Cadence Operation Guide

NOTE 1: Most of the docs are already moved to Cadence-docs (which hosts http://cadenceworkflow.io): Latest PR.

NOTE 2: Feel free to give suggestion / comments to improve in here, or in Cadence-docs(preferred)

Background

This document will cover things that you need to know to run a Cadence cluster in production. Topics including: setup, monitoring, and maintenance.

Setup

This section will help to understand what you need for setting up a Cadence cluster. Obviously, you need Cadence's dependency --- a database and ElastiCache(if you need Advanced visibility feature), and metric server. But you also need to understand lots of config options in Cadence server. There are two main types of configs in Cadence server, static config and dynamic config.

Cadence server configuration

Static configs

There are lots of configs in Cadence. Usually the default values or the recommended values in development.yaml should be good to go. However you need to understand some others:

Config name	Explanation	Recommended value
numHistoryShards	This is the most important one in Cadence config. It will be a fixed number in the cluster forever. The only way to change it is to migrate to another cluster. Refer to Migrate cluster section. Some facts about it: 1. Each workflow will be mapped to a single shard. Within a shard, all the workflow creation/updates are serialized.	1K~16K depending on the size ranges of the cluster you expect to run, and the instance size.

	2. Each shard will be assigned to only one History node to own the shard, using a Consistent Hashing Ring. Each shard will consume a small amount of memory/CPU to do background processing. Therefore, a single History node cannot own too many shards. You may need to figure out a good number range based on your instance size(memory/CPU). 3. Also, you can't add an infinite number of nodes to a cluster because this config is fixed. When the number of History nodes is closed or equal to numHistoryShards, there will be some History nodes that have no shards assigned to it. This will be wasting resources. Based on above, you don't want to have a small number of shards which will limit the maximum size of your cluster. You also don't want to have a too big number, which will require you to have a quite big initial size of the cluster. Also, typically a production cluster will start with a smaller number and then we add more nodes/hosts to it. But to keep high availability, it's recommended to use at least 4 nodes for each service(Frontend/History/Matching) at the beginning.	
ringpop	This is the config to let all nodes of all services connected to each other. **ALL** the bootstrap nodes **MUST** be reachable by ringpop when a service is starting up, within a MaxJoinDuration. defaultMaxJoinDuration is 2 minutes. It's not required that bootstrap nodes need to be Frontend/History or Matching. In fact, it can be running none of them as long as it runs Ringpop protocol.	For **dns** mode: Recommended to put the DNS of Frontend service For **hosts or hostfile** mode: A list of Frontend service node addresses if using hosts mode. Again make sure all the bootstrap nodes are reachable at startup.

publicClient	The Cadence Frontend service addresses that internal Cadence system(like system workflows) need to talk to. NOTE: After connected, all nodes in Ringpop will form a ring with identifiers of what service they serve. Ideally Cadence should be able to get Frontend address from there. But Ringpop doesn't expose this API yet.	Recommended be DNS of Frontend service, so that requests will be distributed to all Frontend nodes. Using localhost+Port or local container IP address+Port will not work if the IP/container is not running frontend
services.NAME.rpc	Configuration of how to listen to network ports and serve traffic. bindOnLocalHost:true will bind on 127.0.0.1. It's mostly for local development. In production usually you have to specify the IP that containers will use by using bindOnIP NAME is the matter for the "--services" option in the server startup command.	bindOnIP : an IP address that the container will serve the traffic with
services.NAME.pprof	Golang profiling service , will bind on the same IP as RPC	a port that you want to serve pprof request
services.Name.metrics	See Metrics&Logging section	
clusterMetadata	Cadence cluster configuration. enableGlobalDomain：true will enable Cadence Cross datacenter replication(aka XDC) feature. failoverVersionIncrement: This decides the maximum clusters that you will have replicated to each other at the same time. For example 10 is sufficient for most cases. masterClusterName: a master cluster must be one of the enabled clusters, usually the very first cluster to start. It is only meaningful for internal purposes. currentClusterName: current cluster name using this config file	See explanation.

	clusterInformation is a map from clusterName to the cluster configure initialFailoverVersion: each cluster must use a different value from 0 to failoverVersionIncrement-1. rpcName: must be “cadence-frontend”. Can be improved in this issue. rpcAddress: the address to talk to the Frontend of the cluster for inter-cluster replication. **Note that even if you don’t need XDC replication right now, if you want to migrate data stores in the future, you should enable xdc from every beginning. You just need to use the same name of cluster for both masterClusterName and currentClusterName. See more details in Migration section.**	
dcRedirectionPolicy	For allowing forwarding frontend requests from passive cluster to active clusters. Use	use “selected-apis-forwarding” as policy
archival	This is for archival history feature, skip if you don’t need it. See more in History Archival section	
domainDefaults	default config for each domain. Right now only being used for Archival feature.	
dynamicConfigClient	Dynamic config is a config manager that you can change config without restarting servers. It’s a good way for Cadence to keep high availability and make things easy to configure. Default cadence server is using FileBasedClientConfig. But you can implement the dynamic config interface if you have a better way to manage.	Same as the sample development config
blobstore	This is for archival history feature Default cadence server is using file based blob store implementation.	

persistence	configuration for data store / persistence layer. Values of DefaultStore VisibilityStore AdvancedVisibilityStore should be keys of map DataStores. DefaultStore is for core Cadence functionality. VisibilityStore is for basic visibility feature AdvancedVisibilityStore is for advanced visibility	

Dynamic configs

There are more dynamic configurations than static configurations. Dynamic configs can be changed at the run time without restarting any server instances. The format of dynamic configuration is defined here.

Here is the list of all the dynamic configurations defined from the code:

NOTE#1: the size related configuration numbers are based on byte.

NOTE#2: current default dynamic configuration is implemented as file based configuration. This feature will make it better to use as a real “dynamic” configuration.

NOTE#3: for <frontend,history,matching>.persistenceMaxQPS versus <frontend,history,matching>.persistenceGlobalMaxQPS --- persistenceMaxQPS is local for single node while persistenceGlobalMaxQPS is global for all node. persistenceGlobalMaxQPS is preferred if set as greater than zero. But by default it is zero so persistenceMaxQPS is being used.

TODO: *some default values are **N/A** because the default value is determined during the run time. We need some work to describe it in the text here.*

Config Key	Explanation	Default Values
Shared for all four services: Frontend/Matching/History/Worker		
system.enableGlobalDomain	key for enable global domain	based on static config value: clusterMetadata.EnableGlobalDomain
~~system.enableNewKafkaClient~~	~~key for using New Kafka client~~	~~#N/A~~
system.enableVisibilitySampling	key for enable visibility sampling	TRUE
system.enableReadFromClosedExecutionV2	key for enable read from cadence_visibility.closed_executions_v2	FALSE

system.advancedVisibilityWritingMode	key for how to write to advanced visibility	common.GetDefaultAdvancedVisibilityWritingMode(isAdvancedVisConfigExist)
history.emitShardDiffLog	whether emit the shard diff log	FALSE
system.enableReadVisibilityFromES	key for enable read from elastic search	based on static config value: PersistenceConfig.AdvancedVisibilityStore
frontend.disableListVisibilityByFilter	config to disable list open/close workflow using filter	FALSE
system.historyArchivalStatus	key for the status of history archival	#N/A
system.enableReadFromHistoryArchival	key for enabling reading history from archival store	#N/A
system.visibilityArchivalStatus	key for the status of visibility archival	#N/A
system.enableReadFromVisibilityArchival	key for enabling reading visibility from archival store	#N/A
system.enableDomainNotActiveAutoForwarding	whether enabling DC auto forwarding to active cluster for signal / start / signal with start API if domain is not active	TRUE
system.enableGracefulFailover	whether enabling graceful failover	FALSE
system.transactionSizeLimit	the largest allowed transaction size to persistence	#N/A
system.MaxRentionDays	The maximum number of retention days allowed for domain while registering.	30
system.minRetentionDays	the minimal allowed retention days for domain	1
system.maxDecisionStartToCloseSeconds	the minimal allowed decision start to close timeout in seconds	240
system.disallowQuery	the key to disallow query for a domain	FALSE
system.enablePriorityTaskProcessor	the key for enabling priority task processor	TRUE
system.enableAuthorization	the key to enable authorization for a domain	#N/A
limit.blobSize.error	the per event blob size limit, exceeding this will reject requests	2*1024*1024
limit.blobSize.warn	the per event blob size limit for warning	256*1024
limit.historySize.error	the per workflow execution history size limit, exceeding this will kill workflows	200*1024*1024
limit.historySize.warn	the per workflow execution history size limit for warning	50*1024*1024
limit.historyCount.	the per workflow execution history event count limit,	200*1024

