Original Description:-

Changes - Support primitives array in recasterOrchestration frameworkImpact - Its adding fallback to recaster creating instance framework so that Primitives and String array can be converted properly to objects instances.Testing - Tested locally and unit test for the same. Main thing to notice -> String[] or any primitive array when read using RecastOrchestrationUtils.fromJson() -> the json is converted to Map<String, Object> thus object here becomes ArrayList always thus the result will always come as List and not JavaArrays.

Generated Description:-

- Added tests for handling string, integer and boolean arrays in RecastOrchestrationUtilsTest.java
 - Added a new method createGivenInstance() in RecastObjectCreator.java to handle array instances
 - Added a new method createGivenInstance() in RecastObjectFactory.java to handle array instances
 - Added handling for array instances in fromMap() method of Recaster.java
 - Added a new method writeArrayInternal() in Recaster.java to handle array instances
---
Cosine Similarity: 0.8743926797881004

Original Description:-

Changes - Added new ParameterFieldExpressionProcessor to CDExpressionResolverImpact - Ideally should be no impact as its just a wrapper over all cd steps, sanity failure in any step is true for all cd steps.Why the change? Currently all edge cases for parameterField is not handled at cdExpressionResolverFunctor, thus added generic one and adopted to all cd steps.Testing - Run basic cd steps flows and its expressions
---
Generated Description:-

* The changes made in the code involve replacing the usage of `CDExpressionResolveFunctor` with `CDExpressionResolver` in multiple files.
 * This affects the functionality of the product by introducing a new class that supports both `EngineExpressionEvaluatorResolver` and `EngineExpressionServiceResolver`.
 * Additionally, there are some minor changes to imports and code formatting
---
Cosine Similarity: 0.8437395042190226

Original Description:-

Changes Introduce EngineExpressionResolver to have support for both EngineExpressionEvaluatorResolver and EngineExpressionServiceResolverImpact Currently

the framework is only introduced and files are movedTesting Basic CD and Custom stage pipeline with and without expressions are runImprovement Provide basic parameterField processor support in all engineExpression for other service to utilise and not create their custom ones, thus edge cases will be reduced.Secondly, this will reduce CI engine expression resolution for Initialise step, where if any step is matrix they use strategy expansion, where they convert into recast map, which has ParameterFieldDocumentMap, whose processor is added to StrategyExpressionEvalutor. After this, recaster conversion is not required, thus CPU processing will be saved and it will be more fast
---
Generated Description:-

* The changes involve moving the ExpressionModeMapper to a new package.
 * A new class called EngineExpressionResolver has been introduced to support both EngineExpressionEvaluatorResolver and EngineExpressionServiceResolver.
 * Imports have been fixed.
 * Some changes have been made to specific files such as AllowedValuesValidator.java, InputSetValidatorFactory.java, and AmbianceExpressionEvaluator.java.
 * A new copyright notice has been added

Cosine Similarity: 0.8343705318427038

Original Description:-

Changes - Fixing support for ternary, semi-colon, OR, NOT operator for concatenate expression flowImpact - Concatenate expression flow behind FF is only impactedTesting - UTs are added for it
---
Generated Description:-

- Added support for OR operator in concatenate expression flow
 - Added support for NOT operator in concatenate expression flow
 - Added support for json list operator and semicolon operator in concatenate expression flow
 - Fixed support for ternary, semicolon, OR, and NOT operator in concatenate expression flow
---
Cosine Similarity: 0.8960463093959123

Original Description:-

Changes - Add support for json list operator and semicolon operator.Impact - Only for expression concatenationTesting - Added UTs for the same
---
Generated Description:-

- Added support for ternary and when conditions in concatenate expression flow.

- Added support for json list operator and semicolon flow.
 - Modified the checkIfExpressionValueCanBeConcatenated method to handle concatenation inside method invocation and ternary operator.
 - Added NGJsonFunctor to the context.

