REPLY STORM

Motor Control: Cloud Transformation for Embedded Development

Storm Reply Roma è l'azienda del gruppo Reply specializzata nella progettazione e implementazione di soluzioni e servizi cloud innovativi. Attraverso competenze consolidate e molti anni di esperienza nella creazione e gestione di servizi cloud Infrastructure as a Service (IaaS), Software as a Service (SaaS) e Platform as a Service (PaaS), Storm Reply Roma offre il proprio supporto ad importanti aziende in Europa ed in tutto il mondo nell'implementazione di sistemi ed applicazioni basati sul cloud.

Storm Reply Roma ha supportato STMicroelectronics nella modernizzazione della toolchain di generazione firmware per controllo motore basati su board con MCU STM32, affiancando alla soluzione desktop una piattaforma cloud-native e scalabile, integrata nativamente all'interno dell'ecosistema eDesignSuite. Grazie ad un'architettura ibrida Kubernetes + Serverless su AWS, il sistema consente la generazione automatica di progetti compilabili in pochi click, eliminando installazioni locali ed accelerando il time-to-market.

STMicroelectronics è una multinazionale italo-francese specializzata nella progettazione e produzione di dispositivi semiconduttori. Fondata nel 1987 attraverso la fusione di SGS Microelettronica (Italia) e Thomson Semiconducteurs (Francia), l'azienda si è affermata come uno dei principali attori nel settore dell'elettronica a livello globale. STMicroelectronics offre una vasta gamma di prodotti, tra cui microcontrollori, circuiti integrati, sensori e soluzioni per la gestione dell'energia. Questi dispositivi sono impiegati in diverse applicazioni, tra cui l'automotive, l'industriale, l'elettronica di consumo e l'Internet delle Cose (IoT).

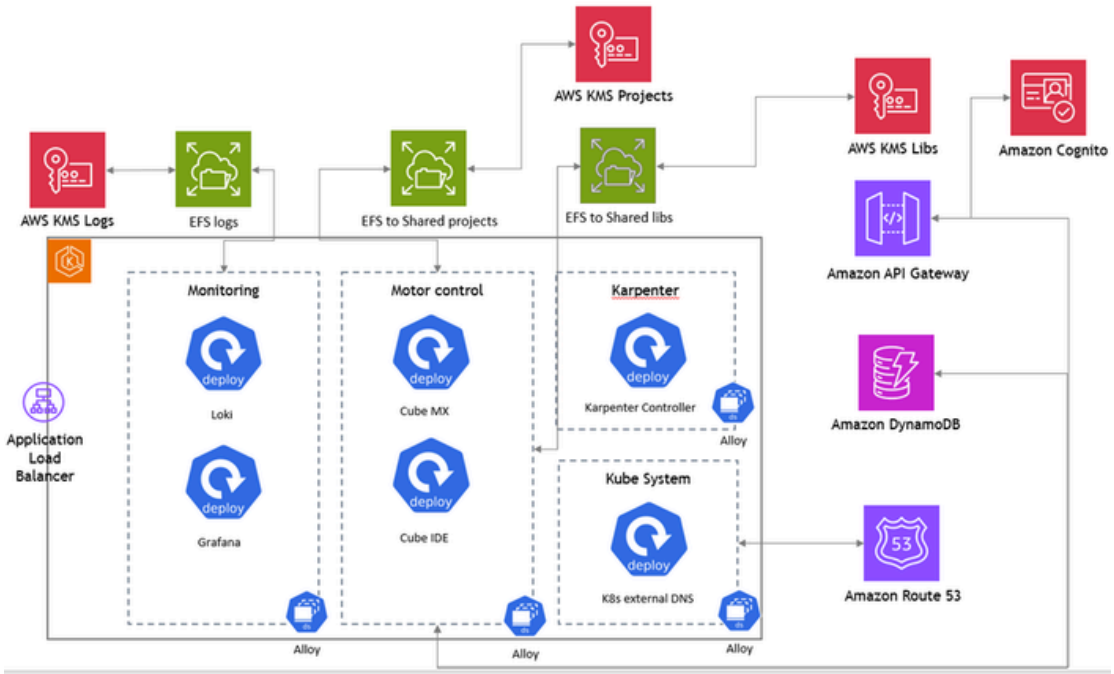
DA TOOLCHAIN DESKTOP A PIATTAFORMA CLOUD-NATIVE PER LA GENERAZIONE DI FIRMWARE MOTOR CONTROL