error	exceeding this will kill workflows	
limit.historyCount.warn	the per workflow execution history event count limit for warning	50*1024
limit.maxIDLength	the length limit for various IDs, including: Domain, TaskList, WorkflowID, ActivityID, TimerID,WorkflowType, ActivityType, SignalName, MarkerName, ErrorReason/FailureReason/CancelCause, Identity, RequestID	1000
limit.maxIDWarnLength	the warn length limit for various IDs, including: Domain, TaskList, WorkflowID, ActivityID, TimerID, WorkflowType, ActivityType, SignalName, MarkerName, ErrorReason/FailureReason/CancelCause, Identity, RequestID	150
Frontend Service		
frontend.persistenceMaxQPS	the max qps frontend host can query DB	2000
frontend.persistenceGlobalMaxQPS	the max qps frontend cluster can query DB	0
frontend.visibilityMaxPageSize	default max size for ListWorkflowExecutions in one page	1000
frontend.visibilityListMaxQPS	max qps frontend can list open/close workflows	10
frontend.esVisibilityListMaxQPS	max qps frontend can list open/close workflows from ElasticSearch	30
frontend.esIndexMaxResultWindow	ElasticSearch index setting max_result_window	10000
frontend.historyMaxPageSize	default max size for GetWorkflowExecutionHistory in one page	common.GetHistoryMaxPageSize
frontend.rps	workflow rate limit per second	1200
frontend.domainrps	workflow domain rate limit per second per domain per frontend instance	1200
frontend.globalDomainrps	workflow domain rate limit per second for the whole Cadence cluster	0
frontend.historyMgrNumConns	for persistence cluster.NumConns	10
frontend.throttledLogRPS	the rate limit on number of log messages emitted per second for throttled logger	20
frontend.shutdownDrainDuration	the duration of traffic drain during shutdown	0
frontend.enableClientVersionCheck	enables client version check for frontend	FALSE
frontend.maxBadBinaries	the max number of bad binaries in domain config	domain.MaxBadBinaries
frontend.validSearchAttributes	legal indexed keys that can be used in list APIs	definition.GetDefaultIndexedKeys()
frontend.sendRawWorkflowHistory	whether to enable raw history retrieving	sendRawWorkflowHistory
frontend.searchAtt	the limit of number of keys	100

ributesNumberOfKeysLimit		
frontend.searchAttributesSizeOfValueLimit	the size limit of each value	2*1024
frontend.searchAttributesTotalSizeLimit	the size limit of the whole map	40*1024
frontend.visibilityArchivalQueryMaxPageSize	the maximum page size for a visibility archival query	10000
frontend.visibilityArchivalQueryMaxRangeInDays	the maximum number of days for a visibility archival query	#N/A
frontend.visibilityArchivalQueryMaxQPS	the timeout for a visibility archival query	#N/A
frontend.domainFailoverRefreshInterval	the domain failover refresh timer	10*time.Second
frontend.domainFailoverRefreshTimerJitterCoefficient	the jitter for domain failover refresh timer jitter	0.1
Matching Service		
matching.rps	request rate per second for each matching host	1200
matching.persistenceMaxQPS	the max qps matching host can query DB	3000
matching.persistenceGlobalMaxQPS	the max qps matching cluster can query DB	0
matching.minTaskThrottlingBurstSize	the minimum burst size for task list throttling	1
matching.getTasksBatchSize	the maximum batch size to fetch from the task buffer	1000
matching.longPollExpirationInterval	the long poll expiration interval in the matching service	time.Minute
matching.enableSyncMatch	to enable sync match	TRUE
matching.updateAckInterval	the interval for update ack	1*time.Minute
matching.idleTasklistCheckInterval	the IdleTasklistCheckInterval	5*time.Minute
matching.maxTasklistIdleTime	the max time tasklist being idle	5*time.Minute
matching.outstandingTaskAppendsThreshold	the threshold for outstanding task appends	250
matching.maxTaskBatchSize	max batch size for task writer	100
matching.maxTask	the max batch size for range deletion of tasks	100

DeleteBatchSize		
matching.throttledLogRPS	the rate limit on number of log messages emitted per second for throttled logger	20
matching.numTasklistWritePartitions	the number of write partitions for a task list. It's a little tricky to use this config. See Client worker setup section.	1
matching.numTasklistReadPartitions	the number of read partitions for a task list. It's a little tricky to use this config. See Client worker setup section.	1
matching.forwarderMaxOutstandingPolls	the max number of inflight polls from the forwarder	1
matching.forwarderMaxOutstandingTasks	the max number of inflight addTask/queryTask from the forwarder	1
matching.forwarderMaxRatePerSecond	the max rate at which add/query can be forwarded	10
matching.forwarderMaxChildrenPerNode	the max number of children per node in the task list partition tree	20
matching.shutdownDrainDuration	the duration of traffic drain during shutdown	0
History Service		
history.rps	request rate per second for each history host	3000
history.persistenceMaxQPS	the max qps history host can query DB	9000
history.persistenceGlobalMaxQPS	the max qps history cluster can query DB	0
history.historyVisibilityOpenMaxQPS	max qps one history host can write visibility open_executions	300
history.historyVisibilityClosedMaxQPS	max qps one history host can write visibility closed_executions	300
history.longPollExpirationInterval	the long poll expiration interval in the history service	time.Second*20
history.cacheInitialSize	initial size of history cache	128
history.cacheMaxSize	max size of history cache	512
history.cacheTTL	TTL of history cache	time.Hour
history.shutdownDrainDuration	the duration of traffic drain during shutdown	0
history.eventsCacheInitialSize	initial count of events cache	128
history.eventsCacheMaxSize	max count of events cache	512
history.eventsCacheMaxSizeInBytes	max size of events cache in bytes	0
history.eventsCach	TTL of events cache	time.Hour

eTTL		
history.eventsCacheGlobalEnable	enables global cache over all history shards	FALSE
history.eventsCacheGlobalInitialSize	initial count of global events cache	4096
history.eventsCacheGlobalMaxSize	max count of global events cache	131072
history.acquireShardInterval	interval that timer used to acquire shard	time.Minute
history.acquireShardConcurrency	number of goroutines that can be used to acquire shards in the shard controller.	1
history.standbyClusterDelay	the artificial delay added to standby cluster's view of active cluster's time	5*time.Minute
history.standbyTaskMissingEventsResendDelay	the amount of time standby cluster's will wait (if events are missing)before calling remote for missing events	15*time.Minute
history.standbyTaskMissingEventsDiscardDelay	the amount of time standby cluster's will wait (if events are missing)before discarding the task	25*time.Minute
history.taskProcessRPS	the task processing rate per second for each domain	1000
history.taskSchedulerType	the task scheduler type for priority task processor	int(task.SchedulerTypeWRR)
history.taskSchedulerWorkerCount	the number of workers per host in task scheduler	200
history.taskSchedulerShardWorkerCount	the number of worker per shard in task scheduler	0
history.taskSchedulerQueueSize	the size of task channel for host level task scheduler	10000
history.taskSchedulerShardQueueSize	the size of task channel for shard level task scheduler	200
history.taskSchedulerDispatcherCount	the number of task dispatcher in task scheduler (only applies to host level task scheduler)	1
history.taskSchedulerRoundRobinWeight	the priority weight for weighted round robin task scheduler	common.ConvertIntMapToDynamicConfigMapProperty(DefaultTaskPriorityWeight)
history.activeTaskRedispatchInterval	the active task redispatch interval	5*time.Second
history.standbyTaskRedispatchInterval	the standby task redispatch interval	30*time.Second
history.taskRedispatchIntervalJitterCoefficient	the task redispatch interval jitter coefficient	0.15
history.standbyTaskReReplicationContextTimeout	the context timeout for standby task re-replication	3*time.Minute

history.queueProcessorEnableSplit	indicates whether processing queue split policy should be enabled	FALSE
history.queueProcessorSplitMaxLevel	the max processing queue level	2 // 3 levels, start from 0
history.queueProcessorEnableRandomSplitByDomainID	indicates whether random queue split policy should be enabled for a domain	FALSE
history.queueProcessorRandomSplitProbability	the probability for a domain to be split to a new processing queue	0.01
history.queueProcessorEnablePendingTaskSplitByDomainID	indicates whether pending task split policy should be enabled	FALSE
history.queueProcessorPendingTaskSplitThreshold	the threshold for the number of pending tasks per domain	common.ConvertIntMapToDynamicConfigMapProperty(DefaultPendingTaskSplitThreshold)
history.queueProcessorEnableStuckTaskSplitByDomainID	indicates whether stuck task split policy should be enabled	FALSE
history.queueProcessorStuckTaskSplitThreshold	the threshold for the number of attempts of a task	common.ConvertIntMapToDynamicConfigMapProperty(DefaultStuckTaskSplitThreshold)
history.queueProcessorSplitLookAheadDurationByDomainID	the look ahead duration when spliting a domain to a new processing queue	20*time.Minute
history.queueProcessorPollBackoffInterval	the backoff duration when queue processor is throttled	5*time.Second
history.queueProcessorPollBackoffIntervalJitterCoefficient	backoff interval jitter coefficient	0.15
history.queueProcessorEnablePersistQueueStates	indicates whether processing queue states should be persisted	FALSE
history.queueProcessorEnableLoadQueueStates	indicates whether processing queue states should be loaded	FALSE
history.timerTaskBatchSize	batch size for timer processor to process tasks	100
history.timerTaskWorkerCount	number of task workers for timer processor	10
history.timerTaskMaxRetryCount	max retry count for timer processor	100
history.timerProce	retry count for timer processor get failure operation	5

