Überblick der Softwarearchitektur von AV1

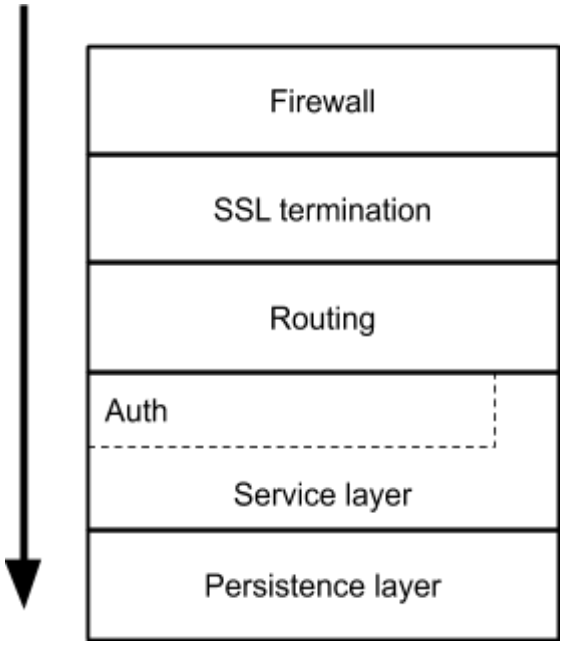
Apps	Backend	Gerät
		

Apps

- **React Native**: Die iOS and Android Apps sind in React Native geschrieben.
- **Redux**: Funktion Manager für Apps
- **socket.io**: Socket-Ereignisse zum Backend.
- **Superagent**: HTTP-Anfragen zu Backend.
- **WebRTC**: Video Streaming
- **Front Chat**: Support Chat
- **Fabric**: Statistiken für Ereignisse und App-Abstürze (anonym)
- **Sentry**: Fehlermeldungen und Crash-Daten (anonym)
- **i18next + Crowdin**: Lokalisierung

Backend

Der externe Service wird von uns am Port 443 zu Verfügung gestellt, und der ganze Datenfluss von und zum Backend ist verschlüsselt. Unser Backend ist auf AWS hergestellt worden und verwendet AWS für Folgendes: Lastausgleich, Datenbanken, interne DNS-Verwaltung, Instanz Skalierung und dergleichen.



Alle eingehenden Befehle werden auf die gleiche Weise bearbeitet, wobei sie, durch die SSL-Beendigung, von der Firewall akzeptiert werden müssen. Anschließend werden die Router zum richtigen Service aufgefordert. Der Service führt die Authentifizierung und Autorisierung aus, bevor die Anfrage selbst verarbeitet wird.

Im Backend nutzen wir eine Microservice-Architektur, die aus zwanzig Dienste besteht. Einige Beispiele für diese Dienste sind

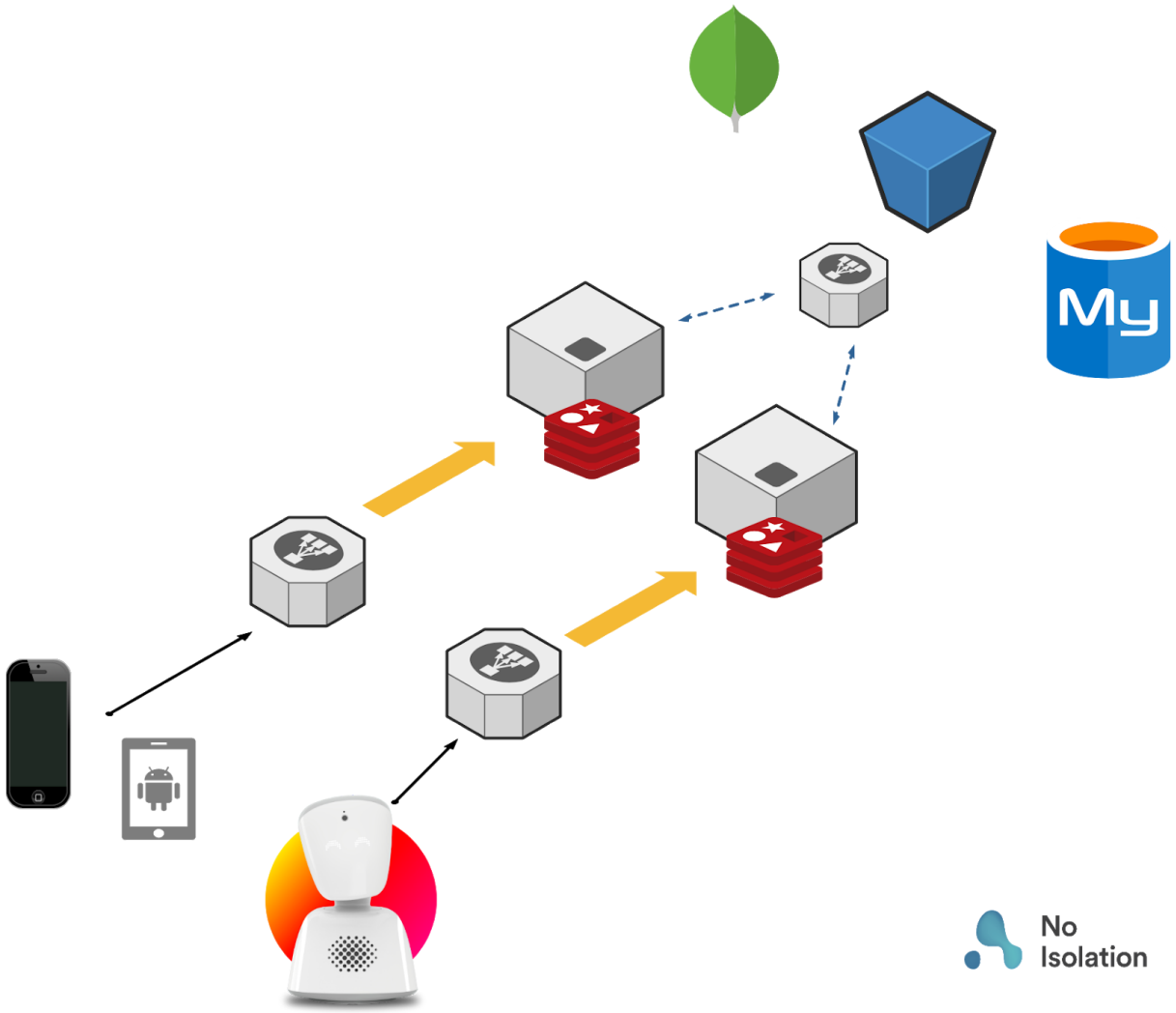
- Lesen und schreiben von AV1 Log-Daten
- Versenden des ersten E-Mails an Neukunden

Diese Dienste werden auf eine Reihe von EC2-Instanzen ("Server") verteilt, die je nach Service gruppiert sind, zu denen das Produkt gehört: AV1- oder allgemeine Dienste. Auf jeder dieser Instanzen wird Linux mit dem neuesten Nginx ausgeführt, um die Weiterleitung des Datenverkehrs an den richtigen Dienst, sowie die Drosselung von Anforderungen bei Bedarf sicherzustellen.

Jeder Dienst wird isoliert ausgeführt und hat keinen Zugriff auf andere Dienste oder deren Daten, anders als das, was jeder Service in seiner REST-API ausstellt. Die Kommunikation zwischen den Diensten erfolgt über HTTP mit REST-Endpunkten. Die Dienste sind hauptsächlich in node.js geschrieben, teils wird zusätzlich Golang verwendet.

Instanzen haben ihre eigenen Firewall-Regeln und lassen nur den erforderlichen eingehenden Datenverkehr zu. Nur eine kleine Teilmenge der Dienste wird extern verfügbar, und für diese Dienste ist ein verschlüsselter Datenverkehr erforderlich.

Die Infrastruktur, einschließlich Firewall-Regeln, Verkehrsrouting, Datenbanken und dergleichen ist in Terraform definiert («Infrastruktur als Code»), sodass die Infrastruktur in Git eingecheckt und selbstdokumentierend ist.



