Software Design and Architecture

Table of Contents

Introduction

The project will read log data and send it to a centralized storage-collection server.

The log collection service should take an event log, process each event line, and send the data to a central server. The release goals aim to satisfy the confidentiality, integrity, and availability of all data flowing through service pipeline.

back to top

Architectural Goals and Principles

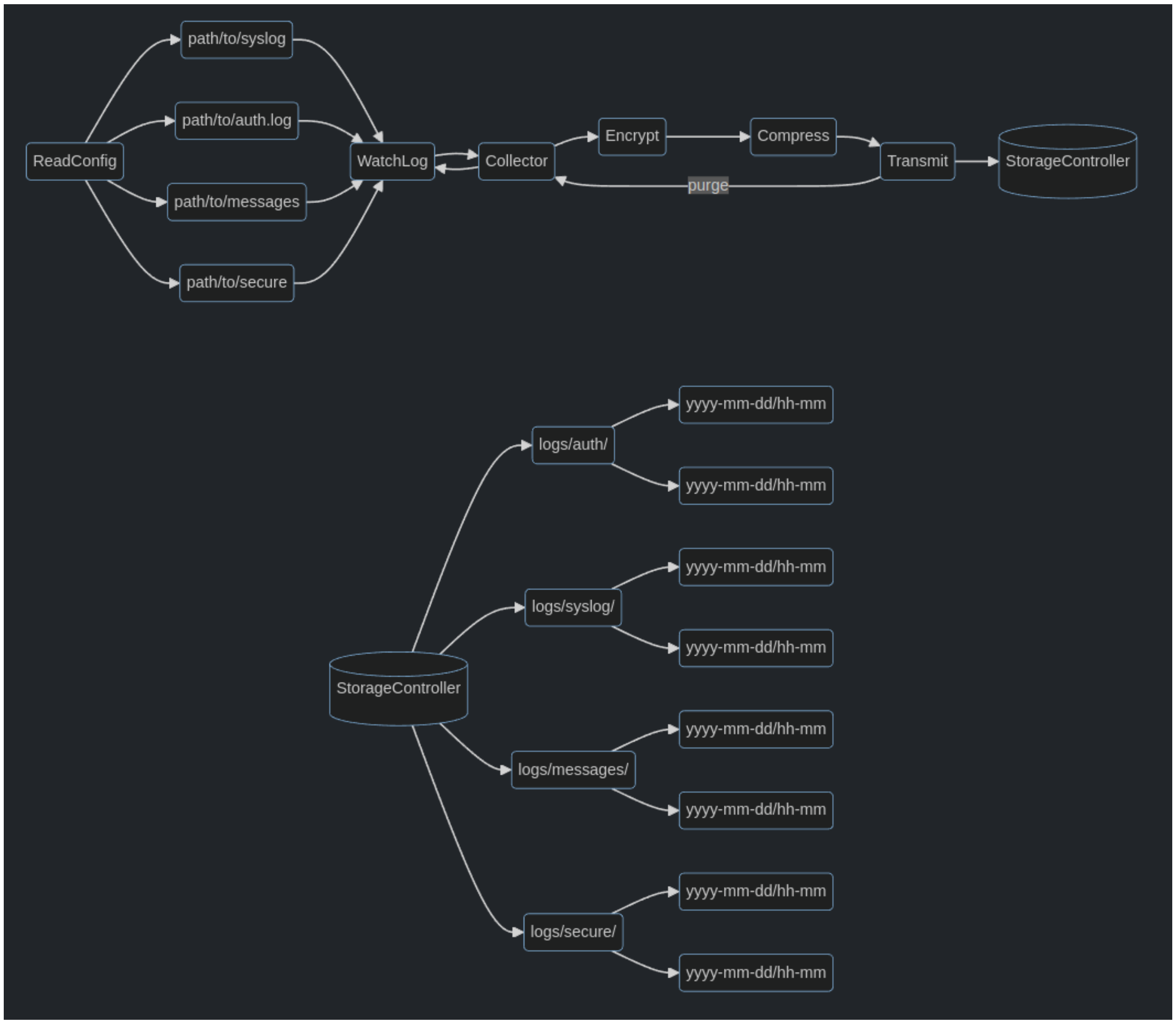
The main architectural goals and principles are security, scalability, abstraction, and modularity. Architectural principles include maintaining proper software security practices, designing modular systems, and performing comprehensive tests to prove functionality.

back to top

System Overview

An open-source tool that is available to users. The application will function on various operating systems, which will be tested through deployment on VMS. The centralized server is expected to receive all the event logs specified by user.

Starting at the Data Source, event lines are sent to the watchlog module, where they are serialized and staged to be sent across the network. The communication will be managed over TCP sockets, allowing for low-latency monitoring and data feeding.



back to top

Architectural Pattern

We will adapt the client-server, producer-consumer, event-sourcing pattens.

Client Server Pattern

The client-server model, where the clients on endpoints send event logs to a centralized server for processing and storage.

Producer Consumer Pattern

Any component that needs to handle the input -> output of data asyncronously will benefit from using threads to break up tasks for efficiency of time and/or memory.

Event Sourcing Pattern

For flexible, real-time data handling, this pattern can be beneficial to stream and store log data continuously.

back to top

Component Descriptions

Client Component

The client will consist of a cli (default). Once installed, the user will have full visibility of outgoing log data. The user will be able to interact with each log source, download log data from the central server, and add/remove log sources.

The user will utilize a configuration file to specify information such as: - a path (or paths) to log sources - a destination IP/port - private key information.

Client

Server Component

Considering that the central server will be a destination for many users, it is important that this component of the project can scale in the future. For now, during beta testing, the server should be able to create isolated instances for each user that is generating incoming logs.

Each instance will be tied to a user using a unique key generated an existing, and validated, user. Each instance will store the incoming logs into a database instance and provide an extension to ElasticSearch should the user want that additional functionality.

Server

Documentation Component

It does not have to be complicated but it must be organized. Projects die when the documentation is either unavailable or lacks enough information to help users (and developers) get started.

At a minimum, the user documentation should include the following: 1. Getting Started: - how to install software - how to run the client and server - configuration guidelines

2. Service Information:
    - log layout and directory structure

back to top

Data Management

Client Data

Data types defined here: https://github.com/endepointe/watchlog/blob/main/README.md#data-types

Incoming log data will be defined using the configuration file located on the client.

```
{
    "defaults": {
        "compression_level": 5,
        "key": "/path/to/key",
        "tx_buffer": "1KB"
    },
    "logs" : [
        {
            "source": {
                "name": "test1.log",
                "path": "./test1.log"
            },
            "destination": {
                "address": "192.168.1.1",
                "port": 5052
            },
            "compression_level": 5,
            "key": "/path/to/key",
            "tx_buffer": "1KB"
        },
        {
            "source": {
                "name": "test2.log",
                "path": "./test2.log"
            },
            "destination": {
                "address": "192.168.1.1",
                "port": 5052
            },
            "compression_level": 5,
            "key": "/path/to/key",
            "tx_buffer": "1KB"
        }
    ]
}
```

Server Data

Log data will be stored in the watchlog directory, under logs/.

back to top

Considerations

Security

Focus on: Confidentiality, Integrity, Availability

Private key will be generated using openssl. The client's private key should not be stored on the server and a key-rotation playbook should be developed.

Performance

Long-term goal is that the server is able to scale as the number of users grows.

While system dependent, the client should be able to handle around 5K events per second. Read the following for Event-Per-Second:

Event Per Second

Maintenance and Support

Members of the organization are able to help maintain the project, with a handful of CODEOWNERS that help guide the updates as the project grows.

Anyone using the product will be able to submit issues that will be used to make improvements and take suggestiongs under consideration.

back to top

Testing Strategy

Tests will:

```
1. Analyze the requirements
2. Plan appropriate tests (security, performance, regression, user)
3. Execute those tests
4. Use the results of the test for further discussion
```

back to top

Glossary

Terminology	**Definition**
User Interface	User interactions are managed by a front-end implementation.
Backend Server	Processes requests, business logic, and interfaces with the database.
Database	Stores and manages data within a database for event logs in AWS.
Data Producer	Grabs a raw line from the event log using a thread
Data Consumer	Receives event from thread
Collector	Formatted and Serialized data is queued for Sender
Sender	Depending on central server status, the data is sent to its location
Listeners	Listen for status
Central Server	The destination of formatted event logs
Offline Storage	Data reservoir for redundancy/recovery if needed.
Endpoint	Device of a system that sends or receives data
CRUD	Create, Read, Update and Delete
SSL TLS	Secure Sockets Layer or Transport Layer Security are protocols for encrypting data transmitted over the web. Ensures secure communication between a client and server
Event Logs	Rercods events or activities generated by a system or network
Centralized Server	Single server to collect and manage services from multiple endpoints

back to top