ssorGetFailureRetryCount		
history.timerProcessorCompleteTimerFailureRetryCount	retry count for timer processor complete timer operation	10
history.timerProcessorUpdateShardTaskCount	update shard count for timer processor	#N/A
history.timerProcessorUpdateAckInterval	update interval for timer processor	30*time.Second
history.timerProcessorUpdateAckIntervalJitterCoefficient	the update interval jitter coefficient	0.15
history.timerProcessorCompleteTimerInterval	complete timer interval for timer processor	60*time.Second
history.timerProcessorFailoverMaxPollRPS	max poll rate per second for timer processor	1
history.timerProcessorMaxPollRPS	max poll rate per second for timer processor	20
history.timerProcessorMaxPollInterval	max poll interval for timer processor	5*time.Minute
history.timerProcessorMaxPollIntervalJitterCoefficient	the max poll interval jitter coefficient	0.15
history.timerProcessorSplitQueueInterval	the split processing queue interval for timer processor	1*time.Minute
history.timerProcessorSplitQueueIntervalJitterCoefficient	the split processing queue interval jitter coefficient	0.15
history.timerProcessorMaxRedispatchQueueSize	the threshold of the number of tasks in the redispatch queue for timer processor	10000
history.timerProcessorEnablePriorityTaskProcessor	indicates whether priority task processor should be used for timer processor	TRUE
history.timerProcessorEnableMultiCursorProcessor	indicates whether multi-cursor queue processor should be used for timer processor	FALSE
history.timerProcessorMaxTimeShift	the max shift timer processor can have	1*time.Second
history.timerProcessorHistoryArchivalSizeLimit	the max history size for inline archival	500*1024
history.timerProcessorArchivalTimeLi	the upper time limit for inline history archival	1*time.Second

mit		
history.transferTaskBatchSize	batch size for transferQueueProcessor	100
history.transferProcessorFailoverMaxPollRPS	max poll rate per second for transferQueueProcessor	1
history.transferProcessorMaxPollRPS	max poll rate per second for transferQueueProcessor	20
history.transferTaskWorkerCount	number of worker for transferQueueProcessor	10
history.transferTaskMaxRetryCount	max times of retry for transferQueueProcessor	100
history.transferProcessorCompleteTransferFailureRetryCount	times of retry for failure	10
history.transferProcessorUpdateShardTaskCount	update shard count for transferQueueProcessor	#N/A
history.transferProcessorMaxPollInterval	max poll interval for transferQueueProcessor	1*time.Minute
history.transferProcessorMaxPollIntervalJitterCoefficient	the max poll interval jitter coefficient	0.15
history.transferProcessorSplitQueueInterval	the split processing queue interval for transferQueueProcessor	1*time.Minute
history.transferProcessorSplitQueueIntervalJitterCoefficient	the split processing queue interval jitter coefficient	0.15
history.transferProcessorUpdateAckInterval	update interval for transferQueueProcessor	30*time.Second
history.transferProcessorUpdateAckIntervalJitterCoefficient	the update interval jitter coefficient	0.15
history.transferProcessorCompleteTransferInterval	complete timer interval for transferQueueProcessor	60*time.Second
history.transferProcessorMaxRedispatchQueueSize	the threshold of the number of tasks in the redispatch queue for transferQueueProcessor	10000
history.transferProcessorEnablePriorityTaskProcessor	indicates whether priority task processor should be used for transferQueueProcessor	TRUE
history.transferProcessorEnableMulti	indicates whether multi-cursor queue processor should be used for transferQueueProcessor	FALSE

CursorProcessor		
history.transferProcessorVisibilityArchivalTimeLimit	the upper time limit for archiving visibility records	200*time.Millisecond
history.replicatorTaskBatchSize	batch size for ReplicatorProcessor	100
history.replicatorTaskWorkerCount	number of worker for ReplicatorProcessor	10
history.replicatorReadTaskMaxRetryCount	the number of read replication task retry time	3
history.replicatorTaskMaxRetryCount	max times of retry for ReplicatorProcessor	100
history.replicatorProcessorMaxPollRPS	max poll rate per second for ReplicatorProcessor	20
history.replicatorProcessorUpdateShardTaskCount	update shard count for ReplicatorProcessor	#N/A
history.replicatorProcessorMaxPollInterval	max poll interval for ReplicatorProcessor	1*time.Minute
history.replicatorProcessorMaxPollIntervalJitterCoefficient	the max poll interval jitter coefficient	0.15
history.replicatorProcessorUpdateAckInterval	update interval for ReplicatorProcessor	5*time.Second
history.replicatorProcessorUpdateAckIntervalJitterCoefficient	the update interval jitter coefficient	0.15
history.replicatorProcessorMaxRedispatchQueueSize	the threshold of the number of tasks in the redispatch queue for ReplicatorProcessor	10000
history.replicatorProcessorEnablePriorityTaskProcessor	indicates whether priority task processor should be used for ReplicatorProcessor	FALSE
history.executionMgrNumConns	persistence connections number for ExecutionManager	50
history.historyMgrNumConns	persistence connections number for HistoryManager	50
history.maximumBufferedEventsBatch	max number of buffer event in mutable state	100
history.maximumSignalsPerExecution	max number of signals supported by single execution	10000
history.shardUpdateMinInterval	the minimal time interval which the shard info can be updated	5*time.Minute
history.shardSync	the minimal time interval which the shard info should be	5*time.Minute

MinInterval	sync to remote	
history.shardSyncMinInterval	the sync shard jitter coefficient	#N/A
history.defaultEventEncoding	the encoding type for history events	string(common.EncodingTypeThriftRW)
history.numArchiveSystemWorkflows	key for number of archive system workflows running in total	1000
history.archiveRequestRPS	the rate limit on the number of archive request per second	300 // should be much smaller than frontend RPS
history.enableAdminProtection	whether to enable admin checking	FALSE
history.adminOperationToken	the token to pass admin checking	common.DefaultAdminOperationToken
history.historyMaxAutoResetPoints	the key for max number of auto reset points stored in mutableState	DefaultHistoryMaxAutoResetPoints
history.enableParentClosePolicy	whether to ParentClosePolicy	TRUE
history.parentClosePolicyThreshold	decides that parent close policy will be processed by sys workers(if enabled) ifthe number of children greater than or equal to this threshold	10
history.numParentClosePolicySystemWorkflows	key for number of parentClosePolicy system workflows running in total	10
history.throttledLogRPS	the rate limit on number of log messages emitted per second for throttled logger	4
history.stickyTTL	to expire a sticky tasklist if no update more than this duration	time.Hour*24*365
history.decisionHeartbeatTimeout	for decision heartbeat	time.Minute*30
history.DropStuckTaskByDomain	whether stuck timer/transfer task should be dropped for a domain	FALSE
System Worker Service		
worker.persistenceMaxQPS	the max qps worker host can query DB	500
worker.persistenceGlobalMaxQPS	the max qps worker cluster can query DB	0
worker.replicatorMetaTaskConcurrency	the number of coroutine handling metadata related tasks	#N/A
worker.replicatorTaskConcurrency	the number of coroutine handling non metadata related tasks	#N/A
worker.replicatorMessageConcurrency	the max concurrent tasks provided by messaging client	#N/A
worker.replicatorActivityBufferRetryCount	the retry attempt when encounter retry error on activity	#N/A
worker.replicatorHi	the retry attempt when encounter retry error on history	#N/A

storyBufferRetryCount		
worker.replicationTaskMaxRetryCount	the max retry count for any task	#N/A
worker.replicationTaskMaxRetryDuration	the max retry duration for any task	#N/A
worker.replicationTaskContextDuration	the context timeout for apply replication tasks	#N/A
worker.workerReReplicationContextTimeout	the context timeout for end to end re-replication process	#N/A
worker.enableReplication	the feature flag for kafka replication	#N/A
worker.indexerConcurrency	the max concurrent messages to be processed at any given time	1000
worker.ESProcessorNumOfWorkers	num of workers for esProcessor	1
worker.ESProcessorBulkActions	max number of requests in bulk for esProcessor	1000
worker.ESProcessorBulkSize	max total size of bulk in bytes for esProcessor	2<<24 // 16MB
worker.ESProcessorFlushInterval	flush interval for esProcessor	1*time.Second
worker.EnableArchivalCompression	indicates whether blobs are compressed before they are archived	#N/A
worker.WorkerHistoryPageSize	indicates the page size of history fetched from persistence for archival	#N/A
worker.WorkerTargetArchivalBlobSize	indicates the target blob size in bytes for archival, actual blob size may vary	#N/A
worker.ArchiverConcurrency	controls the number of coroutines handling archival work per archival workflow	50
worker.ArchivalsPerIteration	controls the number of archivals handled in each iteration of archival workflow	1000
worker.DeterministicConstructionCheckProbability	controls the probability of running a deterministic construction check for any given archival	#N/A
worker.BlobIntegrityCheckProbability	controls the probability of running an integrity check for any given archival	#N/A
worker.TimeLimitPerArchivalIteration	controls the time limit of each iteration of archival workflow	archiver.MaxArchivalIterationTimeout()
worker.throttledLogRPS	the rate limit on number of log messages emitted per second for throttled logger	20
worker.scannerPersistenceMaxQPS	the maximum rate of persistence calls from worker.Scanner	100
worker.taskListScannerEnabled	indicates if task list scanner should be started as part of worker.Scanner	TRUE
worker.historyScan	indicates if history scanner should be started as part of	TRUE

nerEnabled	worker.Scanner	
worker.executions ScannerEnabled	indicates if executions scanner should be started as part of worker.Scanner	FALSE
worker.executions ScannerConcurrency	indicates the concurrency of concrete execution scanner	25
worker.executions ScannerBlobstore FlushThreshold	indicates the flush threshold of blobstore in concrete execution scanner	100
worker.executions ScannerActivityBatchSize	indicates the batch size of scanner activities	25
worker.executions ScannerPersistencePageSize	indicates the page size of execution persistence fetches in concrete execution scanner	1000
worker.executions ScannerInvariantCollectionMutableState	indicates if mutable state invariant checks should be run	TRUE
worker.executions ScannerInvariantCollectionHistory	indicates if history invariant checks should be run	TRUE
worker.currentExecutionsScannerEnabled	indicates if current executions scanner should be started as part of worker.Scanner	FALSE
worker.currentExecutionsConcurrency	indicates the concurrency of current executions scanner	25
worker.currentExecutionsBlobstoreFlushThreshold	indicates the flush threshold of blobstore in current executions scanner	100
worker.currentExecutionsActivityBatchSize	indicates the batch size of scanner activities	25
worker.currentExecutionsPersistencePageSize	indicates the page size of execution persistence fetches in current executions scanner	1000
worker.currentExecutionsScannerInvariantCollectionHistory	indicates if history invariant checks should be run	FALSE
worker.currentExecutionsInvariantCollectionMutableState	indicates if mutable state invariant checks should be run	TRUE
worker.enableBatcher	decides whether start batcher in our worker	FALSE
system.enableParentClosePolicyWorker	decides whether or not enable system workers for processing parent close policy task	TRUE
system.enableStickyQuery	indicates if sticky query should be enabled per domain	TRUE

Cross DC replication feature		
history.ReplicationTaskFetcherParallelism	determines how many go routines we spin up for fetching tasks	1
history.ReplicationTaskFetcherAggregationInterval	determines how frequently the fetch requests are sent	2*time.Second
history.ReplicationTaskFetcherTimerJitterCoefficient	the jitter for fetcher timer	0.15
history.ReplicationTaskFetcherErrorRetryWait	the wait time when fetcher encounters error	time.Second
history.ReplicationTaskFetcherServiceBusyWait	the wait time when fetcher encounters service busy error	60*time.Second
history.ReplicationTaskProcessorErrorRetryWait	the initial retry wait when we see errors in applying replication tasks	50*time.Millisecond
history.ReplicationTaskProcessorErrorRetryMaxAttempts	the max retry attempts for applying replication tasks	5
history.ReplicationTaskProcessorNoTaskInitialWait	the wait time when not ask is returned	2*time.Second
history.ReplicationTaskProcessorCleanupInterval	determines how frequently the cleanup replication queue	1*time.Minute
history.ReplicationTaskProcessorCleanupJitterCoefficient	the jitter for cleanup timer	0.15
history.ReplicationTaskProcessorReadHistoryBatchSize	the batch size to read history events	5
history.ReplicationTaskProcessorStartWait	the wait time before each task processing batch	5*time.Second
history.ReplicationTaskProcessorStartWaitJitterCoefficient	the jitter for batch start wait timer	0.9
history.ReplicationTaskProcessorHostQPS	the qps of task processing rate limiter on host level	1500
history.ReplicationTaskProcessorShardQPS	the qps of task processing rate limiter on shard level	5
history.ReplicationTaskGenerationQPS	the wait time between each replication task generation qps	100

history.EnableConsistentQuery	indicates if consistent query is enabled for the cluster	TRUE
history.EnableConsistentQueryByDomain	indicates if consistent query is enabled for a domain	FALSE
history.MaxBufferedQueryCount	indicates the maximum number of queries which can be buffered at a given time for a single workflow	1
history.mutableStateChecksumGenProbability	the probability [0-100] that checksum will be generated for mutable state	0
history.mutableStateChecksumVerifyProbability	the probability [0-100] that checksum will be verified for mutable state	0
history.mutableStateChecksumInvalidateBefore	the epoch timestamp before which all checksums are to be discarded	0
history.ReplicationEventsFromCurrentCluster	a feature flag to allow cross DC replicate events that generated from the current cluster	FALSE
history.NotifyFailoverMarkerInterval	determines the frequency to notify failover marker	5*time.Second
history.NotifyFailoverMarkerTimerJitterCoefficient	the jitter for failover marker notifier timer	0.15

Tuning Cadence Cluster: Todo

Dependency Setup

This PR will remove the hard requirement of Kafka when there is no advanced visibility feature.

Database

Cadence officially supports Cassandra, MySQL and Postgres as persistence storage. You can also implement with other databases as plugin and build with Cadence repo.
Here we take use of MySQL as an example.

Best Practices for Database Connection Configuration

- Connection is shared within a Cadence server host
- For each host, The max number of connections it will consume is maxConn of defaultStore + maxConn of visibilityStore.
- The total max number of connections your Cadence cluster will consume is the summary from all hosts(from Frontend/Matching/History/SysWorker services)
- Frontend and history nodes need both default and visibility Stores, but matching and sys workers only need default Stores, they don't need to talk to visibility DBs.

- For default Stores, history service will take the most connection, then Frontend/Matching. SysWorker will use much less than others
- Default Stores is for Cadence' core data model, which requires strong consistency. So it cannot use replicas. VisibilityStore is not for core data models. It's recommended to use a separate DB for visibility store if using DB based visibility.
- Visibility Stores usually take much less connection as the workload is much lightweight(less QPS and no explicit transactions).
- Visibility Stores require eventual consistency for read. So it can use replicas.
- MaxIdelConns should be less than MaxConns, so that the connections can be distributed better across hosts.

Metric&Logging

Cadence support emitting metrics through M3/Prometheus and Statsd. A feature request is created to let Cadence support any other metric emitter like Datadog.

Advanced Visibility

This is optional, you only need it when you want the advanced visibility feature.
To enable this feature:

1. Register index schema on ElasticSearch
    a. You don't need tools like Cassandra/SQL, all you need is run two CURL commands following this script.
        i. Create a index template by using the schema , choose v6/v7 based on your ElasticSearch version
        ii. Create an index follow the index template, remember the name
2. Register topic on Kafka
    a. set up the right number of partitions based on your expected throughput(can be scaled up later), remember the name
3. Configure Cadence for ElasticSearch + Kafka
    a. This is an example configuration to follow for ElasticSearch, and based on the full structure config, you may add some other fields like AuthN.
    b. This is an example configuration to follow for Kafka, and based on the full structure config, you may add some other fields like AuthN.

History Archival

WebUI

This is optional to run a Cadence server.

Usage and Maintenance

This includes how to use and maintain a Cadence cluster for both clients and server clusters.

Client worker setup

Java and Go clients should have almost the same guidance for setup, except that the Java client has a few more configurations about threading. That's because Java language doesn't have native support for lightweight threading like Golang goroutines.

Scalable tasklist

By default a tasklist is not scalable enough to support hundreds of tasks per second. That's mainly because each tasklist is assigned to a Matching service node, and dispatching tasks in a tasklist is in sequence.

In the past, Cadence recommended using multiple tasklists to start workflow/activity. The idea is quite simple. You just need to make a list of tasklists and randomly pick one when starting workflows. And then when starting workers, let them listen to all the tasklists.

Today, Cadence has a feature called "Scalable tasklist". It will divide a tasklist into multiple logical partitions, which can distribute tasks to multiple Matching service nodes. By default this feature is not enabled because there is some performance penalty on the server side, plus people believe that it's not common that a tasklist needs to support more than 100 tasks per second.

You must make a dynamic configuration change in Cadence server to use this feature:
matching.numTasklistWritePartitions
and
matching.numTasklistReadPartitions

matching.numTasklistWritePartitions is the number of partitions when a Cadence server sends a task to the tasklist.
matching.numTasklistReadPartitions is the number of partitions when your worker accepts a task from the tasklist.

There are a few things to know when using this feature:

1. Always make sure **matching.numTasklistWritePartitions <= matching.numTasklistReadPartitions** . Otherwise there may be some tasks that are sent to a tasklist partition but no poller(worker) will be able to pick up.
2. Because of above, when scaling down the number of partitions, you must decrease the WritePartitions first, to wait for a certain time to ensure that tasks are drained, and then decrease ReadPartitions.
3. Both domain names and taskListName should be specified in the dynamic config. An example of using this feature. See more details about dynamic config format using file based dynamic config.

```
matching.numTasklistWritePartitions:
  - value: 10
    constraints:
      domainName: "samples-domain"
      taskListName: "aScalableTasklistName"
matching.numTasklistReadPartitions:
  - value: 10
    constraints:
      domainName: "samples-domain"
      taskListName: "aScalableTasklistName"
```

NOTE #1: the value must be integer without double quotes.
NOTE #2: in the future, this scalable tasklist can be enabled automatically without worries. Issue filed here.

Writing Workflow

There are a few things you need to learn when writing workflows.

1. Setting correct/proper timeout/retry configs. Timeout/retry are the most important part for a Cadence workflow/activity to be resilient to any failure. Please read through the Cadence document about timeout/retry for workflow and activity.
2. Some common caveats:
    a. Activity with retry will only write ActivityStartedEvent once activity is closed(completed or fail/timeout). During the activity execution, activity status is stored in its workflow. Therefore you should use a Describe CLI command or WebUI for the “pending activities” section of the workflow, to look at the activity status including: how many attempts it has tried, and what is the last error returned, etc.
    b. ScheduleToCloseTimeout is not required for activity with retry. Because it will be recalculated from the retry config: ExpirationInterval or Max of workflow execution timeout.

3. Writing workflow orchestration code following some good patterns. Here is an article that summarizes some recommendations to write a workflow.

Client deployment

The key thing needed to be aware of is non deterministic error(NDE) of a workflow. NDE doesn't kill your workflows, but it will clog workflows and also waste worker/server CPU/memory.

To prevent NDE, you should test workflow code as much as you can:

1. Unit test with TestEnvironment from SDK
2. Unit test to replay history when making changes
3. Integration test in local/staging/QA environment
4. Use Versioning&SideEffect if there must be a nondeterministic change.

You may want to do deployment to mitigate NDE, to fix a bad code change event it doesn't cause NDE. But usually it’s not recommended to rollback deployment directly. Because it could cause NDE issues:
For example you have rolled out code to execute workflow with a new activity A, and a workflow run executed activityA but the run hasn't finished. Then if you roll back the code, the workflow run won't find activity A. That's a non-deterministic error.

Instead, most of the time you need to :

1. If you have to roll back, you need to make sure all runs are not using the new code, you can wait, or using reset feature(primitive Reset or AutoReset) to reset the progress
2. You can apply a hotfix. If there is already a NDE and you don't have to rollback, you can roll out a fix. You probably need to use workflow.Now() or workflow.GetBinaryChecksum() to branch out, letting the old run go to the old code so that they are not running into NDE.

Above are all captured in this article.

Using Domain XDC replication

1. Enable global domain feature in static config. Let say you create clusterDCA and clusterDCB. And pick clusterDCA as master cluster.
   The only difference of a master cluster” is that it is responsible for domain registration. Master can be changed later but it needs to be the same across all clusters.

   And ClusterMeta config of clusterDCA will be

```
clusterMetadata:
  enableGlobalDomain: true
  failoverVersionIncrement: 10
  masterClusterName: "clusterDCA"
  currentClusterName: "clusterDCA"
  clusterInformation:
    clusterDCA:
      enabled: true
      initialFailoverVersion: 1
      rpcName: "cadence-frontend"
      rpcAddress: "<>:<>"
    clusterDCB:
      enabled: true
      initialFailoverVersion: 0
      rpcName: "cadence-frontend"
      rpcAddress: "<>:<>"
```

And ClusterMeta config of clusterDCB will be

```
clusterMetadata:
  enableGlobalDomain: true
  failoverVersionIncrement: 10
  masterClusterName: "clusterDCA"
  currentClusterName: "clusterDCB"
  clusterInformation:
    clusterDCA:
      enabled: true
      initialFailoverVersion: 1
      rpcName: "cadence-frontend"
      rpcAddress: "<>:<>"
    clusterDCB:
      enabled: true
      initialFailoverVersion: 0
```

rpcName: "cadence-frontend"
rpcAddress: "<>:<>"

2. Monitor cluster replication by looking at metrics
3. Register a global domain

./cadence --do <domain_name> domain register --global_domain true --clusters clusterDCA clusterDCB --active_cluster clusterDCA

4. Run some workflow and failover domain from one to another

./cadence --do <domain_name> domain update --active_cluster clusterDCB

5. Then the domain should be failed over to clusterDCB. Now worklfows are read-only in clusterDCA. So your workers polling tasks from clusterDCA will become idle.

Note #1: that even though clusterDCA is standy/read-only for this domain, it can be active for another domain. So being active/standy is per domain basis not per clusters. In other words, for example if you use XDC in case of DC failure of clusterDCA, you need to failover all domains from clusterDCA to clusterDCB.

Note #2: even though a domain is standy/read-only in a cluster, say clusterDCA, sending write requests(startWF, signalWF, etc) could still work because there is a forwarding component in the Frontend service. It will try to re-route the requests to an active cluster for the domain.

Restarting Server

You may want to do rolling restart to keep high availability

Upgrade server cluster

Things need to keep in mind before upgrading a cluster:

1. Database schema changes need to apply first.

2. Usually schema change is backward compatible. So rolling back usually is not a problem. It also means that Cadence allows running a mixed version of schema, as long as they are all greater than or equal to the required version of the server.
3. Other requirements for upgrading should be found in the release notes. It may contain information about config changes, or special rollback instructions if normal rollback may cause problems.
4. It's recommended to upgrade one minor version at a time. E.g, if you are at 0.10, you should upgrade to 0.11, stabilize it with running some normal workload to make sure that the upgraded server is happy with the schema changes. After ~1 hour, then upgrade to 0.12. then 0.13. etc.
5. The reason above is that for each minor upgrade, you should be able to follow the release notes about what you should do for upgrading. The release notes may require you to run some commands. This will also help to narrow down the cause when something goes wrong.
6. Do not use "auto-setup" images to upgrade your schema.

For how to upgrade database schema, refer to this doc: SQL tool README
The tool makes use of a table called "schema_versions" to keep track of upgrading History. But there is no transaction guarantee for cross table operations. So in case of some error, you may need to fix or apply schema change manually.
Also, the schema tool by default will upgrade schema to the latest, so no manual is required. (you can also specify to let it upgrade to any place, like 0.14).

Database schema changes are versioned in the folders: Versioned Schema Changes for Default Store
and Versioned Schema Changes for Visibility Store if you use database for basic visibility instead of ElasticSearch.

Migrate server cluster

You may want to migrate to a different cluster for some reasons like:

1. Scale up -- move to a bigger cluster, with larger number of shards
2. Migrate to different storage, for example from Postgres/MySQL to Cassandra
3. Split traffic

Note that it's better to enable the XDC feature from the beginning for all domains. Because a local domain cannot be converted to a global one. You need to:

1. Enable global domain feature in static config.
2. Create your domain with the global domain feature(XDC). Since you only have one cluster, there is no replication happening. But you still need to tell the replication topology when creating your domain.

./cadence --do <domain_name> domain register --global_domain true --clusters <initialClustersName> --active_cluster <initialClusterName>

3. Later on, after you setting up a new cluster, you can add the cluster to domain replication config:

./cadence --do <domain_name> domain update --clusters <initialClusterName> <newClusterName>

4. It will start replication right after for all the active workflows.
5. After you are sure the new cluster is healthy, you then switch the active cluster to the new cluster.

./cadence --do <domain_name> domain update --active_cluster <newClusterName>

6. After some time, you make sure the new cluster is running fine, then remove the old cluster from replication:

./cadence --do <domain_name> domain update --clusters <newClusterName>

NOTE: if your current domain is not a global domain, you cannot use the XDC feature to migrate. The only way is to create a new global domain, and start to use the new domains for new workflows, and drain the old workflows to finish. After all old workflows are finish, you then use the above instruction to migrate.

Dashboard and Monitoring

This section will cover what metrics you need to use to build a dashboard and setup monitors.

A comprehensive list of client metrics are defined in Golang client's metric definition and Java client's metric definition. They are almost the same, except that the Java client has a few more metrics about threading. That's because Java language doesn't have native support for lightweight threading like Golang goroutines.

A comprehensive list of server metrics are defined in this definition.

Because of the complexity of server metrics, this section is written into three parts: client dashboard/monitors, basic server dashboard/monitors and advanced server dashboard/monitors. In "advanced server dashboard/monitors", metrics are introduced by topic groups. Usually you will need those metric dashboards/monitors for particular reasons.

This guide uses tables to describe metrics. One column is "Metric Query Pseudo Code". It's not a real query that you can directly copy/paste into M3/Prometheus/etc. But you should be able to apply with the M3/Prometheus/etc syntax.

Client dashboard/monitors

As a client user(developer) of Cadence workflows, if you don't own/run a Cadence service, then you only need to look at client metric dashboards, and setup monitors. Client metrics should be sufficient to tell you how well your cadence workflows and workers are running. When something goes wrong, you should be able to know if you should talk to the Cadence service owner.

Note: there are two exceptions here. Workflow timeout and size related metrics are emitted by Cadence service, but they should be treated as client side metrics as well, and should be monitored by client users.

Check here for an example of a client dashboard.

Title	Description	Metric Query Pseudo Code	Monitors
Workflow Counters	All operation counters over workflows, to let you know how many workflows are started/stopped/etc at what time.	domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist workflowtype:$workflowtype name:{cadence-workflow-start, cadence-workflow-completed,cadence-workflow-canceled,cadence-workflow-continue-as-new,cadence-workflow-failed,cadence-workflow-signal-with-start} type:counter	Maybe
Workflow Latency	The end to end workflow latency to record how long it takes for workflow executions to complete	domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist workflowtype:$workflowtype name:cadence-workflow-endtoend-latency timertype:{lower,median,p99,upper}	Yes, if you have expected latency for particular workflow types.
Activity Task Counter	All activity counters like activity started/completed/failed/timeouted, to tell you execution statuses of activities	domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:cadence-activity-poll-succeed type:counter	Maybe
Activity Execution Latency		domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:cadence-activity-execution-latency timertype:{lower,median,p99,upper}	**Yes**
Activity Poll		domain:$domain dc:$dc	Maybe

Counters		runtime_env:$env tasklist:$tasklist name:cadence-activity-poll-* type:counter	
Activity Poll Latency		domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:cadence-activity-poll-latency timertype:{lower,median,p99,upper}	
Decision Task Counters		domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:{cadence-decision-poll-succeed} type:counter	
Decision Execution Latency		domain:$domain dc:$dc tasklist:$tasklist runtime_env:$env name:cadence-decision-execution-latency timertype:{lower,median,p99,upper}	**Yes**
Decision Poll Counters		domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:cadence-decision-poll-* type:counter	
Decision Poll Latency		domain:$domain dc:$dc runtime_env:$env tasklist:$tasklist name:cadence-decision-poll-latency timertype:{lower,median,p99,upper}	
Workflow Timeout !!!NOTE:from server metrics		name:`workflow_timeout`	**Yes**
History Event Length !!!NOTE:from server metrics		name:history_count	**Yes**
History total size !!!NOTE:from	history size in bytes, use timer as a histogram of nanoSeconds	name:history_size	**Yes**

server metrics			
Blob size !!!NOTE:from server metrics	input/output of workflow/activity/signal size in bytes, use timer as a histogram of nanoSeconds	name:event_blob_size	**Yes**

Basic server dashboard/monitors

Check here for an example of a basic server dashboard.

Title	Description	Metric Query Pseudo Code	Monitors
Synchronous APIs Uptime		Use metric from network layer or computing service	**Yes**
Overall Service Availability		service:cadence-Frontend deployment:$deployment name:cadence_requests / cadence_errors	**Yes**
Persistence Availability		service:{cadence-History,cadence-Frontend,cadence-Matching} deployment:$deployment name:persistence_requests / persistence_erros	**Yes**
External Events per sec		service:cadence-History deployment:$deployment name:cadence_requests operation:{signalwithstartworkflowexecution,signalworkflowexecution,startworkflowexecution}	
Activities per sec		service:cadence-History deployment:$deployment name:cadence_requests operation:recordactivitytaskstarted	
Decisions per sec		service:cadence-History deployment:$deployment name:cadence_requests operation:recorddecisiontaskstarted	
Polls per sec		service:cadence-Frontend deployment:$deployment name:cadence_requests operation:pollforactivitytask	
Shard Movements	During restarts this is normal. Otherwise this indicates something wrong with the cluster consistency ring	service:cadence-History deployment:$deployment name:{membership_changed_count,shard_closed_count,shar	Probably

		ditem_created_count,sharditem_removed_count} operation:shardcontroller	
Service Restarts		service:cadence* deployment:$deployment name:restarts	
Service Panics		service:cadence* deployment:$deployment name:panics	**Yes**
Goroutines		This should be emitted by the computing service	
Timer Tasks Requests Vs Errors		service:cadence-History deployment:$deployment name:{task_requests,task_errors} operation:timeractive*	**Yes**
Timer Tasks Per Type		service:cadence-History deployment:$deployment name:task_requests operation:timeractive*	
Timer Task Latency		service:cadence-History deployment:$deployment name:task_latency operation:timeractive* timertype:$latency	**Yes**
Timer Tasks Per Domain		service:cadence-History deployment:$deployment name:task_requests_per_domain operation:timeractive*	
Timer Task $latency Latency per Domain		service:cadence-History deployment:$deployment name:task_latency_per_domain operation:timeractivetaskusertimer timertype:$latency	**Yes**
Transfer Tasks Requests Vs Errors		service:cadence-History deployment:$deployment name:{task_requests,task_errors} operation:transferactive*	**Yes**
Transfer Tasks Per Type		service:cadence-History deployment:$deployment name:task_requests operation:transferactive*	
Transfer Task Latency		service:cadence-History deployment:$deployment name:task_latency operation:transferactive* timertype:$latency	**Yes**

Transfer Tasks Per Domain		service:cadence-History deployment:$deployment name:task_requests_per_domain operation:transferactive*	
Transfer Task $latency Latency per Domain		service:cadence-History deployment:$deployment name:task_latency_per_domain operation:transferactive* timertype:$latency	**Yes**
Frontend Requests Top 20 Domains		service:cadence-Frontend deployment:$deployment type:counter	
Service Busy Top 20 Domains	For watching/monitoring hot domains	service:cadence-Frontend deployment:$deployment type:counter name:cadence_errors_service_busy	
External Requests Top 20 Domains	same as above	service:cadence-Frontend deployment:$deployment type:counter operation:{signalwithstartworkflowexecution,signalworkflowexecution,startworkflowexecution,terminateworkflowexecution,resetworkflowexecution,requestcancelworkflowexecution,listopenworkflowexecutions,listclosedworkflowexecutions}	
Workflow Worker Requests Top 20 Domains	same as above	service:cadence-Frontend deployment:$deployment type:counter operation:{pollfordecisiontask,getworkflowexecutionHistory,respondecisiontaskcompleted}	
Activity Worker Requests Top 20 Domains	same as above	service:cadence-Frontend deployment:$deployment type:counter operation:{pollforactivitytask,respondactivitytaskcompleted}	

