Item 375
git_comments:

git_commits:

1. **summary:** Record clusterId performing change in CosmosDB (#4312)
   message: Record clusterId performing change in CosmosDB (#4312) Records the clusterId (if configured) with the updated document while persisting in ComosDB. This is an optional property. By default no `clusterId` info is saved

github_issues:

1. **title:** Multi Region OpenWhisk deployment using CosmosDB
   body: For OpenWhisk setups using CosmosDB its possible to deploy multiple OpenWhisk cluster in different regions via using multi region CosmosDB setup. Purpose of this issue is to track high level changes required to support such deployment ## Metadata and CosmosDB CosmosDB supports multi region deployments where it supports transparent replication of data across regions. In doing that it supports 2 modes 1. `Multi Master` - In this mode multiple regions can handles writes locally. When used with OpenWhisk this mode can be enabled for `actovations` collection as objects in this collection are immutable and hence no conflict resolution need to be performed 2. `Single Master` - In this mode only one region handles the write while reads work locally for each region. This mode would be used for `whisks` and `subjects` collection ### Separate Account for Collections As multi master is needed only for activation we would need to support configuring separate connection details on a per collection basis. This is now possible via #4198 ### Cache Invalidation OpenWhisk cluster uses a Kafka topic `cacheInvalidation` to communicate changes to any cached entity. Messages on this topic are of the form ```json {"instanceId":"controller0","key":{"mainId":"guest/hello"}} ``` When deploying multiple seprate cluster of OpenWhisk which do not share same Kafka instance we would need a way to propagate the cache change event across cluster. For CosmosDB based setups this can be done by using [CosmosDB ChangeFeed][1] support. It enables reading changes that are made to any specific collection. #### Cache Invalidation Service (#4314) To propagate changes across cluster we can implement a new service which makes use of the [changefeedprocessor][2] library to listen to changes happening in `whisks` and `subject` collection and then convert them into Kafka message events which can be sent to `cacheInvalidation`. So each cluster would have its own cache invalidation service running which would ensure that changes done by other clusters are picked up and routed to local cluster `cacheInvalidation` topic #### Handling Deletes (#4339) Per [change feed support docs][1] there is no direct support for listening to deletes > The change feed includes inserts and update operations made to documents within the collection. You can capture deletes by setting a "soft-delete" flag within your documents in place of deletes So we would need to implement such a support in our `CosmosDBArtifactStore` where 1. Upon delete we set a `_deleted` flag 2. Fliter out such soft deleted entries from query results 3. Use this flag in cache invalidation service to detect deletes #### Identifying Cluster making the change (#4312) Controllers would pickup changes from `cacheInvalidation` topic and then invalidate cached entry if the instanceId of the message differs. When propagating changes from other clusters we would need to determine if change picked via CosmosDB changefeed occurred from current cluster or other cluster. For this purpose we would need to add some metadata while performing any CRUD operation which identifies the current cluster. ## Code attachments and S3 (#4392) S3 support replication of a bucket only to 1 other region. For multi region setups we can leverage using CDN like CloudFront to act as a local cache which reduces latency for ready code binary in other regions [1]: https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed [2]: https://github.com/Azure/azure-documentdb-changefeedprocessor-java
2. **title:** Multi Region OpenWhisk deployment using CosmosDB
   body: For OpenWhisk setups using CosmosDB its possible to deploy multiple OpenWhisk cluster in different regions via using multi region CosmosDB setup. Purpose of this issue is to track high level changes required to support such deployment ## Metadata and CosmosDB CosmosDB supports multi region deployments where it supports transparent replication of data across regions. In doing that it supports 2 modes 1. `Multi Master` - In this mode multiple regions can handles writes locally. When used with OpenWhisk this mode can be enabled for `actovations` collection as objects in this collection are immutable and hence no conflict resolution need to be performed 2. `Single Master` - In this mode only one region handles the write while reads work locally for each region. This mode would be used for `whisks` and `subjects` collection ### Separate Account for Collections As multi master is needed only for activation we would need to support configuring separate connection details on a per collection basis. This is now possible via #4198 ### Cache Invalidation OpenWhisk cluster uses a Kafka topic `cacheInvalidation` to communicate changes to any cached entity. Messages on this topic are of the form ```json {"instanceId":"controller0","key":{"mainId":"guest/hello"}} ``` When deploying multiple seprate cluster of OpenWhisk which do not share same Kafka instance we would need a way to propagate the cache change event across cluster. For CosmosDB based setups this can be done by using [CosmosDB ChangeFeed][1] support. It enables reading changes that are made to any specific collection. #### Cache Invalidation Service (#4314) To propagate changes across cluster we can implement a new service which makes use of the [changefeedprocessor][2] library to listen to changes happening in `whisks` and `subject` collection and then convert them into Kafka message events which can be sent to `cacheInvalidation`. So each cluster would have its own cache invalidation service running which would ensure that changes done by other clusters are picked up and routed to local cluster `cacheInvalidation` topic #### Handling Deletes (#4339) Per [change feed support docs][1] there is no direct support for listening to deletes > The change feed includes inserts and update operations made to documents within the collection. You can capture deletes by setting a "soft-delete" flag within your documents in place of deletes So we would need to implement such a support in our `CosmosDBArtifactStore` where 1. Upon delete we set a `_deleted` flag 2. Fliter out such soft deleted entries from query results 3. Use this flag in cache invalidation service to detect deletes #### Identifying Cluster making the change (#4312) Controllers would pickup changes from `cacheInvalidation` topic and then invalidate cached entry if the instanceId of the message differs. When propagating changes from other clusters we would need to determine if change picked via CosmosDB changefeed occurred from current cluster or other cluster. For this purpose we would need to add some metadata while performing any CRUD operation which identifies the current cluster. ## Code attachments and S3 (#4392) S3 support replication of a bucket only to 1 other region. For multi region setups we can leverage using CDN like CloudFront to act as a local cache which reduces latency for ready code binary in other regions [1]: https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed [2]: https://github.com/Azure/azure-documentdb-changefeedprocessor-java
3. **title:** Multi Region OpenWhisk deployment using CosmosDB
   body: For OpenWhisk setups using CosmosDB its possible to deploy multiple OpenWhisk cluster in different regions via using multi region CosmosDB setup. Purpose of this issue is to track high level changes required to support such deployment ## Metadata and CosmosDB CosmosDB supports multi region deployments where it supports transparent replication of data across regions. In doing that it supports 2 modes 1. `Multi Master` - In this mode multiple regions can handles writes locally. When used with OpenWhisk this mode can be enabled for `actovations` collection as objects in this collection are immutable and hence no conflict resolution need to be performed 2. `Single Master` - In this mode only one region handles the write while reads work locally for each region. This mode would be used for `whisks` and `subjects` collection ### Separate Account for Collections As multi master is needed only for activation we would need to support configuring separate connection details on a per collection basis. This is now possible via #4198 ### Cache Invalidation OpenWhisk cluster uses a Kafka topic `cacheInvalidation` to communicate changes to any cached entity. Messages on this topic are of the form ```json {"instanceId":"controller0","key":{"mainId":"guest/hello"}} ``` When deploying multiple seprate cluster of OpenWhisk which do not share same Kafka instance we would need a way to propagate the cache change event across cluster. For CosmosDB based setups this can be done by using [CosmosDB ChangeFeed][1] support. It enables reading changes that are made to any specific collection. #### Cache Invalidation Service (#4314) To propagate changes across cluster we can implement a new service which makes use of the [changefeedprocessor][2] library to listen to changes happening in `whisks` and `subject` collection and then convert them into Kafka message events which can be sent to `cacheInvalidation`. So each cluster would have its own cache invalidation service running which would ensure that changes done by other clusters are picked up and routed to local cluster `cacheInvalidation` topic #### Handling Deletes (#4339) Per [change feed support docs][1] there is no direct support for listening to deletes > The change feed includes inserts and update operations made to documents within the collection. You can capture deletes by setting a "soft-delete" flag within your documents in place of deletes So we would need to implement such a support in our `CosmosDBArtifactStore` where 1. Upon delete we set a `_deleted` flag 2. Fliter out such soft deleted entries from query results 3. Use this flag in cache invalidation service to detect deletes #### Identifying Cluster making the change (#4312) Controllers would pickup changes from `cacheInvalidation` topic and then invalidate cached entry if the instanceId of the message differs. When propagating changes from other clusters we would need to determine if change picked via CosmosDB changefeed occurred from current cluster or other cluster. For this purpose we would need to add some metadata while performing any CRUD operation which identifies the current cluster. ## Code attachments and S3 (#4392) S3 support replication of a bucket only to 1 other region. For multi region setups we can leverage using CDN like CloudFront to act as a local cache which reduces latency for ready code binary in other regions [1]: https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed [2]: https://github.com/Azure/azure-documentdb-changefeedprocessor-java
4. **title:** Multi Region OpenWhisk deployment using CosmosDB
   body: For OpenWhisk setups using CosmosDB its possible to deploy multiple OpenWhisk cluster in different regions via using multi region CosmosDB setup. Purpose of this issue is to track high level changes required to support such deployment ## Metadata and CosmosDB CosmosDB supports multi region deployments where it supports transparent replication of data across regions. In doing that it supports 2 modes 1. `Multi Master` - In this mode multiple regions can handles writes locally. When used with OpenWhisk this mode can be enabled for `actovations` collection as objects in this collection are immutable and hence no