Cosine Similarity: 0.8629363876777812

Original Description:-

Changes: Add support for NOT operator in concatenate expression flowImpact : In Concatenate expression flow onlyTesting : Added unit test for both NOT operators
---
Generated Description:-

- Added support for the ternary operator and the not operator in concatenate expression flow.
 - Added test cases to validate the functionality of the new changes.
 - Improved the StringReplacer class to handle the new operators.
 - Merged changes from the \'develop\' branch into the current branch.

Cosine Similarity: 0.8769051930910668

Original Description:-

Changes: Add support for OR operator in concatenate expression flowImpact : In Concatenate expression flow onlyTesting : Added unit test for both OR and AND operators
---
Generated Description:-

- Added support for ternary and when conditions in concatenate expression flow.
 - Modified the checkIfStringMathematicalOperator method to include OR operator.
 - Added test cases for OR and AND operators in EngineExpressionEvaluatorTest class.
 - Added test cases for complex double nesting with concatenate expressions with prefix combinations.
 - Added a new context for testing OR and AND operators.

Cosine Similarity: 0.903179858099786

Original Description:-

Add support for ternary and when condition in concatenate expression flow.Changes - concatenation inside method invocations like secrets functor etc were not covering all casesTesting: Added more UTs for the same
---
Generated Description:-

- Added support for ternary and when condition in concatenate expression flow.
 - Modified the StringReplacer class to handle method invocation inside an expression.
 - Updated the EngineExpressionEvaluatorTest class to include new test cases for the modified StringReplacer class.

Cosine Similarity: 0.9014579122384893

Original Description:-

Add FF to disable new json support in expressionsTesting: Local testing done
---
Generated Description:-

- Added a new feature flag to disable returning complex objects as JSON.
 - Modified the EngineExpressionEvaluator class to use the new feature flag to determine whether to use asJson or not.
 - Updated the build version to 1.34.6.
 - Modified the PlanNodeExecutionStrategy class to use the new feature flag to determine whether to add the feature flag to the enabledFeatureFlags list.

Cosine Similarity: 0.7930060953878189

Original Description:-

Support nested prefix expression resolutionImpact: The changes are behind new FFTesting: All existing UTs are passing and added new UT for prefix nested concatenate expression
---
Generated Description:-

- Added support for resolving nested prefix expressions.
 - Added a check to evaluate prefix combinations if the expression is evaluated as null.
 - Added a check to avoid cyclic loop while doing recursive check.
 - Modified the evaluateByCreatingExpressionV2 method to return null instead of the expression if the response has only rendered expressions.
 - Added a new variable var3 for testing concatenate expressions with prefix combinations.
 - Added a new context variable nestedConcatenate for testing complex double nesting with concatenate expressions with prefix combinations.
 - Modified the fetchPrefixes method to include the new context variable nestedConcatenate.

Cosine Similarity: 0.8596358892774189

Original Description:-

Same expression in ctx was going into recursive loop in concat feature, which is fixed
---
Generated Description:-

- Added support for string render concat in resolve evaluate behind FF PIE_EXPRESSION_CONCATENATION.
 - Added a check to return the evaluated expression if it is equal to the original expression.
 - Added handling for unresolved expressions to return the expression until which the resolution has happened correctly.
 - Added handling for EngineFunctorException and throwing EngineExpressionEvaluationException.
 - Minor formatting changes.

Cosine Similarity: 0.8023041222362659

Original Description:-

Adding string render concat support in resolve and evaluate behind FF PIE_EXPRESSION_CONCATENATIONThis method checks if string expression can be rendered (to concatenate or not)
Based on if left substring of expression or right substring of expression has first non-space char other than + operator.Example - <+v1><+v2> if either of them is string -> then resolve/evaluate both will allow concatenating this stringTesting: Added Unit tests for
a) Nested concatenate expression, example - <+pipeline.variables.var1>. It's value is <+env.variables.var2> . env variable var2 value is abc<+env.variables.var1>def
env variables var1 value is 424
b) Added tests for string concatenate on existing expressions to check backward compatible expressions are working
c) Added Same UTs under script FF on, so when we globally script support -> These expressions should still work
d) Failed expressions are failing

