Arkose Ensemble

Document status	DRAFT UNDER REVIEW APPROVED
CS Team	TechOps
Document owner	@ owner
Document reviewer	@ reviewer
Document approver	@ approver
Collaborators	@ collaborator
Created	/date
Last update	May 24, 2022
Document Sign-off	☐ @ approver /date

High Level Summary

This document describes Arkose Ensemble, a framework that will form the substrate for the development of advanced modular components for detecting telltales and managing traffic pressure. This will enable multiple components to more seemlessly integrate across the platform, and will allow for greater transparency in data transfers between components.

We believe that the efforts at automation to-date have suffered from two major problems:

1. The have largely taken a standalone and monolithic approach to solving the problem. We believe that the problem is too complex and too dynamic to be solved using such an approach.
2. They have been incorrectly scoped, by trying to solve the problem using limited information, requiring the solution solve too many problems, or not being able to properly integrate with existing technologies.

Arkose Ensemble is a centralized event and job management tracking system, designed to allow multiple independent systems to pass information and work to be done to other systems. This will ideally allow for a centralized method of integration, and requires less

entanglement of different systems. Additionally, all interactions with the system are designed with a write once, read many paradigm. This means more systems can be added without affecting existing flows.

Ensemble aims to drastically reduce the complexity of modularizing components, by creating a standard data layer for components to interact. By providing a centralized store of this information, we can have one location to view the higher level performance of all components. This should make the behaviour of the system more transparent and reliable.

Ensemble manages information by allowing components to send `Events` through. These events can be anything from simple notifications to detailed analysis results. By default, it is assumed that any event may be read by any component, and any event may lead to further events. By tracking the parent events, we can build visualizations of the long term impacts of different components in a centralized place.

For instance, ensemble will allow us to see the entire lifecycle of a telltale, from alerts, robosnitch results, telltale creation, auditing, pressuring and deleting, regardless of which components execute each step.

Training Videos:

Introductory videos to Ensemble can be found here:

- https://drive.google.com/drive/folders/1DPEZG0VTMGzwMunHEfpp8IkZNeYbM6WB?usp=sharing Connect your Google account

Terminology

- **Event Type**: A event type is a set of events that all conform to the information type. Each Event type has a standardized schema which dictates the necessary fields that all events MUST conform to be added to the database. Examples might include:
  - `alert_results`: the output of a series of alerts,
  - `robosnitch_analysis_result:` The results of an analysis engine,
  - `add_telltale` might contain a series of jobs to add telltales.
- **Event**: An event can represent anything kind of state change across the Arkose Labs platform. By default, any component can read events and take actions. The fields included on each event include:
  - **Event Type** - Which type of thing this event represents (e.g. an alert output, a telltale creation event, etc.)
  - **Source** - Which component created this event. e.g. if the Event was created by Robosnitch, then “Robosnitch”. If this was done via manual intervention, then include the user who conducted the action
  - **Entry** - A title for the action that needs to be taken. A human readable summary of the event contents
  - **Data** - A JSON blob containing all the necessary data. This Data blob is validated against the event type Schema
  - **TelltaleName -** If the action is linked to a single telltale, then include it here. Makes searching for telltale actions much easier
  - **PublicKey -** If the action is linked to a single Public Key, then include it here. Makes searching for specific changes on a key much easier.
  - **CreatedAt** - A timestamp of when the event was added to the database.
- **Component**: A component represents any producer or consumer of Arkose Ensemble. Anything that writes or reads to Ensemble must be a registered component. A component does not represent an instance, but rather a system. For example, Robosnitch might have 3 EC2 instances, but it will only have one component, the generic “Robosnitch”. When consuming events, a component can either read generically from any event_type, or consume using an Exactly Once consumption pattern.
- **EventStatus:** Whenever a component accesses an event using the Exactly Once consumption pattern, Arkose Ensemble will store an EventStatus to show that the component has accepted the event. It is the responsibility of the component to update the status of the event to signal that the event was successfully processed. The valid states for EventStatus are:
  - `Started`
  - `In Progress`
  - `Succeeded`
  - `Failed`
  - `Cancelled`
  - `Acknowledged`