation Manual.
8. Start the peer nodes and search heads, if they are not already running.
9. Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode`.

You can view the master dashboard to verify that all cluster nodes are up and running.

**Upgrade each tier separately**

When upgrading tiers separately:
• You must upgrade the tiers in the prescribed order.
• Within each tier, you must upgrade all nodes as a single operation.

Functionality introduced in the new release will not be available until all tiers complete the upgrade.

Caution: Even when upgrading each tier separately, it is strongly recommended that you complete the entire upgrade process quickly, to avoid any possibility of incompatibilities between node types running different versions.

You must follow this order of upgrade when upgrading the tiers in discrete operations:

1. Upgrade the master node.
2. Upgrade the search head tier.
3. Upgrade the peer node tier.

1. Upgrade the master node

1. Stop the master.
2. Upgrade the master, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the Installation Manual.
3. Start the master, accepting all prompts, if it is not already running.

You can view the master dashboard to verify that all cluster nodes are up and running.

2. Upgrade the search head tier

The method that you use to upgrade the search head tier depends on whether or not the tier consists of a search head cluster:

• If the search head tier consists of a search head cluster, follow the procedure in Upgrade a search head cluster. If desired, you can perform a rolling upgrade of the search head cluster, as described in that topic.

• If the search head tier consists of independent search heads, follow this procedure:

1. Stop all the search heads.
2. Upgrade the search heads, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in
You can view the master dashboard to verify that all cluster nodes are up and running.

3. **Upgrade the peer node tier**

1. Run `splunk enable maintenance-mode` on the master.
   To confirm that the master is in maintenance mode, run `splunk show maintenance-mode` on the master.
   This step prevents unnecessary bucket fix-ups. See Use maintenance mode.
2. Stop all the peer nodes.
   When bringing down the peers, use the `splunk stop` command, not `splunk offline`.
3. Upgrade the peer nodes, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.
4. Start the peer nodes, if they are not already running.
5. Run `splunk disable maintenance-mode` on the master.
   To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode` on the master.

You can view the master dashboard to verify that all cluster nodes are up and running.

**Upgrade from 7.1 or later using rolling upgrade**

When upgrading from 7.1 or later, you can perform a searchable rolling upgrade of the indexer cluster. See Use rolling upgrade.

**Site-by-site upgrade for multisite indexer clusters**

If you have a multisite cluster, you can upgrade one site at a time, as long as you are upgrading across no more than one sequential n.n version (for example, from 6.5 to 6.6, or 6.6 to 7.0, but not 6.5 to 7.0). Because each site has a full set of primary copies, this method allows searches to continue uninterrupted during the upgrade.

**Caution:** You cannot perform a site-by-site upgrade if you are upgrading across more than one sequential n.n version (for example, from 6.4 to 6.6 or 6.5 to 7.0). To upgrade across multiple sequential n.n versions, you must take down all peer
nodes across all sites during the upgrade process. You can do so by following either of the procedures outlined in Upgrade a 6.x or 7.x indexer cluster.

Alternatively, to upgrade across multiple sequential n.n versions, you can upgrade via interim releases of not more than a single n.n version. For example, if you are upgrading from 6.4 to 6.6, you can first upgrade to a 6.5 interim release using the site-by-site method. You can then upgrade to 6.6.

Functionality introduced in the new release will not be available until all nodes complete the upgrade.

For a two-site cluster, the upgrade procedure has three distinct phases:

1. Upgrade of the master node.
2. Upgrade of the site1 peers and search heads.
3. Upgrade of the site2 peers and search heads.

Here are the steps in detail:

1. Stop the master.
2. Upgrade the master node, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the Installation Manual.
3. Start the master, accepting all prompts, if it is not already running.
4. Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.
5. Stop all the peers and search heads on site1 with the `splunk stop` command.
6. Upgrade the site1 peer nodes and search heads.
7. Start the site1 peer nodes and search heads, if they are not already running.
8. Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode`. 
9. Wait until the master dashboard shows that both the search factor and replication factor are met.

10. Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`.

11. Stop all the peers and search heads on site2 with the `splunk stop` command.

12. Upgrade the site2 peer nodes and search heads.

13. Start the site2 peer nodes and search heads, if they are not already running.

14. Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode`.

You can view the master dashboard to verify that all cluster nodes are up and running.

### Upgrade to a maintenance release

To upgrade a cluster to a maintenance release (for example, from 6.1.0 to 6.1.1), you do not need to bring down the entire cluster at once. Instead, you can perform a rolling, online upgrade, in which you upgrade the nodes one at a time.

**Caution:** Even with a rolling upgrade, you should upgrade all nodes quickly, for several reasons:

- Proper functioning of the cluster depends on all peer nodes running the same version of Splunk Enterprise, as stated in System requirements and other deployment considerations for indexer clusters.
- Other version compatibility requirements must also be met, as described in System requirements and other deployment considerations for indexer clusters.
- If you upgrade the master but not the peers, the cluster might generate errors and warnings. This is generally okay for a short duration, but you should complete the upgrade of all nodes as quickly as possible.

To upgrade a cluster node, follow the normal procedure for any Splunk Enterprise upgrade, with the few exceptions described below. For general information on upgrading Splunk Enterprise instances, see How to upgrade Splunk Enterprise.

To perform a rolling maintenance upgrade, follow these steps:
1. **Upgrade the master node**

Upgrade the master node first.

For information on what happens when the master goes down, as well as what happens when it comes back up, see What happens when the master node goes down.

2. **Upgrade the search heads**

The only impact to the cluster when you upgrade the search heads is disruption to searches during that time.

3. **Put the master into maintenance mode**

Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.

4. **Upgrade the peer nodes**

When upgrading peer nodes, note the following:

   - Peer upgrades can disrupt ongoing searches.

   - To minimize downtime and to limit any disruption to indexing and searching, upgrade the peer nodes one at a time.

   - To bring a peer down prior to upgrade, use the `splunk offline` command, as described in Take a peer offline.

   - During the interim between when you upgrade the master and when you finish upgrading the peers, the cluster might generate various warnings and errors.

   - For multisite clusters, the site order of peer upgrades does not matter.

5. **Take the master out of maintenance mode**

Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode`. 
Upgrade from 5.x to 6.x or later

When you upgrade from a 5.x indexer cluster to a 6.x or later cluster, you must take all cluster nodes offline. You cannot perform a rolling, online upgrade.

Perform the following steps:

1. On the master, run the safe_restart_cluster_master script with the --get_list option:

   splunk cmd python safe_restart_cluster_master.py <master_uri> --auth <username>:<password> --get_list
   
   Note: For the master-uri parameter, use the URI and port number of the master node. For example: https://10.152.31.202:8089

   This command puts a list of all cluster bucket copies and their states into the file $SPLUNK_HOME/var/run/splunk/cluster/buckets.xml. This list is fed back to the master after the master upgrade.

   To obtain a copy of this script, copy it from here: The safe_restart_cluster_master script.

   For information on why this step is needed, see Why the safe_restart_cluster_master script is necessary.

2. Stop the master.

3. Stop all the peers and search heads.

   When you bring down the peers, use the splunk stop command, not splunk offline.

4. Upgrade the master node, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the Installation Manual. Do not upgrade the peers yet.

5. Start the master, accepting all prompts, if it is not already running.

6. Run the splunk apply cluster-bundle command, using the syntax described in Update cluster peer configurations and apps. (This step is necessary to avoid extra peer restarts, due to a 6.0 change in how the configuration bundle checksum is calculated.)
7. Run `splunk enable maintenance-mode` on the master. To confirm that the master is in maintenance mode, run `splunk show maintenance-mode`. This step prevents unnecessary bucket fix-ups. See Use maintenance mode.

8. Upgrade the peer nodes and search heads, following the normal procedure for any Splunk Enterprise upgrade, as described in How to upgrade Splunk Enterprise in the *Installation Manual*.

9. Start the peer nodes and search heads, if they are not already running.

10. On the master, run the `safe_restart_cluster_master` script again, this time with the `freeze_from` option, specifying the location of the bucket list created in step 1:

    splunk cmd python safe_restart_cluster_master.py <master_uri> --auth <username>:<password> --freeze_from <path_to_buckets_xml>

    For example:

    splunk cmd python safe_restart_cluster_master.py <master_uri> --auth admin:your_password --freeze_from
    $SPLUNK_HOME/var/run/splunk/cluster/buckets.xml

    This step feeds the master the list of frozen buckets obtained in step 1.

11. Run `splunk disable maintenance-mode` on the master. To confirm that the master is not in maintenance mode, run `splunk show maintenance-mode`.

    You can view the master dashboard to verify that all cluster nodes are up and running.

    **Why the safe_restart_cluster_master script is necessary**

    The `safe_restart_cluster_master` script solves a problem in the way that the 5.x master node handles frozen bucket copies. This problem is fixed starting with release 6.0. However, it is still an issue during master upgrades from 5.x. This section provides detail on the issue.

    When a peer freezes a copy of a bucket, the master stops doing fix-ups on that bucket. It operates under the assumption that the other peers will eventually freeze their copies of that bucket as well.

    This works well as long as the master continues to run. However, because (in 5.x) the knowledge of frozen buckets is not persisted on either the master or the
peers, if you subsequently restart the master, the master treats frozen copies (in the case where unfrozen copies of that bucket still exist on other peers) as missing copies and performs its usual fix-up activities to return the cluster to a complete state. If the cluster has a lot of partially frozen buckets, this process can be lengthy. Until the process is complete, the master is not able to commit the next generation.

To prevent this situation from occurring when you restart the master after upgrading to 6.0, you must run the safe_restart_cluster_master script on the master. As described in the upgrade procedure, when you initially run this script on the 5.x master with the --get_list option, it creates a list of all cluster bucket copies and their states, including whether they are frozen. When you then rerun it after upgrading the master to 6.x, using the freeze_from option, it feeds the list to the upgraded master so that it does not attempt fix-up of the frozen buckets.

**The safe_restart_cluster_master script**

To perform steps 1 and 9 of the upgrade procedure, you must run the safe_restart_cluster_master script. This script does not ship with the product. To obtain the script, copy the listing directly below and save it as $SPLUNK_HOME/bin/safe_restart_cluster_master.py.

Important: You must also copy and save the parse_xml_v3 script, as described in the next section, The parse_xml_v3 script.

Here are the contents of the script:

```python
import httplib2
from urllib import urlencode
import splunk, splunk.rest, splunk.rest.format
from parse_xml_v3 import *
import json
import re
import time
import os
import subprocess

# before restarting the master, store the buckets list in /var/run/splunk/cluster
BUCKET_LIST_PATH = os.path.join(os.environ['SPLUNK_HOME'], 'var', 'run', 'splunk', 'cluster', 'buckets.xml')

def get_buckets_list(master_uri, auth):
    f = open(BUCKET_LIST_PATH, 'w')
    atom_buckets = get_xml_feed(master_uri
```
def change_quiet_period(master_uri, auth):
    args={'quiet_period':'600'}
    return

def num_peers_up(master_uri, auth):
    count = 0
    f= open('peers.xml','w')
    atom_peers =
    get_xml_feed(master_uri+'/services/cluster/master/peers?count=-1',auth,'GET')
    f.write(atom_peers)
    f.close()
    file = open('peers.xml','r')
    for line in file:
        match = regex.findall(line)
        for line in match:
            count = count + 1
    file.close()
    os.remove('peers.xml')
    return count

def wait_for_peers(master_uri,auth,original_number):
    while(num_peers_up(master_uri,auth) != original_number):
        num_peers_not_up = original_number -
        num_peers_up(master_uri,auth)
        print "Still waiting for " +str(num_peers_not_up) +" peers to join ..."
        time.sleep(5)
        print "All peers have joined"