Generated Description:-

- Added support for string render concatenation in resolve and evaluate methods behind feature flag PIE_EXPRESSION_CONCATENATION
 - Modified evaluateExpressionInternal method to evaluate nested expressions if present
 - Modified shouldRenderAndNotEvaluate method to check for nested expressions before returning false
 - Added checkIfValueHasMethodInvocation method to StringReplacer class to check if right substring has method invocation
 - Added test cases for method invocations, unresolved expressions, and expressions used in path of existing expression

---
Cosine Similarity: 0.9317248585077577

Original Description:-

Adding string render concat support in resolve and evaluate behind FF PIE_EXPRESSION_CONCATENATIONThis method checks if string expression can be rendered (to concatenate or not)
Based on if left substring of expression or right substring of expression has first non-space char other than + operator.Example - <+v1><+v2> if either of them is string -> then resolve/evaluate both will allow concatenating this stringTesting: Added Unit tests for
a) Nested concatenate expression, example - <+pipeline.variables.var1>. It's value is <+env.variables.var2> . env variable var2 value is abc<+env.variables.var1>def
env variables var1 value is 424
b) Added tests for string concatenate on existing expressions to check backward compatible expressions are working
c) Added Same UTs under script FF on, so when we globally script support -> These expressions should still work
---
Generated Description:-

- Added support for new string concatenation in the expression engine.
 - Enabled sync step in GitOps.
 - Fetched pipeline health and execution data from a new timescale table.
 - Updated the EngineExpressionEvaluator and EngineExpressionSecretUtils classes.
 - Added new tests for valid nested expressions with script feature flags and valid nested expressions for resolve and evaluate with concatenate
---
Cosine Similarity: 0.8639091137916767

Original Description:-

Added concurrenthashmap on contexts map in MdcContextSetter, as its being modified by multiple threads thus the concurrentModificationExceptionTesting: Locally running all services and basic crud operations
---
Generated Description:-

- Added ConcurrentHashMap to contexts map in MdcContextSetter to prevent ConcurrentModificationException
 - Replaced direct assignment of contexts with new ConcurrentHashMap in MdcContextSetter
 - Updated build.patchVersion from 0 to 1 in pipeline-service build.properties file
---

Cosine Similarity: 0.8655514191008428

Original Description:-

Added concurrenthashmap on contexts map in MdcContextSetter, as its being modified by multiple threads thus the concurrentModificationExceptionTesting: Locally running service and basic crud operationsWe were not able to identify the root cause for this as per the code and stack trace, there are no multiple threads working on it, and its occurrence is only 25 in last one month, thus for safeguard added concurrentHashMap inside it
---
Generated Description:-

- Added ConcurrentHashMap to contexts map in MdcContextSetter to prevent ConcurrentModificationException.
 - The contexts map was being modified by multiple threads, hence the need for a thread-safe implementation.
 - The for loop now iterates over the new ConcurrentHashMap instead of the original contexts map.
 - This change improves the overall stability and reliability of the product.

Cosine Similarity: 0.8864618715109874

Original Description:-

Adding expression script support for pipeline expression v2 JSON supportTesting: Added unit test to confirm if the script support in normal flow is working for all flowsThe change is backward compatible as Unit test is working now.

Generated Description:-

- Added support for a new feature flag called PIE_EXECUTION_JSON_SUPPORT
 - Added a new method in EngineJexlContext to check if a feature flag is enabled
 - Modified EngineExpressionEvaluator to use the new feature flag to execute scripts
 - Modified PlanNodeExecutionStrategy to use the new feature flag to resolve step parameters
 - Added unit tests for script expressions and feature flag support
---
Cosine Similarity: 0.8188196833281048

Original Description:-

Remove schema cache per accountTesting: Locally generated schema before and after change are equal using diff checker
---
Generated Description:-

- Removed schema cache per account
 - Changed the schema cache key to only include module type
 - Removed some commented out code

Cosine Similarity: 0.8513747731203366

Original Description:-

Converting WaitNotifyEngine value from override error to override nests as it was wrongly set and causing increase in error logsTesting: LocalImpact: Related to logs where if same key is added to context it will override the previous value, which is expected. No other impact apart from logging context.

Generated Description:-

- Changed the value of AutoLogContext.OverrideBehavior from OVERRIDE_ERROR to OVERRIDE_NESTS in WaitNotifyEngine.java file.
 - This change was made to fix the issue of an increase in error logs caused by the wrongly set value.
 - The sendNotification method now uses the updated value of AutoLogContext.OverrideBehavior.
 - The commit message was repeated twice.

Cosine Similarity: 0.8891276314001796

Original Description:-

CCM is preparing to launch a new feature called perspective preferences, which is outlined in the default account settings. As part of this implementation, whenever there is an update in the perspective preferences, we require an event to trigger certain operations on the BE side. To accomplish this, we are publishing redis events on settings update.Testing:Event Messages:Successfully updated default settings across various categories and able to publish individual events for each category.Updated default settings that include a groupIdentifier. Able to publish event based on the <category, groupIdentifier> combination.Updated default settings without a groupIdentifier. Handled null groupIdentifier in the event's metadata and able to successfully published the event.

Generated Description:-

* Added functionality to publish a Redis event on settings update.
 * Fixed a metadata issue.
 * Added logs for debugging.
 * Constructed a new HashMap object instead of using an emptyMap.

* Removed redundant logs and added a try-catch block.
* Handled null groupIdentifier in Redis event request.
* Added a JUnit test case for Redis event on setting update.
* Moved the setting event proto under entity_crud

Cosine Similarity: 0.8290202882425363

Original Description:-

CCM has approximately 19 customers who have configured GCP connectors. Currently, we are not factoring in the GCP discount when calculating the total cost. Enabling the GCP discount default setting to true would result in a notable impact on the GCP total cost, potentially affecting budgets, anomalies, and other factors.As a result, we have decided to set the default value to false, allowing customers to enable it themselves.
---
^COriginal Description:-

Recently, we have added a NOT_IN check on COMMON cloud provider field to get the cloud provider values and add into the datasources.We have to migrate all the perspectives to reflect these changes.

Generated Description:-

- Added a new import statement for ViewIdCondition entity
- Added a new method getDataSourcesFromCloudProviderField() to CEViewService interface
- Modified CEViewServiceImpl class to implement the new method
- Added a new ViewIdOperator EQUALS to CEViewServiceImpl class
- Modified CEViewCloudProviderDataSourcesMigration class to use the new method from CEViewService interface
- Modified CEViewCloudProviderDataSourcesMigration class to use the accountId parameter in the new method
- Modified CEViewCloudProviderDataSourcesMigration class to remove the old method getDataSourcesFromCloudProviderField()
- Added a new migration step to modify CEView data sources based on cloud provider field values
---
Cosine Similarity: 0.8249025330970031

Original Description:-

T-Mobile utilizes a third-party tool for configuring their nodepools. As a result, the nodepool name is assigned a distinct label that differs from the default AKS nodepool-name label.In order to generate nodepool recommendations for T-Mobile, we need to extract the nodegroup label that corresponds to the nodepool-name.

Generated Description:-

- Added support for a new label \'nodegroup\' to be used for T-Mobile\'s recommendations.
 - Updated the \'populateNodePoolNameFromLabel\' method to check for the new label in addition to the existing label.
 - Added a new constant \'GENERAL_NODE_GROUP_KEY\' to represent the new label.
 - No other changes were made.

Cosine Similarity: 0.8540984609176197

Original Description:-

Added a check to handle flattened labels if they are not present in the cluster tables.

Generated Description:-

- Added a boolean variable \'isClusterTable\' to the class \'ViewLabelsFlattened\'
 - Modified the method \'getCondition\' in \'ViewsQueryBuilder\' to check if \'isClusterTable\' is true before checking if \'flattenedLabelsTableColumns\' contains \'labelSubQuery\'
 - Modified the method \'getFlattenedLabelsTableColumns\' in \'ViewsQueryBuilder\' to take \'tableIdentifier\' instead of \'cloudProviderTableName\'
 - Modified the method \'getViewLabelsFlattened\' in \'ViewsQueryBuilder\' to set \'isClusterTable\' based on \'tableIdentifier\'
 - Commit message: \'Handled flattened labels for cluster tables only\'

Cosine Similarity: 0.8421761143479376

Original Description:-

Implemented a check to exclude nodepools that have more than one instance family from generating recommendations. This is necessary as the current calculations for such nodepools result in incorrect recommendations.
Additionally, for customers whose clusters are already managed by spot.io, it is unnecessary to generate recommendations for them.

Generated Description:-

- Added new constants for subfield format, not equal and and
 - Modified filter conditions to use subfield format for instance data and instance metadata
 - Added new method to fetch distinct instance families based on filter conditions
 - Modified index to include instance state and instance family for instance data
 - Added check to skip node pools with multiple instance families in K8sNodeRecommendationTasklet

- Updated index to fetch instance data for node pool recommendations
---
Cosine Similarity: 0.8020837717138739

Original Description:-

If an invalid perspective ID is provided, our code used to fail with a null pointer exception. Now, we first check if the perspective exists or not. If it does not exist, we send a bad request error.

Generated Description:-

- Added a check to ensure that the requested perspective exists before performing any operation on it.
- Added an exception to be thrown if the requested perspective does not exist.
- Improved error handling by adding an invalid request check.

Cosine Similarity: 0.8774210565367404

Original Description:-

Fixed Labels Flattening binding issue.

Generated Description:-

- Added LabelFlattenedService to BatchProcessingModule and CENextGenModule
- Removed LabelFlattenedService binding from DataResponseService in BatchProcessingModule and CENextGenModule
---
Cosine Similarity: 0.7569423092366517

Original Description:-

Generating cost category query for the selected cost buckets only except in case of Unattributed.

Generated Description:-

- Added import statements for new entity classes and collections
- Added new method parameters for filter and modified existing method calls to include the new parameters
- Modified the logic for generating SQL case statements for cost categories based on the filter operator and selected cost targets
- Fixed issues with cost category operator checks and inclusion of cost buckets for IN operator
- Removed unnecessary parentheses around expressions

- Made other minor changes and improvements

Cosine Similarity: 0.6877238979434126

Original Description:-

Fixed null pointer exception in case of creating nested cost category
---
Generated Description:-

- Fixed a null pointer exception that occurred when creating a cost category
- Improved validation of the same business mapping in cost bucket and view condition
- Ensured that the same cost category cannot be added to a cost bucket
- Improved the overall stability and reliability of the product
---
Cosine Similarity: 0.8546275573647435

Original Description:-

Enabled labels flattening for all cloud providers.Running migration script to update dataSources based on COMMON Cloud provider field.

Generated Description:-

* The changes made in the code enable labels flattening for all cloud providers.
* The method `shouldUseFlattenedLabelsColumn` has been removed.
* The parameter `shouldUseFlattenedLabels` has been removed from several methods.
* The `getDataSourcesFromViewMetadataFilter` method has also been removed.\\n\\nThese changes affect the functionality of the product by simplifying the code and removing unnecessary parameters and methods
---
Cosine Similarity: 0.8146401012360247

Original Description:-

Added key-value based search support on Labels for executions.Key is searched in set of label keys from executions and same applies to value as well.Tested searching key:value label pairs in executions
---
Generated Description:-

- Added a new annotation to NGLabel class to generate inner class for key constants.
- Added a new hidden field \'labels\' to FilterPropertiesDTO class.
- Added new key constants for labels in PipelineExecutionSummaryEntity class.

- Added a new field \'pipelineLabels\' to PipelineExecutionFilterPropertiesDTO class.
- Added new methods to add pipeline labels criteria in PMSExecutionServiceImpl class.