conflict resolution need to be performed 2. `Single Master` - In this mode only one region handles the write while reads work locally for each region. This mode would be used for `whisks` and `subjects` collection ### Separate Account for Collections As multi master is needed only for activation we would need to support configuring separate connection details on a per collection basis. This is now possible via #4198 ### Cache Invalidation OpenWhisk cluster uses a Kafka topic `cacheInvalidation` to communicate changes to any cached entity. Messages on this topic are of the form ```json {"instanceId":"controller0","key":{"mainId":"guest/hello"}} ``` When deploying multiple seprate cluster of OpenWhisk which do not share same Kafka instance we would need a way to propagate the cache change event across cluster. For CosmosDB based setups this can be done by using [CosmosDB ChangeFeed][1] support. It enables reading changes that are made to any specific collection. #### Cache Invalidation Service (#4314) To propagate changes across cluster we can implement a new service which makes use of the [changefeedprocessor][2] library to listen to changes happening in `whisks` and `subject` collection and then convert them into Kafka message events which can be sent to `cacheInvalidation`. So each cluster would have its own cache invalidation service running which would ensure that changes done by other clusters are picked up and routed to local cluster `cacheInvalidation` topic #### Handling Deletes (#4339) Per [change feed support docs][1] there is no direct support for listening to deletes > The change feed includes inserts and update operations made to documents within the collection. You can capture deletes by setting a "soft-delete" flag within your documents in place of deletes So we would need to implement such a support in our `CosmosDBArtifactStore` where 1. Upon delete we set a `_deleted` flag 2. Fliter out such soft deleted entries from query results 3. Use this flag in cache invalidation service to detect deletes #### Identifying Cluster making the change (#4312) Controllers would pickup changes from `cacheInvalidation` topic and then invalidate cached entry if the instanceId of the message differs. When propagating changes from other clusters we would need to determine if change picked via CosmosDB changefeed occurred from current cluster or other cluster. For this purpose we would need to add some metadata while performing any CRUD operation which identifies the current cluster. ## Code attachments and S3 (#4392) S3 support replication of a bucket only to 1 other region. For multi region setups we can leverage using CDN like CloudFront to act as a local cache which reduces latency for ready code binary in other regions [1]: https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed [2]: https://github.com/Azure/azure-documentdb-changefeedprocessor-java

github_issues_comments:

1. Nice writeup, thanks! Can we potentially normalize between CouchDB and CosmosDB here by listening to the CouchDB changes feed as well for cache invalidation?
2. Not well versed with CouchDB global replication support. Does it support across regions? Or we need to read from primary and just take care of cache invalidation part using its change feed?

github_pulls:

1. **title:** Record clusterId performing change in CosmosDB
body: Enables storing the clusterId information as part of document json in db. ## Description For Multi Region deployments (#4277) we need to differentiate which cluster made change to a document while emitting the cache invalidation event. This PR enables storing a `_clusterId` attribute which can be configured to uniquely identify current cluster. Later this attribute can be used by the cache invalidation service to filter out changes which are done by same cluster and only convert those changes from other cluster to local cache invalidation kafka topic ## Related issue and scope <!--- Please include a link to a related issue if there is one. --> - [ ] I opened an issue to propose and discuss this change (#4277) ## My changes affect the following components <!--- Select below all system components are affected by your change. --> <!--- Enter an `x` in all applicable boxes. --> - [ ] API - [ ] Controller - [ ] Message Bus (e.g., Kafka) - [ ] Loadbalancer - [ ] Invoker - [ ] Intrinsic actions (e.g., sequences, conductors) - [ ] Data stores (e.g., CouchDB) - [ ] Tests - [ ] Deployment - [ ] CLI - [ ] General tooling - [ ] Documentation ## Types of changes <!--- What types of changes does your code introduce? Use `x` in all the boxes that apply: --> - [ ] Bug fix (generally a non-breaking change which closes an issue). - [ ] Enhancement or new feature (adds new functionality). - [ ] Breaking change (a bug fix or enhancement which changes existing behavior). ## Checklist: <!--- Please review the points below which help you make sure you've covered all aspects of the change you're making. --> - [ ] I signed an [Apache CLA](https://github.com/apache/incubator-openwhisk/blob/master/CONTRIBUTING.md). - [ ] I reviewed the [style guides](https://github.com/apache/incubator-openwhisk/wiki/Contributing:-Git-guidelines#code-readiness) and followed the recommendations (Travis CI will check :). - [ ] I added tests to cover my changes. - [ ] My changes require further changes to the documentation. - [ ] I updated the documentation where necessary.

