Here are the different types of nodes that can be found in a graph:

An "EventStream" node Identifies a stream of events to listen for and serves as the root task in an App. It accepts the following configuration properties:
An enumerated property called "inboundResource" which can only be set to one of "topics", "sources", "types" or "services".
A string property called "inboundResourceId" which contains the name of the inboundResource.

A "Client" node can be used to display information to the user.

A "Transformation" node Defines a procedure or mapping that modifies an event in the stream before being passed to downstream tasks.

A "Log Stream" node Logs all inbound events.

A "Publish To Service" node Sends an event to an Inbound Event Type of a Service. It takes a configuration property called "service" which gives the name of the target Service. It takes a configuration property called "eventTypeName" which gives the name of the input event on the target Service.

An "Analytics" node analyzes the incoming data and computes one or many of the following statistical operations: average, mean, median, count, min, max, standard deviation, geometric mean, variance, skewness, and kurtosis. Configuration properties:
inboundProperties – The list of property names (e.g. “speed”, “temperature”, etc.) on the inbound events to compute statistics for. Each property must be numeric.
reservoirType – The type of reservoir used to store data points.
windowLength – How long events should stay in the window. For time based reservoirs this is an interval string (e.g.: “10 minutes”). For count based reservoirs this should be the number of events.
operations – Which statistics operations should be applied to the data. It must be one of "mean", "median", "count", "min", "max", "standard deviation", "geometric mean", "variance", "skewness", and "kurtosis".

A "Save To Type" node Saves inbound data to a type.

A "Filter" node Defines a condition that restricts which events flow on to downstream tasks. It takes a string configuration property called "condition" which describes which events will be emitted.

A "Join" node Join pairs of events on two parallel streams.

A "Notify" node Sends a notification to a Vantiq user using the Vantiq Mobile App.

A "Track" node Begins tracking one or more Vantiq users until they arrive at a specified location or the collaboration ends.

An "Accumulate State" node aggregates state over a series of events by calling a procedure that takes the existing state object and the new event and produces the new state object.

An "Assign" node assigns the inbound event to the specified entity or collaborator role.

A "Build And Predict Path" node assembles the path from the tracked motion of objects with locations. Predict the location of the object when it is removed from tracking.

A "Cached Enrich" node enriches and caches the results for future events. The cached association is refreshed at a configurable interval.

A "Chat" node initiates a Chatroom that uses a Chatbot to process chat message and react appropriately.

A "Compute Statistics" node computes the min, max, mean, median, standard deviation and count for a single property of events that pass through the task.

A "Convert Coordinates" node converts coordinates (typically from image analysis) to another coordinate space.

A "Collaborations Status" node updates the status of the current collaboration to ‘failed’ or ‘completed’.

A "DBScan" node applies a density based clustering algorithm to determine if there are one or many clusters.

A "Delay" node Delay the emission of the inbound event for a configurable length of time.

A "Dwell" node Detect a sequence of events with a consistent state across over a period of time.

A "Enrich" node Augment events with associated data from a persistent Type.

A "Establish Collaboration" node Ensures that only one collaboration is created for the entity role specified by the roleName.

A "Get Collaboration" node Return the current collaboration.

A "Interpret" node Get the Natural Language Interpretation of an Utterance from an external interpreter like LUIS.

A "K-Means Cluster" node Applies a K-Means Clustering algorithm to find the specified number of clusters in the data.

A "Limit" node Limits the flow of events through an App to a configurable maximum throughput.

A "Linear Regression" node Applies linear regression to the data and generates a prediction procedure to predict future data points.

A "Loop While" node loop through the tasks that are part of the loop body while the specified condition is true.

A "Merge" node Joins multiple independent streams into a single stream.

A "Missing" node Detects the absence of an event for a time interval.

A "Polynomial Fitter" node Applies a polynomial fit algorithm to the data and generates a prediction, derivative, and integration procedures.

A "Predict Paths by Age" node Predicts positions for paths not recently updated.

A "Process Intent" node Processes and executes the custom or system intent that was returned by the Interpret pattern.

A "Publish To Source" node Sends inbound data out to a source. It takes a configuration property called "source" which gives the name of the target Source.

A "Publish To Topic" node Sends inbound data out to a topic.

A "Procedure" node Executes a procedure and emit the results. It takes a configuration property called "procedure" to specify the name of the Procedure to be invoked.

A "Rate" node Emits the frequency of inbound events at a regular interval.

A "Recommend" node Generates a list of recommendations based on the incoming event and the match directives.

A "Record Event" node Records an instance of an Event Type in the Event Ledger.

A "Run TensorFlow Model on Document" node Uses a TensorFlow model to analyze a Vantiq Document (typically an image).

A "Run TensorFlow Model on Image" node Uses a TensorFlow model to analyze a Vantiq Image.

A "Run TensorFlow Model on Tensors" node Uses a TensorFlow model to analyze data.

A "Sample" node Samples events from the inbound stream to produce fewer events to the downstream tasks.

A "Smooth" node Smooths a bursty workload out into a more steady workload by delaying the delivery of some events.

A "Split By Group" node Splits the upstream events into sub-streams for the downstream tasks by a group key.

A "Start Collaboration (deprecated)" node creates a situation which can trigger a collaboration.

A "Threshold" node Detects the crossing of a threshold across sequential events.

A "Time Difference" node Measures the time between two consecutive events.

A "Track Motion" node Assembles the movement of objects with locations (typically YOLO image analysis output).

A "Unwind" node Splits an array property on an event into multiple separate events that can be processed independently.

A "VAIL" node Runs an arbitrary block of VAIL code for every event.

A "VisionScript" node enables the user to build a Vision Script in order to manipulate images.

A "Window" node buffers inbound events to emit batches of events. Overlapping and non-overlapping windows are supported.

A "YOLO From Documents" node uses a YOLO model for image (stored in a document) analysis.

A "YOLO From Images" node uses a YOLO model for image analysis.

Assume that these existing JavaScript functions can be used describe the structure of a graph.

```
/**
   * Create a node
   * @param type - The type of the node
   * @param name - The name of the node
   * @param x - The x position of the node
   * @param y - The y position of the node
   */
createNode(type,name,x,y);

/**
   * Connect two nodes
   * @param from - The name of the source node
   * @param to - The name of the target node
   */
connectNode(from,to);

/**
   * Set a configuration property on a  node
   * @param name - The name of the node
   * @param property - The name of the property
   * @param value - The value of the property
   */
configureNode(name,property,value);
```

Format your answer in Github markdown. Assume the graph is laid out from top to bottom.

Write code using the above functions that will build a graph for the following application: