SOLR Knowledge Base Document:
Upgrade by Replacing &
Disaster Recovery

- Overview

This document is intended to cover how to Upgrade SOLR to a newer version any time we want to. The intended audience is IT and Dev. This document covers "Upgrade by Replacing" and Disaster recovery for SOLR only. It does not cover Zookeeper, Kafka, or any other moving parts we might add later.

The idea is that instead of worrying about how to upgrade a SOLR server, we can simply:

- Create (or clone) new VM's with appropriate networking
- Install the latest version of SOLR on the new VM's
- Modify any SOLR configs that have changed between versions(1)
- Put those SOLR instances into Zookeeper (under a chroot for the version) (2)
- Load the data from Kafka or pdssupport into SOLR
- Test that all is well
- Switch load balancer(s) to the new SOLR instances
- Done

This is intended to also work for fast, successful disaster recovery.

For disaster recovery, a stored "base image" of whatever the latest production version of a SOLR server looks like could probably be spun up in minutes and the data could be loaded from Kafka in another few minutes. I have not tested that, although it would basically start with "Fix the Settings in the SOLR Config Files" below and use the rest of the instructions verbatim, which is why I believe this works for DR.

This is a SOLR wiki page about disaster recovery. The environments they discuss are more complex than ours, but it may be worth a quick glance.

https://cwiki.apache.org/confluence/display/solr/NRT%2C+Replication%2C+and+Disaster+Recovery+with+SolrCloud

Please read this document with both of the objectives in mind – and point out any missing pieces to JohnB.(3)

Footnotes:

(1)I'll just state the obvious. These instructions were derived while upgrading from SOLR 4.9 to SOLR 5.4. Testing will be necessary for ANY other versions of SOLR.

(2) Using the zookeeper_IP_address/chroot approach will allow us to have several different SOLR "collections" if we wish. STATdx, ExpertPath, etc… We'll have to discuss the final approach here. They can live on the same SOLR machines as different "cores" or they could live on totally separate VMs. We will want Kevin's mind on this one for sure.

(3)This doc still has a few TODO tasks in it. It's a work in progress, although the steps to create new SOLR instances are tested and do work. I'll get the TODO's handled (along with any feedback I get from the team) and re-issue the doc as appropriate.

Note: an alternative way of getting things done on the command line using the "solr" user: *sudo su - solr -c* "/opt/solr/bin/solr create -c gettingstarted -n data_driven_schema_configs"

Upgrade by Replacing & DR Steps below:

- Make or Clone your VMs

See "Instructions for cloning the VM (Ubuntu)" document for steps to clone the Ubuntu Servers I'm currently using.

(Consider making just one with software installed and then cloning as many as you need, thus avoiding the time cost of installing repeatedly. Not a big deal either way.)

Alternatively, just install your Linux OS of choice in a VM and go from there. You mileage may vary slightly on networking etc, depending on the choice of OS. Everything here was done on Ubuntu Server 14.x

- Get the latest SOLR bits

Download and/or scp the necessary SOLR tar file to the VM

http://lucene.apache.org/solr/downloads.html

- Untar and Install SOLR per SOLR Wiki

Website for the instructions used for this doc. Later versions could be different.

https://cwiki.apache.org/confluence/display/solr/Taking+Solr+to+Production

The "Install for Production" instructions listed at the link above get you
/opt/solr/server/solr [this is where your "cores" will go]
And /var/solr as well. Logs are in /var/solr/logs

The install process also gets you a "solr" user -- I was unable to find a password so I had to sudo and make one I would know.

```
sudo su
passwd solr
  (Enter your password.)
```

I changed it to the same as the login for the "john" user)

I don't know if it's worth it to put the "solr" user into sudoers… It's sometimes a pain to navigate to solr-user-owned files and directory structures, but if you can get root, it's not a big deal.

==========

Note: I am NOT happy with how SOLR boots into "Stand-alone Mode" when the "set up for production" instructions from the wiki are followed. There is a ZK_HOST setting in the solr.in.sh file, and all the docs -- including a "fixed" bug -- suggest that this entry will cause SOLR to start up in "Cloud Mode" properly.

I have not been able to cause that to happen. Even when ZK_HOST has what appear to me to be proper values for the Zookeeper VMs, SOLR comes up in "Stand-alone

Mode" after booting the OS or when you issue sudo service solr start or restart. This is obvious in the SOLR Admin console (Cloud → Graph) where the nodes are either "down" or "not found" or, if you're pinging the exact VM that came up in "Stand-alone Mode" - the "Cloud" button/link will not even be available, and the core (statdx) is not found.

Instead, at this point, you have to use the restart command on the command line to get things working correctly. This could be automated to happen at boot of course, but I haven't done that. We'll either need to polish that particular stone a little more, or know that SOLR has to be restarted from the command line every time the OS is re-booted.