Werkzeug

Wir verwenden Gitlab als Git-Reporter und Tool für Code-Reviews, Merge / Pull-Befehle, sowie die Bereitstellung und Veröffentlichung von Softwares im Backend. Der Service wird auf einer EC2-Instanz unter unserem eigenen Konto ausgeführt, deshalb bleiben wir von Gitlab Ausfällen verschont. Die Fehler werden an sentry.io gemeldet, und wir erhalten Statistiken und Graphen, die die Infrastruktur darstellen, die sowohl von AWS Cloudwatch, als auch von newrelic.com verwendet werden.

Technologie

- Amazon Web Services
- Linux
- Nginx
- Node.js
- Golang

- MySQL
- MongoDB

Gerät / Unit

Die Units in den Produkten der zweiten Generation befinden sich auf einer sehr ähnlichen Hardware-Plattform mit dem selben System auf einem Modul (SoM; Qualcomm SD410c) und dem selben hochmodernen WIFI / Bluetooth-Modul (2x2, alle Kanäle, 2.4) + 5 GHz) usw.

Zusätzlich zur Hardware-Plattform betreiben wir eine eigene Linux-Distribution, die wir "noisos" nennen. Dies gibt uns die volle Kontrolle über die Low-Level-Treiber, wenn wir die neuesten Technologien verwenden, und ermöglicht uns die Implementierung sehr sicherer Upgrades und Sicherheitsmaßnahmen, indem wir die gesamte Software steuern, die auf dem Produkt ausgeführt wird.

Das Betriebssystem allein ist nur Linux mit GNU-Software und daher öffentlich, aber die Software, auf dem AV, ist vollständig unser Eigentum. Dazu gehören das Betriebssystem, die Steuerungssoftware für Bluetooth und drahtlose Netzwerke, sowie andere Daemons und Tools. Diese sind hauptsächlich in der Programmiersprache Golang geschrieben und verwenden nur freie oder Open-Source-basierte Bibliotheken außerhalb der Standardbibliothek (BSD, MIT, Apache usw.).

Technologie

In den folgenden Abschnitten werden die verschiedenen wesentlichen Technologien beschrieben, auf denen unsere Produkte basieren, einschließlich Apps, Back-End und Geräten.

Metadata-API für App und AV1

AV1 versendet Metadaten zwischen allen Teilen der Infrastruktur. Ein Beispiel für AV1 sind Verfügbarkeit, Abdeckung, Batterie Ladezustand usw. Diese Informationen werden aktualisiert und wir zwischenspeichern die Daten im Backend, um die Informationen für die App und den Benutzer verfügbar zu machen.

Die Metadaten-APIs sind die wichtigsten infrastrukturen für unsere Produkte.

WebRTC / Video Streaming

Als Video / Audio-Streaming-Technologie setzt AV1 WebRTC ein. Hierbei handelt es sich um eine zulässige Technologie mit freier Software Lizenz (BSD-basierte Lizenz), die für die Verwendung in den Browsern Chrome und Firefox vorgesehen ist. Relevante Objekte und

Bereiche der Technologie werden ausgewählt und in der native Version des Video-Streamings verwendet. Auf diese Weise haben wir nacheinander Zugriff auf die neuesten Versionen und Verbesserungen, sobald diese veröffentlicht werden. Die Sicherheit hat oberste Priorität und wird von Googles Sicherheitsteam und einer großen Entwicklungsumfeld, zu der auch das Mozilla Firefox-Team gehört, garantiert.

Das WebRTC-Bit verwendet, wie oben beschrieben, unsere Metadaten-API, um eine signaling durchzuführen (Verbindungsanfragen / -antworten während des Kommunikationsaufbaus). Wenn die App und das Gerät genügend Informationen über die Internetverbindung der anderen Geräte haben, stellen sie eine direkte Verbindung her. Wenn Firewalls oder ähnliches die Verwendung einer direkten Verbindung (Peer-to-Peer) verhindern, wird ein Relay-Server eingesetzt (TURN-Dienst von Twilio https://www.twilio.com/stun-turn). Jede Verwendung von Relay-Servern ist sicher, da das System eine vollständige Ende-zu-Ende-Verschlüsselung bietet.

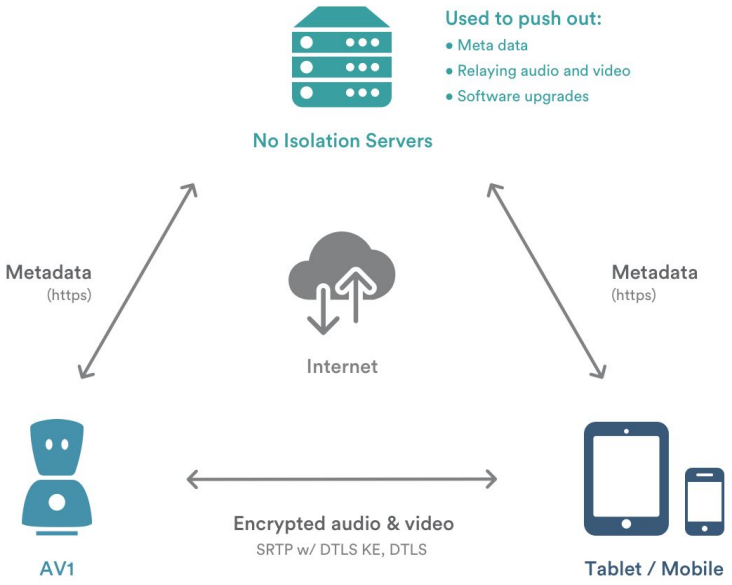


Abbildung: AV1 Informationsfluss

“noisos” und noisos Aktualisierungsvorgang

Unsere Linux-Distribution noisos wurde entwickelt, um Sicherheitslücken zu minimieren, und bietet uns ein kleines und einfaches Betriebssystem, das speziell für die Produkte entwickelt wurde. Diese enthält eine grundlegende Zusammenstellung von GNU-Tools und Linux-Kernel (basierend auf Qcoms Fork).

Auf dieser Basis haben wir eine Upgrade-Lösung entwickelt, die uns bessere Sicherheitsvorkehrungen bietet, als alle kommerziellen und Open-Source-Lösungen auf dem Markt. Die Lösung basiert auf einem „block-based Upgrade“ und unterstützt sowohl ein vollständiges System-Upgrade durch A / B-Partitionierung, als auch ein „Patch-Upgrade“ für kleinere Änderungen, bei denen die gleichen block-based Garantien bei Verwendung von „Overlays“ erzielt werden. Alle generierten Betriebssystemversionen sind vollständig durch

ihre SHA256-Prüfsumme gekennzeichnet, und das System erkennt automatisch, ob die Festplatte in unerwünschter Weise geändert wurde. Es ist daher möglich, Kundenprobleme zu reproduzieren, da der von ihnen ausgeführte Code bekannt ist und daher alle Einheiten derselben Version mit genau demselben Code ausgeführt werden. Alle während der Ausführung im System vorgenommenen Änderungen werden beim nächsten Start / Boot rückgängig gemacht. Upgrades erfolgen automatisch Nachts oder manuell auf Anfrage von Softwareentwicklern oder des Kundendienstes. Upgrades werden in der Regel auf Einheiten ausgeführt, welche die gleiche Software benutzen.

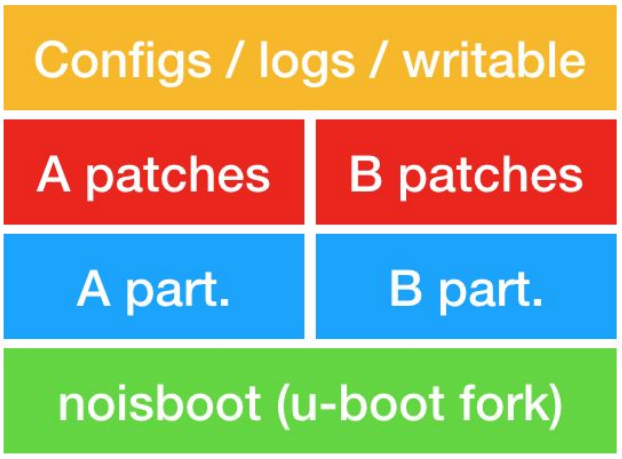


Abbildung: A/B Partitioning.