github_pulls_comments:

1. # [Codecov](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=h1) Report > Merging [#4312](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=desc) into [master](https://codecov.io/gh/apache/incubator-openwhisk/commit/2ddd3c9cfd1b31176a54634f254707774a90f4f0?src=pr&el=desc) will **decrease** coverage by `4.9%`. > The diff coverage is `9.09%`. [![Impacted file tree graph](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/graphs/tree.svg?width=650&token=l0YmsiSAso&height=150&src=pr)](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=tree) ```diff @@ Coverage Diff @@ ## master #4312 +/- ## == - Coverage 85.68% 80.78% -4.91% == Files 168 168 Lines 7861 7870 +9 Branches 523 524 +1 == - Hits 6736 6358 -378 - Misses 1125 1512 +387 ``` | [Impacted Files](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=tree) | Coverage Δ | | |---|---|---| | [...nwhisk/core/database/cosmosdb/CosmosDBConfig.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJDb25maWcuc | `94.11% <ø> (-5.89%)` | :arrow_down: | | [...core/database/cosmosdb/CosmosDBArtifactStore.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJBcnRpZmFjdE | `0% <0%> (-95.36%)` | :arrow_down: | | [...penwhisk/core/database/cosmosdb/CosmosDBUtil.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJVdGlsLnNjYV | `96% <100%> (+0.16%)` | :arrow_up: | | [...core/database/cosmosdb/RxObservableImplicits.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvUnhPYnNlcnZhYmxlSW1wb | `0% <0%> (-100%)` | :arrow_down: | | [...sk/core/database/cosmosdb/CosmosDBViewMapper.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJWaWV3TWFw | `0% <0%> (-92.6%)` | :arrow_down: | | [...whisk/core/database/cosmosdb/CosmosDBSupport.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJTdXBwb3J0L | `0% <0%> (-84.62%)` | :arrow_down: | | [...abase/cosmosdb/CosmosDBArtifactStoreProvider.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvcmUvZGF0YWJhc2UvY29zbW9zZGIvQ29zbW9zREJBcnRpZmFjdE | `4% <0%> (-52%)` | :arrow_down: | | [...in/scala/org/apache/openwhisk/common/Counter.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL2NvbW1vbi9Db3VudGVyLnNjYWxh) | `40% <0%> (-20%)` | :arrow_down: | | [.../scala/org/apache/openwhisk/utils/Exceptions.scala](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree#diff-Y29tbW9uL3NjYWxhL3NyYy9tYWluL3NjYWxhL29yZy9hcGFjaGUvb3BlbndoaXNrL3V0aWxzL0V4Y2VwdGlvbnMuc2NhbGE=) | `20% <0%> (-20%)` | :arrow_down: | | ... and [3 more](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312/diff?src=pr&el=tree-more) | | ------ [Continue to review full report at Codecov](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=continue). > **Legend** - [Click here to learn more](https://docs.codecov.io/docs/codecov-delta) > `Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data` > Powered by [Codecov](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=footer). Last update [2ddd3c9...0d29a5c](https://codecov.io/gh/apache/incubator-openwhisk/pull/4312?src=pr&el=lastupdated). Read the [comment docs](https://docs.codecov.io/docs/pull-request-comments).

github_pulls_reviews:

1. **body:** Minor: It may be worth noting (comment in the cluster id config?) that clusterId may change in the case where doc is created from one cluster, and edited from another, when both clusters use the same collection.
   label: documentation
2. **body:** Documented it now as part of scaladoc for `CosmosDBConstants`
   label: documentation

jira_issues:

jira_issues_comments: