

Alpaca

- [Introduction](#)
 - [Functional requirements](#)
 - [Design Principles](#)
 - [Basic pipeline architecture](#)
 - [Base components](#)
- [Download](#)
 - [Vector](#)
 - [Apache Kafka](#)
 - [Opensearch](#)
 - [Installer](#)
- [Installation](#)
- [Configuration](#)
- [Monitoring](#)
- [Integration](#)

Introduction

Functional requirements

The following requirements should be fulfilled:

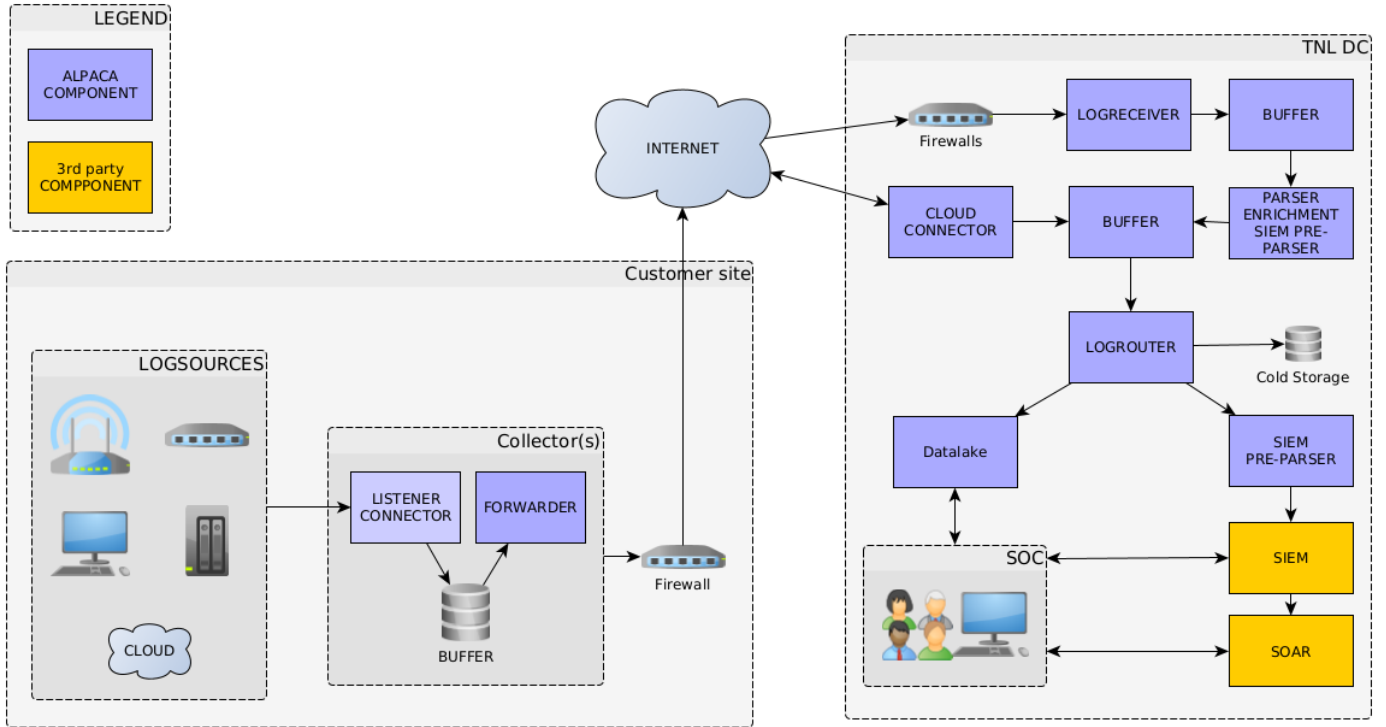
- Fully featured logmanagement-system
 - receive logs in ANY logformat
 - Converting logformats to support SIEM requirements
 - filtering unwanted logs
 - Protecting pipelines from overflowing
- Should fully integrate with Qradar and other SIEMs
- The following functionality should be available
 - buffering (in case of congestion/network outage/component failures)
 - filtering (should be possible anywhere in the pipeline)
 - logs should be searchable in a database-like datalake
 - logs should be stored to cold storage
 - encryption of data in transit AND data at rest should be supported
 - high availability, system should be able to fully recover from any type of intermittent failure
 - Redundancy: components should be replaceable without service-degradation.
 - Solution should be platform-independent (OS/Hardware agnostic)
 - Components must be supported on latest OS/patchlevels.
 - Components should be in active development/support.
 - platform should support log-transformation to meet Qradar log-standards
 - Each part of the data-pipeline should be auditable/monitorable.
 - Multi tenancy
 - Proven technology

Design Principles

To limit the possibilities we also decided on some principles the solution should follow:

- No java unless thoroughly tested
- No docker/containers
- No fancy-schmancy python-code.
- Run on Linux
- both X86/ARM support for key components
- Deployable using industry standard deployers (ansible/chef/puppet)

Basic pipeline architecture



During each (critical) step data will be written to storage (which will be HA/redundant) to ensure no data will be lost when a critical failure occurs.

The amount of data in memory will be limited as much as possible

Base components

During extensive research and experiences from the past, the following software-stacks have been selected as the preferred components to build the new solution.

- Vector (<https://vector.dev>) As the core log-management core.
- Kafka as an high-speed buffering solution
- OpenSearch for core data-lake, metrics and dashboarding and datalake alternative
- Ansible (deployer for configuration and setup)

Other packages will be selected depending on need or to handle specific use-cases.

Download

Download

Vector

At the core of ALPACA vector is doing most of the heavy lifting.

It can be downloaded at: www.vector.dev

Download

Apache Kafka

Between each major operation data is buffered.

Alpaca uses apache-kafka as a proven/reliable/scalable solution.

It can be downloaded here: <https://kafka.apache.org/>

Download

Opensearch

opensearch is used for datalake and dashboarding.

It can be downloaded here: <https://opensearch.org/>

Download

Installer

The installer is a set of Ansible playbooks

A complete tar-file can be downloaded here and can be placed in a pre-existing ansible-environment.

An installer for an "All-in-one" server (single-node) can be found here: <TODO>

Installation

Configuration

Monitoring

Integration