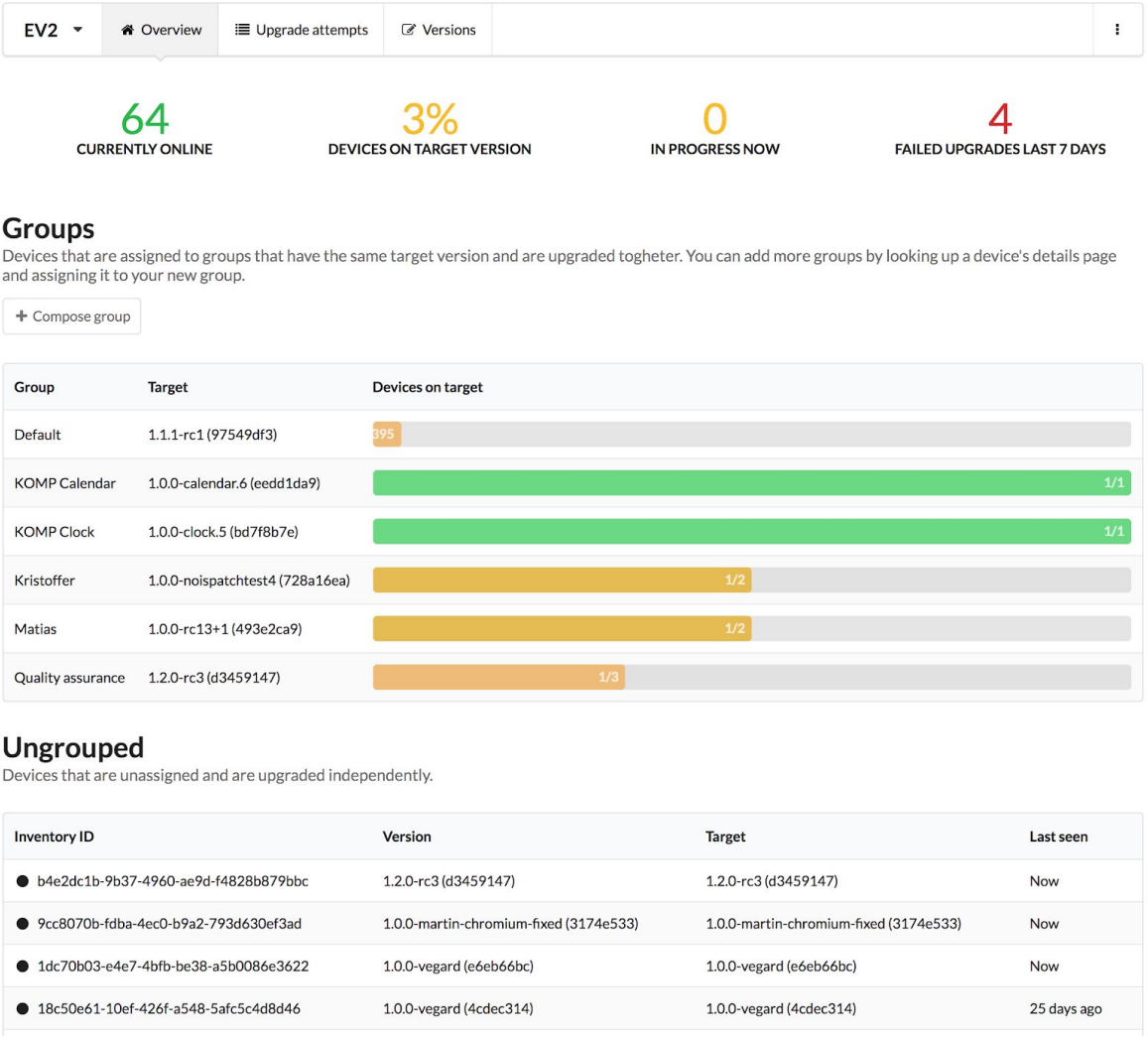


Figure: Upgrade server.

Remoter

Remoter wurde für die intern Fehlersuche bei Produkten, die sich in einer Kunden- / Benutzerumgebung befinden, entwickelt. Der Remoter ist die letzte Instanz, um Probleme zu beheben, die nicht von anderen Einheiten gelöst werden können.
Beispiele sind: Probleme mit dem Upgrade-System oder Log retrieval.