Advanced dashboard/monitors

Frontend service

Title	Description	Metric Query Pseudo Code	Monitors

Request Counts Per API		service:cadence-Frontend deployment:$deployment name:cadence_requests	
Request Vs Error		service:cadence-Frontend deployment:$deployment name:{cadence_errors,cadence_requests}	**Yes**
Error Counts Per API		service:cadence-Frontend deployment:$deployment name:cadence_errors	**Yes**
Error Breakdown	For debugging the above metric	service:cadence-Frontend deployment:$deployment name:cadence_errors_bad_request	
Frontend API Latencies		service:cadence-Frontend deployment:$deployment name:cadence_latency timertype:$latency domain:all	**Yes**
GetReplicationMessage API requests		service:cadence-Frontend deployment:$deployment name:cadence_requests operation:getreplicationmessages	
GetReplicationMessage Latency		service:cadence-Frontend deployment:$deployment name:cadence_latency timertype:$latency operation:getreplicationmessages	**Yes**
GetReplicationMessage Errors		service:cadence-Frontend deployment:$deployment name:cadence_errors operation:getreplicationmessages	**Yes**
GetDomainReplicationMessage Requests		service:cadence-Frontend deployment:$deployment name:cadence_requests operation:getdomainreplicationmessages	
GetDomainReplicationMessage Latency		service:cadence-Frontend deployment:$deployment name:cadence_latency timertype:$latency operation:getdomainreplicationmessages	**Yes**
GetDomainReplicationMessage Errors		service:cadence-Frontend deployment:$deployment name:cadence_errors	

		operation:getdomainreplication messages	

History service

Title	Description	Metric Query Pseudo Code	Monitors
Errors Breakdown		service:cadence-History deployment:$deployment name:cadence_errors*	
Request Per API		service:cadence-History deployment:$deployment name:cadence_requests	
History API $latency Latencies		service:cadence-History deployment:$deployment name:cadence_latency timertype:$latency	**Yes**
History Clients Requests Vs Errors		service:cadence-History deployment:$deployment name:{cadence_client_errors,cadence_client_requests}	**Yes**
History Clients Request Count Per API		service:cadence-History deployment:$deployment name:cadence_client_requests	
History Client Error Counts Per API		service:cadence-History deployment:$deployment name:cadence_client_errors	
History Clients API Latencies		service:cadence-History deployment:$deployment name:cadence_client_latency timertype:$latency	**Yes**
Shard Movement		service: cadence-History deployment:$deployment operation:shardcontroller name:*count	
$latency Latencies		service:cadence-History deployment:$deployment operation:shardcontroller name:*latency timertype:$latency	**Yes**
Replication / Transfer $latency Lag		service:cadence-History deployment:$deployment operation: shardinfo name: {shardinfo_replication_lag,shardinfo_transfer_lag} type:timer timertype:$latency	**Yes**

Timer $latency Lag		service:cadence-History deployment:$deployment operation: shardinfo name: shardinfo_timer_lag type:timer timertype:$latency	**Yes**
Transfer Level $latency Diff		service:cadence-History deployment:$deployment operation: shardinfo name: shardinfo_transfer_diff type:timer timertype:$latency	**Yes**
Timer Level $latency Diff		service:cadence-History deployment:$deployment operation: shardinfo name: shardinfo_timer_diff type:timer timertype:$latency	**Yes**
Failover $latency Latency		service:cadence-History deployment:$deployment operation: shardinfo name: {shardinfo_transfer_failover_latency,shardinfo_timer_failover_latency} type:timer timertype:$latency	
Failover $latency In Progress		service:cadence-History deployment:$deployment operation:shardinfo name:{shardinfo_transfer_failover_in_progress,shardinfo_timer_failover_in_progress} type:timer timertype:$latency	
Pending $latency Tasks		service:cadence-History deployment:$deployment operation: shardinfo name: {shardinfo_replication_pending_task,shardinfo_transfer_active_pending_task,shardinfo_transfer_standby_pending_task,shardinfo_timer_active_pending_task,shardinfo_timer_standby_pending_task} type:timer timertype:$latency	**Yes**
Requests Vs Errors		service:cadence-History deployment:$deployment name:persistence_{errors,requests}	**Yes**
Errors Breakdown		service:cadence-History deployment:$deployment name:persistence_errors	
Requests Per Operation		service:cadence-History deployment:$deployment	

		name:persistence_requests	
Errors Per Operation		service:cadence-History deployment:$deployment name:persistence_errors	
Operation $latency Latencies		service:cadence-History deployment:$deployment name:persistence_latency timertype:$latency	
Visibility Sampled Counters		service:cadence-History deployment:$deployment name:{persistence_sampled,persistence_requests} operation:{recordworkflowexecutionstarted,recordworkflowexecutionclosed}	
Active Ack Level Update		service: cadence-History deployment: $deployment operation: timeractivequeueprocessor name: ack_level_update*	
Standby Ack Level Update		service: cadence-History deployment: $deployment operation: timerstandbyqueueprocessor name: ack_level_update*	
Active New Timer Notifications		service: cadence-History deployment: $deployment operation: timeractivequeueprocessor name: new_timer_notifications	
Standby New Timer Notifications		service: cadence-History deployment: $deployment operation: timerstandbyqueueprocessor name: new_timer_notifications	
Active Timer Request Vs Error		service: cadence-History deployment: $deployment operation: timeractivetask* name: task_requests type: counter	**Yes**
Standby Timer Request Vs Error		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_requests type: counter	
Active Timer Requests		service: cadence-History deployment: $deployment operation: timeractivetask*	

		name: task_requests type: counter	
Standby Timer Requests		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_requests type: counter	
Active Timer Errors		service: cadence-History deployment: $deployment operation: timeractivetask* name: task_errors type: counter	
Standby Timer Errors		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_errors type: counter	
Active Timer Processing $latency Latency		service: cadence-History deployment: $deployment operation: timeractivetask* name: task_latency_processing type: timer timertype: $latency	**Yes**
Standby Timer Processing $latency Latency		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_latency_processing type: timer timertype: $latency	**Yes**
Active Timer Queue $latency Latency		service: cadence-History deployment: $deployment operation: timeractivetask* name: task_latency_queue type: timer timertype: $latency	**Yes**
Standby Timer Queue $latency Latency		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_latency_queue type: timer timertype: $latency	**Yes**
Active Timer $latency Attempt		service: cadence-History deployment: $deployment operation: timeractivetask* name: task_attempt type: timer timertype: $latency	**Yes**
Standby Timer $latency Attempt		service: cadence-History deployment: $deployment operation: timerstandbytask* name: task_attempt type: timer timertype: $latency	**Yes**
Active Timer Task Timeout Details		service: cadence-History deployment: $deployment operation: timeractivetask*	

		name: *timeout type: counter	
Cron timer		service: cadence-History deployment: $deployment operation: timeractivetaskworkflowbackofftimer name: workflow_cron_backoff_timer	
Workflow retry timer		service: cadence-History deployment: $deployment operation: timeractivetaskworkflowbackofftimer name: workflow_retry_backoff_timer	
Number of Applied Replication Tasks		service: cadence-History deployment: $deployment operation: replicationtaskfetcher name: replication_tasks_applied	
Single Replication Task $latency Latency		service: cadence-History deployment: $deployment operation:replicationtaskfetcher name:task_latency_processing type: timer timertype: $latency	
Replication Batch $latency Latency		service: cadence-History deployment: $deployment operation:replicationtaskfetcher name:replication_tasks_applied_latency type: timer timertype: $latency	**Yes**
Fetch DLQ Size Error Count		service: cadence-History deployment: $deployment operation:replicationdlqstats name:replication_dlq_probe_failed	
Enqueue DLQ Error Count		service: cadence-History deployment: $deployment operation:replicationtaskfetcher name:replication_dlq_enqueue_failed	
Replication Queue Cleanup Error Count		service: cadence-History deployment: $deployment operation:replicationtaskcleanup name:replication_task_cleanup_failed	
Sync Shard Error Count		service: cadence-History deployment: $deployment operation:syncshardstatus name:syncshard_remote_failed	

Matching service

Title	Description	Metric Query Pseudo Code	Monitors
Overall Requests Vs Errors		service:cadence-Matching deployment:$deployment name:{cadence_errors,cadence_requests}	**Yes**
Errors Breakdown		service:cadence-Matching deployment:$deployment name:cadence_errors_*	
Request Counts Per API		service:cadence-Matching deployment:$deployment name:cadence_requests	
API Latencies		service:cadence-Matching deployment:$deployment name:cadence_latency timertype:$latency	**Yes**
PollForActivityTask		service:$service deployment:$deployment name:cadence_{errors*,requests} operation:pollforactivitytask	
PollForActivityTask Latency		service:$service deployment:$deployment timertype:{lower,median,p99,upper} name:cadence_latency operation:pollforactivitytask	
PollForDecisionTask		service:$service deployment:$deployment name:cadence_{errors*,requests} operation:pollfordecisiontask	
PollForDecisionTask Latency		service:$service deployment:$deployment timertype:{lower,median,p99,upper} name:cadence_latency operation:pollfordecisiontask	
AddActivityTask		service:$service deployment:$deployment name:cadence_{errors*,requests} operation:addactivitytask	
AddActivityTask Latency		service:$service deployment:$deployment timertype:{lower,median,p99,upper} operation:addactivitytask	
AddDecisionTas		service:$service	

k		deployment:$deployment name:cadence_{errors*,requests} operation:adddecisiontask	
AddDecisionTask Latency		service:$service deployment:$deployment timertype:{lower,median,p99,upper} operation:adddecisiontask	
TaskPoll		service:$service deployment:$deployment name:poll_{errors,success,success_sync,timeouts} operation:tasklistmgr	
Throttle Count		service:$service deployment:$deployment name:{sync,buffer}_throttle_count operation:tasklistmgr	
Decision Task Matcher Stats		service:$service deployment:$deployment operation:pollfordecisiontask type:counter	
Activity Task Matcher Stats		service:$service deployment:$deployment operation:pollforactivitytask type:counter	
TaskList Lease		service:$service deployment:$deployment name:{lease_failures,lease_requests,condition_failed_errors} operation:tasklistmgr	
Requests Vs Errors		service:cadence-Matching deployment:$deployment name:persistence_{errors,requests}	
Errors Breakdown		service:cadence-Matching deployment:$deployment name:persistence_errors_*	
Requests Per Operation		service:cadence-Matching deployment:$deployment name:persistence_requests	
Errors Per API		service:cadence-Matching deployment:$deployment name:persistence_errors	
Operation Latencies p99		service:cadence-Matching deployment:$deployment name:persistence_latency timertype:p99	
Request Vs		service:cadence-Matching	

