Disclaimers

- Since college was creeping up on me during my development process of this application, I sprinted in order to build a stable and advanced POC that can create and search an IVrixDB Index in the production environment of SolrCloud, and in order to write all documentation (and JavaDoc). Because of this, I made several compromises:
    - Currently, the regexes are hard coded as a whitespace delimiter. This not only limits the fields extracted in search-time, but also limits the type of data that can be indexed into IVrixDB. It helped me build the architecture around the core idea, and these problems will be solved by replaced with configurable regexes later.
    - The data is restricted to be a JSON file with fields "_time", and "_raw", for the extracted timestamp and the raw log, respectively. This again was done to help me build the architecture around the core idea. In future, an automatic timestamp extractor will be added into the indexing process, and many types of log formats will be supported for ingestion.
    -
    - The code is currently embedded within Solr since it enabled me to develop quickly. The code will be taken out later in the future.
    - There are no Unit Tests in the code. Everything was tested as manual system tests, as detailed in the "IVrixDB Tests" document.
    - The code (in my opinion, of course) is clean and of high quality. However, there are places in the code that do not meet up to my standards, and I will *definitely* refactor as soon as I can.
    - I did not get the chance to fully develop a resource manager for the search component. The search results are being maintained for the entire run-time duration, and the system will take up more and more memory over time…I may develop a resource manager for this issue in the future.
    - The bucketing cutoff point is a simple criteria of bucket size. This should not be the case since bucket timespan is also an important factor. Work on this will be done in the future.
    -
    - To cover all interfaces between objects/classes, I wrote JavaDoc comments for all public classes and all public functions. In doing so, I caught a lot of small details that are important, a lot of TODOs, and a good number of large concepts that were missing from the written documentation. However, there is a side-effect to this rule-of-thumb -- most JavaDoc on getters/setters are noise, and on some other functions. I will do a large scan over the entire code base to remove such noisy JavaDoc comments in the future.

Warnings/Heads-Up (For Developers)

- All code that reads from Solr's ClusterState object is, in my opinion, kind-of dirty and hard-to-read. Be careful when changing code that reads from ClusterState...this is on my top-priority for refactoring. Below are some strange properties that I found in the ClusterState object:
    - A "gone" replica state does not exist, even though it does exist in the Solr Admin UI. A "gone" replica is a replica that is "active" but not on a node that is live.

    - Replica.getCoreName returns the name of the physical directory of the relica (example – “ivrixdb_8983_replica_n3”), not the actual core name
    - Replica.getName returns the name of the Core (example – “core_node2”)

-
- If a replica dies during indexing, Solr will wait for up to 3min to see if the that replica will boot up. Afterwards, it will continue to index.
-
- The attach/detach procedure is dangerous. It is not supported by Solr, and if done improperly, it can delete the bucket entirely. This is because Solr’s fault tolerancy program for failing to “ADDREPLICA” is to reverse the process, i.e delete what was created. It also is not reliable with attach/detach/roll-to-cold, but with a few quirky hacks, I got them stable, but with some disclaimers:
    - DELETE REPLICA is bugged after bucket has been indexed into (without restarting Solr.) this bug was fixed by changing a low-level Lucene index lock type. This solution should be reviewed VERY CAREFULLY, as it may pose dangers down the line.
    - RELOAD bucket fixes the issue of detaching buckets after indexing. It would be a good idea to figure a different solution.
    - After a node has died during the indexing of the bucket, and the bucket was then rolled to cold, deleting the replicas on the dead node, but it still has removed replicas in shard leader election. This messes up the ATTACH command later on. To solve this, during ROLL-TO-COLD, there is a function that cleans up the potential mess in ZK ("terms" zk node of the collection.)
    - If a DETACH replica command was executed on a replica from a dead node, and afterwards that node boots up, it will DELETE the replica instead of DETACHing it. And so, replicas that are “gone” are not part of the DETACH procedure.
    - Attach replica utilizes a local datadir ellipses “../”. I have tried many other ways of retrieving the datadir, but all the rest failed or deleted the bucket…

Bugs

- There is a bug where if a delete Index command occurred when buckets were created before re-boot, it would stall and mess up the state of the index (this is a bug with the collections API.) To fix this, simply restart the nodes. Since the bug was not bothersome for me, and I needed to continue to build the architecture, I allowed this bug to stay.
    - There is an additional procedure that was not implemented: detached buckets are not being physically removed from the directory, and therefore must be done manually.