Aufgrund der möglichen sensiblen Zugriffsmöglichkeiten den diese Dienste bieten, wird zusätzlicher Sicherheit und Schutz implementiert. Der Service basiert auf TLS 1.2+ -verschlüsselten HTTPS-Verbindungen (Downgrade und solche Angriffe werden durch die HTTP2-Implementierung unmöglich gemacht). Der Backend-Server, der diesen Dienst bereitstellt, verfügt über eine zusätzliche Sicherheitsstufe und ist nur für wenige auserwählten Mitarbeitern zugänglich. Der gesamte Zugriff auf diesen Service ist auf Softwareentwickler beschränkt, die an den relevanten Problemen arbeiten.
Diejenigen, denen Zugriff gewährt wird, verwenden ein Command Line Tool, das auf ihrem eigenen Computer ausgeführt wird und sich mithilfe eines geheimen Benutzer-Schlüssels anhand eines „U2F-Tokens“ als zweitem Faktor authentifiziert. Alle Remoter-Zugriffe werden

in einem detaillierten „Überwachungsprotokoll“ protokolliert. Der Zugriff kann abgerufen werden und die Mitarbeiter können sofort ausfindig gemacht werden, wenn sie ihren U2F-Schlüssel verlieren, sodass ihr Konto deaktiviert werden kann. Die Authentifizierung ist jeweils nur 10 Minuten gültig und erfordert jedes Mal die physische Aktivierung des U2F-Schlüssels.

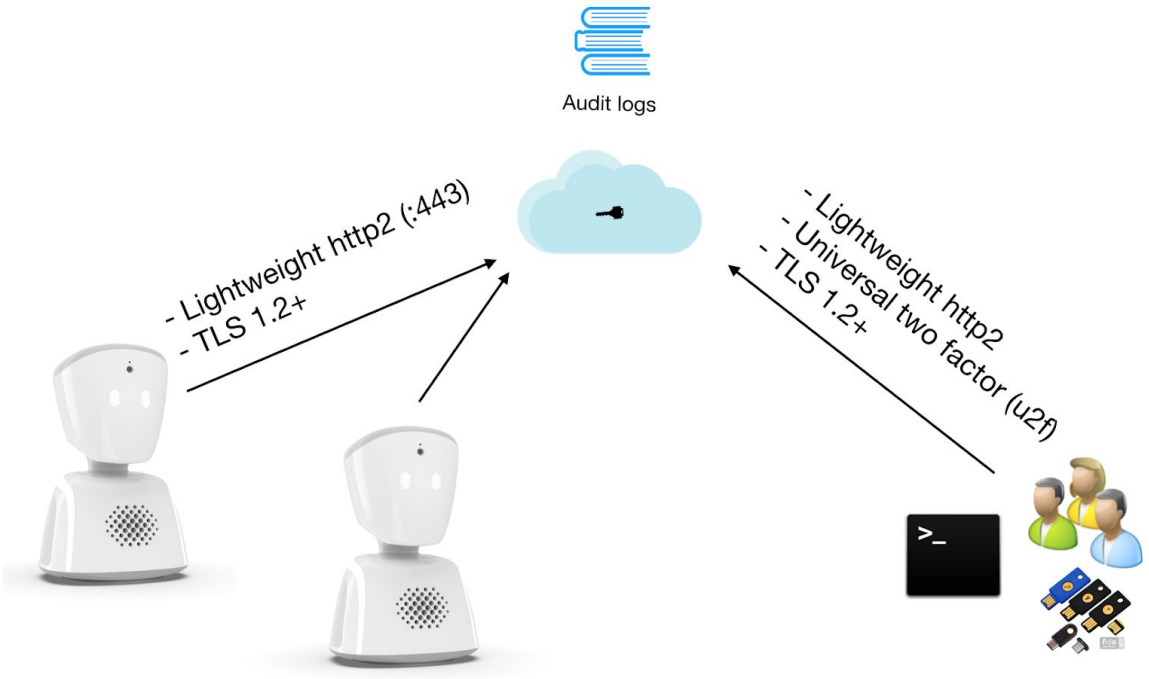


Abbildung: Übergreifende Softwarearchitektur des Remote-Services