def get_response_feed(url, auth, method='GET', body=None):
    (user, password) = auth.split(':')
    h = httplib2.Http(disable_ssl_certificate_validation=True)
    h.add_credentials(user, password)
    if body is None:
        body = {}
    response, content = h.request(url, method, urlencode(body))
    if response.status == 401:
        raise Exception("Authorization Failed", url, response)
    elif response.status != 200:
        raise Exception(url, response)
    return splunk.rest.format.parseFeedDocument(content)

def get_xml_feed(url, auth, method='GET', body=None):
    (user, password) = auth.split(':')
h = httplib2.Http(disable_ssl_certificate_validation=True)
h.add_credentials(user, password)
if body is None:
    body = {}
response, content = h.request(url, method, urlencode(body))
if response.status == 401:
    raise Exception("Authorization Failed", url, response)
el if response.status != 200:
    raise Exception(url, response)
return content

def validate_rest(master_uri, auth):
    return get_response_feed(master_uri + '/services/cluster/master/info', auth)

def freeze_bucket(master_uri, auth, bid):
    return get_response_feed(master_uri + '/services/cluster/master/buckets/' + bid + '/freeze', auth, 'POST')

def freeze_from_file(master_uri, auth, path=BUCKET_LIST_PATH):
    file = open(path) #read the buckets.xml from either path supplied or BUCKET_LIST_PATH
    handler = BucketHandler()
    parse(file, handler)
    buckets = handler.getBuckets()
    fcount = 0
    fdone = 0
    for bid, bucket in buckets.iteritems():
        if bucket.frozen:
            fcount += 1
            try:
                freeze_bucket(master_uri, auth, bid)
                fdone += 1
            except Exception as e:
                print e
    print "Total bucket count:: ", len(buckets), "; number frozen: ", fcount, "; number re-frozen: ", fdone

def restart_master(master_uri, auth):
    change_quiet_period(master_uri, auth)
    original_num_peers = num_peers_up(master_uri, auth)
    print "\n" + "Issuing restart at the master" +"\n"
    subprocess.call([os.path.join(os.environ["SPLUNK_HOME"],"bin","splunk"),"restart"])
    print "\n"+ "Master was restarted" + "\n"
print "Waiting for all " +str(original_num_peers) + " peers to come back up"

wait_for_peers(master_uri,auth,original_num_peers)

print "Making sure we have the correct number of frozen buckets"

if __name__ == '__main__':
    usage = "usage: %prog [options] <master_uri> --auth admin:changeme"
    parser = OptionParser(usage)
    parser.add_option("-a", "--auth", dest="auth",
                      metavar="user:password", default=':',
                      help="Splunk authentication parameters for the master instance");
    parser.add_option("-g", "--get_list", action="store_true",help="get a list of frozen buckets and store them in buckets.xml");
    parser.add_option("-f", "--freeze_from",dest="freeze_from",
                      help="path to the file that contains the list of buckets to be frozen. ie path to the buckets.xml generated by the get_list option above");

    (options, args) = parser.parse_args()

    if len(args) == 0:
        parser.error("master_uri is required")
    elif len(args) > 1:
        parser.error("incorrect number of arguments")

    master_uri = args[0]
    try:
        validate_rest(master_uri, options.auth)
    except Exception as e:
        print "Failed to access the master info endpoint make sure you've supplied the authentication credentials"
        raise

    # Let's get a list of frozen buckets, stored in
    if(options.get_list):
        print "Only getting the list of buckets and storing it at " + BUCKET_LIST_PATH
        get_buckets_list(master_uri,options.auth)
    elif(options.freeze_from):
        print "Reading the list of buckets from " + options.freeze_from + "and refreezing them"
        freeze_from_file(master_uri,options.auth,options.freeze_from)
    else:
        print "Restarting the master safely to preserve knowledge of frozen buckets"
        get_buckets_list(master_uri,options.auth)
        restart_master(master_uri,options.auth)
        freeze_from_file(master_uri,options.auth,BUCKET_LIST_PATH)
The parse_xml_v3 script

The parse_xml_v3 script contains certain helper functions needed by the safe_restart_cluster_master script. This script does not ship with the product. To obtain the script, copy the listing directly below and save it as $SPLUNK_HOME/bin/parse_xml_v3.py.

Here are the contents of the script:

```python
import sys
from xml.sax import ContentHandler, parse
from optparse import OptionParser

class PeerBucketFlags:
    def __init__(self):
        self.primary = False
        self.searchable = False

class Bucket:
    def __init__(self):
        self.peer_flags = {}  # key is peer guid
        self.frozen = False

class BucketHandler(ContentHandler):
    def __init__(self):
        ContentHandler.__init__(self)
        self.buckets = ()  # key is peer guid
        self.frozen = False

    def getBuckets(self):
        return self.buckets

    def startDocument(self):
        pass

    def endDocument(self):
        pass

    def startElement(self, name, attrs):
        if name == 'entry':
            self.in_entry = True
            self.in_peers = False
            self.save_title = False
            self.save_frozen = False
            self.peer_nesting = 0
            self.current_peer_flags = {}
            self.current_guid = None
            self.current_frozen_flag = ''
            self.current_peer_field = None
            self.current_peer_field_value = ''
            self.current_bucket = ''
```

15
if name == 'entry':
    self.in_entry = True
elif self.in_entry and name == 'title':
    self.save_title = True
elif self.in_entry and name == 's:key' and attrs.get('name') == 'frozen':
    self.save_frozen = True
elif name == 's:key' and attrs.get('name') == 'peers':
    self.in_peers = True
elif self.in_peers and name == 's:key':
    self.peer_nesting += 1
    if self.peer_nesting == 1:
        self.current_peer_flags = PeerBucketFlags()
        self.current_guid = attrs.get('name').encode('ascii')
    elif self.peer_nesting == 2:
        self.current_peer_field =
        attrs.get('name').encode('ascii')
        self.current_peer_field_value = ''
def endElement(self, name):
    if name == 'entry':
        self.in_entry = False
        self.current_bucket =''
    elif self.save_title:
        try:
            (idx, local_id, origin_guid) =
            self.current_bucket.split('~')
        except ValueError as e:
            print "Invalid? ", self._locator.getLineNumber()
            print self.current_bucket
            print e
            raise
        self.buckets[self.current_bucket] = Bucket()
        self.save_title = False
    elif self.save_frozen:
        if self.current_frozen_flag in [1, '1', 'True', 'true']:
            self.buckets[self.current_bucket].frozen = True
        self.current_frozen_flag = ''
        self.save_frozen = False
    elif self.peer_nesting == 2 and name == 's:key':
        if self.current_peer_field == 'bucket_flags':
            self.current_peer_flags.primary =
            (self.current_peer_field_value == '0xffffffffffffffff')
        elif self.current_peer_field == 'search_state':
            self.current_peer_flags.searchable =
            self.current_peer_field_value == 'Searchable'
        # Nesting level goes down in either case.
        self.peer_nesting -= 1
    elif self.peer_nesting == 1 and name == 's:key':
        self.buckets[self.current_bucket].peer_flags[self.current_guid] =
        self.current_peer_flags
        self.peer_nesting -= 1
elif self.in_peers and self.peer_nesting == 0 and name == 's:key':
    self.in_peers = False

def characters(self, content):
    if self.save_title:
        self.current_bucket += content.encode('ascii').strip()
    elif self.save_frozen:
        self.current_frozen_flag += content.encode('ascii').strip()
    if self.peer_nesting > 0:
        s = content.encode('ascii').strip()
        if s:
            self.current_peer_field_value += s

Upgrade from 5.0.1 or earlier

During an upgrade from 5.0.1 or earlier, the /cluster directory under
$SPLUNK_HOME/etc/master-apps (on the master) and
$SPLUNK_HOME/etc/slave-apps (on the peers) is renamed to /_cluster. This
happens automatically. For details on this directory, see Update common peer
configurations.

When the master restarts after an upgrade from 5.0.1 or earlier, it performs a
rolling restart on the set of peer nodes, to push the latest version of the
configuration bundle (with the renamed /_cluster directory).