Cosine Similarity: 0.7765214840569904

Original Description:-

Test cases:Adding labels support in V1 pipeline yamlIntegration of labels in execution flowsTags should be resolved only for V0 yaml and labels should be resolved only for V1 yamlTested usual pipeline v0 execution with tags, working fine with all tags visible in execution historyTested V1 execution with labels, working fineTest runtime labels for V1 executions, working fineTest runtime tags for V0 executions, working fine

---

Generated Description:-

- Added NG label and its helper classes
- Added a YAML helper class for basic pipeline
- Integrated NG label into execution summary
- Added labels to execution summary
- Resolved labels at the end of pipeline execution
- Added labels in pipeline YAML
- Refactored files
- Refactored code to use simplified pipeline YAML instead of basic pipeline
- Combined labels and tags resolver into the same orchestration end handler
- Removed orchestration end labels resolver registration
- Added a new field version in pipeline summary entity
- Added a new labels resolve handler at the end of orchestration
- Added null check
- Added UT

Cosine Similarity: 0.8322778174196964

Original Description:-

Added support for transient branch which overrides default branch selection logic if present.If template ref in a yaml has any git branch present, it should be set as transient branch while fetching the template which essentially overrides everything and template is fetched from given branch only.The value should be reset as soon as operation is over as other templates may or may not abide by default logic, thus this transient branch shouldn't affect other flows.

Generated Description:-

- Added support for a new transient branch
- Added methods to check if the transient branch is set and to set/reset the transient branch

- Added a new context guard for the transient branch
- Integrated the new context guard for the transient branch
- Added logs for fetching remote entity

Cosine Similarity: 0.788880024482311

Original Description:-

Added branch as a param in fetching unique identifier for template. This identifier is used as key in template cache map. With static linking in picture, it may happen that 2 templates can be linked in same yaml from different branches. Thus, while parsing the yaml, we may need to fetch 2 same templates but from different branches.Tested out applyTemplatesV2 flow and validate-template-inputs API, working fine with static branch linking
---
^COriginal Description:-

Added new field in UI contract for Template yaml required for static branch linking
---
Generated Description:-

- Added \\"gitBranch\\" field to the JSON schema for DeploymentStage, SecurityStage, IntegrationStage, ApprovalStage, and Pipelines.
- Updated TemplateLinkConfig class to include \\"gitBranch\\" field.
- Fixed unit tests.

Cosine Similarity: 0.7816624671828576

Original Description:-

Solution to : https://harness.atlassian.net/browse/CDS-68945
Revert fix done in : #47957Usage of static map in class being used in concurrent threads results into concurrent modification exception. Have converted static map into private map => Every thread has its own map.

Generated Description:-

- Converted the logContextMap from a static variable to a private variable to handle concurrent modification issues.
- Removed the ConcurrentHashMap and directly assigned the contexts variable to the private variable.
- Modified the setContextIfNotNull method to accept the logContextMap as a parameter instead of using the static variable.

Cosine Similarity: 0.8546642180746117

Original Description:-

Need to add support for creating projects with old git sync experience who are still using the old git sync. Earlier, we removed the FF and all its usages but now have to bring it back.

Generated Description:-

- Removed usages of feature flag USE_OLD_GIT_SYNC
 - Reverted a previous commit related to USE_OLD_GIT_SYNC
 - Removed feature flag USE_OLD_GIT_SYNC

Cosine Similarity: 0.8128319927927914

Original Description:-

Exposed API to update filepath and repo for remote templates
Tested following cases:Happy flow updating repo and filepath for given template id and versionUpdating a template that doesn't exist throws an exception
---
Generated Description:-

- The API has been updated to allow for the updating of file paths and repository names for RE templates.
 - Access control checks have been added to the update git details API.
 - The API request and response have been refactored, as well as the request parameters.
 - Minor fixes and a build number increase

Cosine Similarity: 0.822174985914597

Original Description:-

Handle empty get batch file requests in background cache: We need to ignore empty requests and not submit them for execution to thread poolCurrently, its difficult to trace what happened on SCM as there’s no identifier to find out SCM logs related to a specific task. So, need to add more logs to delegate get file task itself for debugging purpose.

Generated Description:-

- Added a check to not submit a background cache refresh task if the requests to cache are empty.
 - Added response time logs in delegate tasks for getFileContent, getRepoDetails, getFile, and getLatestCommitOnFile methods.

Cosine Similarity: 0.8292393960498944