I spent about 2 hours trying various combinations of config settings and restarting either the OS or SOLR (via sudo service solr restart) and nothing worked, so I've posted something on stack overflow and will update this doc if I get more data.

See the section titled "Start or Restart SOLR" below for the exact command line. Solr.in.sh lives in /opt/solr/bin AND in /etc/default. I'm still not sure which one of these is actually used. TODO: Remove one and reboot SOLR, then remove the other. Removing one of them ought to break something and thus indicate which is actually being used – unless there's a cascading usage of both.
==========

- Fix the Settings in the SOLR Config Files

/opt/solr/bin/solr.in.sh
(Also here /etc/default/solr.in.sh TODO: which is authoritative?)
ZK_HOST="192.168.56.5:2181,192.168.56.6:2181,192.168.56.7:2181/solr5_4"
SOLR_HOST="192.168.56.17" (The IP address of the VM you're working on)

/opt/solr/server/solr/solr.xml
<str name="host">${solr7:}</str>
Again, the VM you're working on. Hostname or IP.
(IP address should be good here too.)
Obviously, the host name had better resolve. On my Ubuntu servers I had this mapped to the IP address in /etc/hosts and I assume it wouldn't have worked otherwise.

- Consider Cloning at this point

This is where you might clone N number of new SOLR VM's.
- This should ensure that the setup etc. is all the same
- There is the matter of networking and the SOLR config files in the previous section
They would have to be modified after cloning.
- These can be brought into the SOLR/Zookeeper "Cloud" easily enough at the end of this process. It will be slightly harder if they're not ready to go when we issue the final "create collection" command.

- Ensure that Zookeeper ensemble is up and running

Ensure that the SOLR machines can "talk" to the Zookeeper machines.
Ensure that each Zookeeper can ping the EXACT IP or hostname that is in the solr.xml file for the SOLR host setting. If you “install for production” the solr.xml file will be found here: /opt/solr/server/solr/solr.xml

sudo nano solr.xml will allow you to make changes.

Watch out for this in solr.xml:
<str name="host">${solr5:}</str>

The above is correct; anything else is problematic.
I suspect this (below) will work, but note the colon. Not tested.

<str name="host">${192.168.56.16:}</str>

This value gets passed up to Zookeeper and if the zookeeper cannot "hit" this successfully then the new core(s) will NOT get created.

TODO: Test this: <str name="host">${192.168.56.16:}</str>

==========
The initial value is <str name="host">${host:}</str> I think I had some trouble with that, but after all the different approaches I tried, I’m not sure what, or if it really mattered. This probably interacts with /etc/hosts or /etc/hostname or some other networking function(s) on the SOLR host.
TODO: VERIFY IF THIS MATTERS. RUN ANOTHER TEST CREATE
==========

Last, but NOT least, ensure that each SOLR machine can resolve the other in networking terms either by hostname, IP or both - depending on how you've decided to set up the network.

On my Ubuntu machines, all of this "ensuring connectivity" came down to proper configuration of the VMs themselves and making sure that all the other machines in your "set" are also in /etc/hosts. I'm aware that's simplistic and obviously DNS or something is probably what is wanted in Prod.

I will leave that networking bit to the experts. It remains true that each machine has to be able to "get to" the others. The SOLR server reports the setting in the solr.xml file to Zookeeper, so whatever you have in there is critical from the perspective of being able to connect in both directions.

- Add Configuration Files to Zookeeper

Add the configs (ONLY) to that new chroot in Zookeeper with the SOLR version of the zkcli.sh thus:

sudo /opt/solr/server/scripts/cloud-scripts/zkcli.sh -cmd upconfig
-confdir /home/john/conf/
-confname statdx
-z 192.168.56.5/solr5_4

(The -confdir should contain all the configs from your existing collection)
Eventually these configs MUST be stored in version control!

NOTE: You can create a just a "node" in Zookeeper like this:

```
./server/scripts/cloud-scripts/zkcli.sh -zkhost 127.0.0.1:2181 \
-cmd makepath /solr
```

This would be useful for setting up what is needed before bringing a collection online – but since the "add the configs" command I mention in this section does both, I've left that alone. The link below provides some good examples and serves as a "better than a man page" for the command.

https://cwiki.apache.org/confluence/display/solr/Command+Line+Utilities

- Verify Zookeeper has expected Nodes

Go to Zookeeper. Verify using the zkCli.sh tool that you now have a /solr5_4 "directory" and a config "directory" below it – with a "directory" inside the config "directory" bearing the name you passed in on the -confname flag

ls /
ls /solr5_4
ls /solr5_4/config (should see a statdx "directory" in this case)

There should also NOT be a collections directory under /solr5_4 as a result of this command. If there is, verify that there is NOT a collection within there of the same name passed with the -confname flag on the previous step.

- Start or Restart SOLR

Start (or restart) the SOLR servers in cloud mode with the correct zookeeper addresses AND a NEW CHROOT for this version of SOLR thus:

sudo /opt/solr/bin/solr restart -c -z
192.168.56.5,192.168.56.6,192.168.56.7/solr5_4

- Create the new collection

Issue this command on the command line or the command in the output below on the URL of a browser that can hit the SOLR VM
./bin/solr create -c statdx -d /home/john/conf -shards 1 -replicationFactor 2

Note: replicationFactor controls how many VMs you get replicated "cores" on, so if you have 3 VMs you want cores on, make that number 3 instead of 2.

You should see output something very much like this:

Connecting to Zookeeper at 192.168.56.5,192.168.56.6,192.168.56.7/solr5_4 ...
Re-using existing configuration directory statdx

Creating new collection 'statdx' using command:
http://192.168.56.15:8983/solr/admin/collections?action=CREATE&name=statdx
&numShards=1&replicationFactor=2&maxShardsPerNode=1
&collection.configName=statdx


```
{
"responseHeader":{
    "status":0,
    "QTime":5056},
"success":{"":{
    "responseHeader":{
          "status":0,
          QTime":4782},
    "core":"statdx_shard1_replica2"}}}
```


- Check Collection Status

For fun, go to the "other" VM - the one you did not issue the create command on and issue this command to check the status of the new collection.

./bin/solr healthcheck -192.168.56.5,192.168.56.6,
192.168.56.7/solr5_4 -c statdx

You should see something like this. NOTE: There are two JSON objects - one for each SOLR VM

(And there was much rejoicing!!!)


```
{
"collection":"statdx",
"status":"healthy",
"numDocs":0,
"numShards":1,
"shards":[{
      "shard":"shard1",
```

```
"status":"healthy",
"replicas":[
        {
        "name":"core_node1",
"url":"http://192.168.56.15:8983/solr/statdx_shard1_replica2/",
                "numDocs":0,
                "status":"active",
                "uptime":"0 days, 0 hours, 6 minutes, 40 seconds",
                "memory":"75.2 MB (%15.3) of 490.7 MB"},
                {
                "name":"core_node2",
"url":"http://192.168.56.16:8983/solr/statdx_shard1_replica1/",
                "numDocs":0,
                "status":"active",
                "uptime":"0 days, 0 hours, 6 minutes, 18 seconds",
                "memory":"33 MB (%6.7) of 490.7 MB",
                "leader":true}]}]}
```


- Verify the core directories are where they should be

Issue this command on both machines and see the correct statdx "core" directory

ls /opt/solr/server/solr

In this case, the output was the following.
We care about the directory name in bold font.
These directories were created “auto-magically” by the create command.

[solr5]
configsets README.txt solr.xml **statdx_shard1_replica2** zoo.cfg

[solr6]
configsets README.txt solr.xml **statdx_shard1_replica1** zoo.cfg

After verifying the directories are where they should be, go to a browser (No promises if you use Internet Exploder!) and hit the standard SOLR URL for each VM's IP address - everything

should be normal and you should be able to see and access the collection (statdx in this case) on EACH VM's Admin console. (You should have one "Leader" and one "follower" in the cloud graph.)

http://192.168.56.15:8983/solr/#/~cloud
http://192.168.56.16:8983/solr/#/~cloud

http://192.168.56.15:8983/solr/#/statdx_shard1_replica1/query
http://192.168.56.16:8983/solr/#/statdx_shard1_replica2/query

- Add another node AFTER the create collection in the previous step

This could be useful for disaster recovery, load "balancing" or backup. The data from the master SOLR instance should be downloaded automatically to this one.

If you want to add another node (replica, not shard) you can do this.

```
http://192.168.56.17:8983/solr/admin/cores?action=CREATE&name=statdx_shard1_replica3&collection=statdx&shard=shard1&collection.configName=statdx
```

Note: the IP address **must** be the VM or machine you want the new node to exist on. That means that SOLR 5.4 would need to be correctly installed etc… For my test, I cloned a VM with SOLR 5.4 installed and removed the collection directory created in the "Create the new Collection" step above.

- Get the SOLR data from Kafka into the newly made SOLR VMs.

Turn on the microservice and dump in all the SOLR docs.

First: On Zookeeper, using zkCli.sh, issue:

rmr /kafka/consumers

This is necessary or the microservice code won't get any new messages because Kafka keeps track of what messages it has sent to what "consumer".

This has to happen each time you run the microservice

(I have a TODO to add code that does it after each run using Zookeeper API)