Errors		deployment:$deployment cadence-role:History name:{cadence_client_errors,cadence_client_requests}	
Request Per API		service:cadence-Matching deployment:$deployment cadence-role:History name:cadence_client_requests	
Errors Per API		service:cadence-Matching deployment:$deployment cadence-role:History name:cadence_client_errors	
API Latency		service:cadence-Matching deployment:$deployment cadence-role:History name:cadence_client_latency timertype:$latency	
ActivityTask Async Match Latency		service:$service deployment:$deployment timertype:{lower,median,p99,upper} name:asyncmatch_latency operation:pollforactivitytask	
DecisionTask Async Match Latency		service:$service deployment:$deployment name:asyncmatch_latency timertype:{lower,median,p99,upper} operation:pollfordecisiontask	

Persistence

Title	Description	Metric Query Pseudo Code	Monitors
Request Vs Errors		deployment:$deployment name:{persistence_requests,persistence_errors}	**Yes**
Errors Breakdown		deployment:$deployment name:persistence_errors_* operation:$operation	
Requests Per Operation		deployment:$deployment name:persistence_requests operation:$operation	
Errors Per Operation		deployment:$deployment name:persistence_errors	
Persistence latency		deployment:$deployment name:persistence_latency	**Yes**

		timertype:$latency operation:$operation	
TXN Requests		deployment:$deployment name:persistence_requests operation:{updateworkflowexecution,createworkflowexecution,deleteworkflowexecution,leasetasklist,updatetasklist,deletetasklist,createshard,updateshard,createtask}	
TXN Latency		deployment:$deployment name:persistence_latency timertype:$latency operation:{updateworkflowexecution,createworkflowexecution,deleteworkflowexecution,leasetasklist,updatetasklist,deletetasklist,createshard,updateshard,createtask}	**Yes**

ElastiCache

Title	Description	Metric Query Pseudo Code	Monitors
Requests Vs Errors		deployment:$deployment name:{elasticsearch_requests,elasticsearch_errors}	**Yes**
Errors Breakdown		deployment:$deployment name:{elasticsearch_errors_*}	
Requests Per Operation		deployment:$deployment name:elasticsearch_requests	
Errors Per Operation		deployment:$deployment name:elasticsearch_errors	
API Latency		deployment:$deployment name:elasticsearch_latency timertype:$latency	**Yes**
ESProcessor requests		deployment:$deployment name:{es_processor_requests}	
Message Latency		deployment:$deployment name:index_processor_process_msg_latency timertype:$latency	**Yes**
Messages Consumed		service:cadence-worker name:kafka.partition.messages-* topic:cadence-visibility-* deployment:$deployment	

Consumer Start		service:cadence-worker type:counter name:kafka.consumer.{started,stopped} consumergroup:cadence-visibility-*-consumer deployment:$deployment	
DLQ		service:cadence-worker deployment:$deployment type:gauge name:messaging.consumer.kafka.partition.dlq-offset-lag cluster:*dlq consumergroup:cadence-visibility-*-consumer	**Yes**
Backlog		service:cadence-worker deployment:$deployment type:gauge name:messaging.consumer.kafka.partition.offset-lag cluster:*lossless consumergroup:cadence-visibility-*-consumer	
Indexer failures		deployment:$deployment name:index_processor_corrupted_data	**Yes**
ESProcessor failures		deployment:$deployment name:es_processor_corrupted_data	

Archival

Title	Description	Metric Query Pseudo Code	Monitors
Archivals Triggered Per Second		deployment:$deployment operation:archiverclient name:archiver_client_History_request	
Blobs Uploaded Per Second		cadence_role:blobstore deployment:$deployment operation:blobstoreclientupload name:cadence_client_requests	
Upload History Success Rate		failAllRetry = service:cadence-worker deployment:$deployment operation:archiver	**Yes**

		name:archiver_upload_failed_all_retries	
success = service:cadence-worker deployment:$deployment operation:archiver name:archiver_upload_success			
success			
Upload History $latency Latency		service:cadence-worker deployment:$deployment operation:archiver name:archiver_upload_with_retries_latency timertype:$latency	
Delete History Success Rate		service:cadence-worker deployment:$deployment timertype:count operation:archiver name:archiver_delete_with_retries_latency	
Delete History $latency Latency		service:cadence-worker deployment:$deployment name:archiver_delete_with_retries_latency timertype:$latency	**Yes**
Visibility Archival Success Rate		failAllRetry = service:cadence-worker deployment:$deployment operation:archiver name:archiver_handle_visibility_failed_all_retries	
success = service:cadence-worker deployment:$deployment operation:archiver name:archiver_handle_visibility_success			
success			
Visibility Archival $latency Latency		service:cadence-worker deployment:$deployment operation:archiver name:archiver_handle_visibility	

		_request_latency timertype:$latency	
Client Requests vs Inline Attempts		service:cadence-History deployment:$deployment operation:archiverclient name:{archiver_client_History_request,archiver_client_History_inline_archive_attempt,archiver_client_History_inline_archive_failure}	
Archival Signals vs Errors		deployment:$deployment operation:archiverclient name:archiver_client_sent_signal	**Yes**
Inline History Archival Success Rate		success = service:cadence-History deployment:$deployment operation:Historyarchiver name:History_archiver_archive_success	
nonretryable = service:cadence-History deployment:$deployment operation:Historyarchiver name:History_archiver_archive_non_retryable_error			
transient = service:cadence-History deployment:$deployment operation:Historyarchiver name:History_archiver_archive_transient_error			
success			
Inline Visibility Archival Success Rate		success = service:cadence-History deployment:$deployment operation:visibilityarchiver name:visibility_archiver_archive_success	
Archive Visibility		service:cadence-worker	

Activity $latency Latency		deployment:$deployment name:cadence_latency operation:archiverarchivevisibilityactivity timertype:$latency domain:all	
Archive Visibility Activity Non-retryable Errors		service:cadence-worker deployment:$deployment name:archiver_non_retryable_error operation:archiverarchivevisibilityactivity	**Yes**
List Archived Workflow Requests		service:cadence-Frontend deployment:$deployment name:cadence_requests operation:listarchivedworkflowexecutions	
List Archived Workflow $latency Latency		service:cadence-Frontend deployment:$deployment name:cadence_latency timertype:$latency operation:listarchivedworkflowexecutions	**Yes**
List Archived Workflow Errors		service:cadence-Frontend deployment:$deployment name:cadence_errors* operation:listarchivedworkflowexecutions	

Cadence system worker service

Title	Description	Metric Query Pseudo Code	Monitors
Replication Tasks Requests		service:cadence-History deployment:$target operation:replicator* name:task_requests	
Replication Tasks Errors		service:cadence-History deployment:$target operation:replicator* name:task_errors	**Yes**
Replication Task Generation Latency		service:cadence-History deployment:$target operation:replicator* name:task_latency type:timer timertype:$latency	**Yes**
Task Requests		service:cadence-worker deployment:$target	

		operation:sequentialtaskprocessing name:sequentialtask_submit_request type:counter	
Task $latency Latency		service:cadence-worker deployment:$target operation:sequentialtaskprocessing name:sequentialtask_submit_latency type:timer timertype:$latency	**Yes**
Task By Type		service:cadence-worker name:replicator_messages deployment:$target	
Task By Type $latency Latency		service:cadence-worker name:replicator_latency type: timer timertype:$latency deployment:$target	
Events Breakdown		service:cadence-History deployment:$target operation:replicateHistoryevents type: counter deployment:$target	
Task Queue $latency Size		service:cadence-worker deployment:$target operation:sequentialtaskprocessing name:sequentialtask_queue_size type:timer timertype:$latency	**Yes**
Batcher		service:cadence-worker deployment:$target operation:batcher name:*	
History Scavenger		service:cadence-worker deployment:$target operation:Historyscavenger name:*	