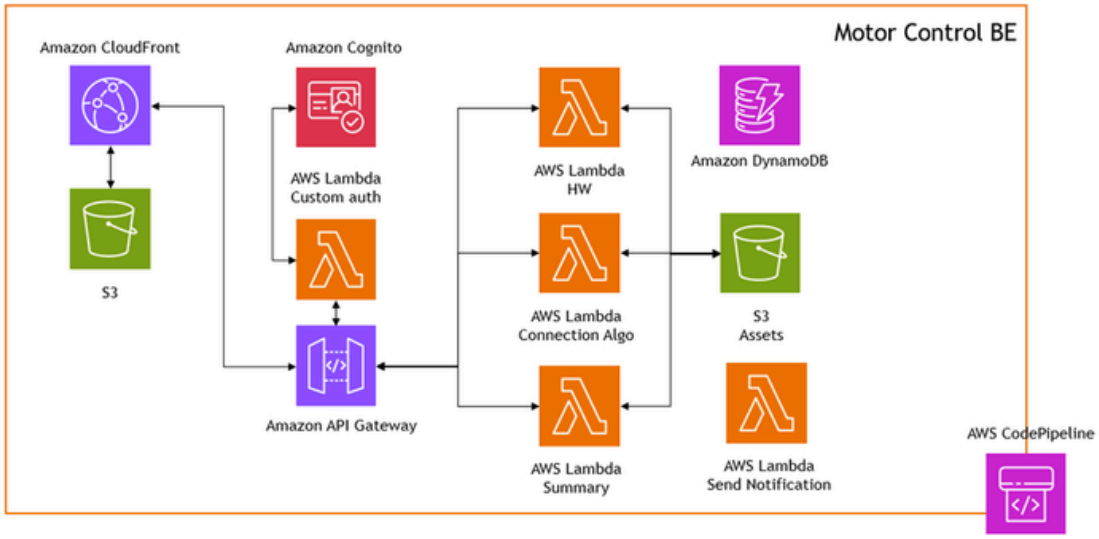
Il sistema Motor Control di ST è un ecosistema completo dedicato al controllo di motori elettrici in ambito industriale, consumer ed e-mobility. La generazione dei progetti firmware si basa su una toolchain locale composta da tre componenti principali: MC Workbench per la configurazione dei parametri, STM32CubeMX per la configurazione dell'MCU e l'integrazione del middleware Motor Control e STM32CubeIDE per la compilazione del binario finale.

L'attuale processo offline risulta vincolato all'ambiente locale dell'utente, comportando installazioni onerose, configurazioni manuali ed una rigida dipendenza dal versionamento delle librerie. Parallelamente, un Connection Algorithm (CA), sviluppato in Python, determina le connessioni tra le periferiche dell'STM32 e i segnali elettrici, mantenendo una stretta interdipendenza con il file system locale.

La soluzione sviluppata con Storm Reply introduce un cambio di paradigma: l'estensione della gestione locale verso una piattaforma cloud K8s + Serverless completamente orchestrata. Così facendo, CubeMX e CubeIDE vengono eseguiti in modalità headless su container Docker distribuiti su pod K8s dedicati. MC Workbench, che nella versione desktop opera come tool separato, viene qui integrato all'interno del container CubeMX, ottimizzando il flusso di lavoro. Il CA, invece, è stato portato in cloud come funzione AWS Lambda.

I due pilastri: un livello Serverless per orchestrazione delle richieste, autenticazione e gestione del ciclo di vita degli asset, gestito tramite CDK; un livello K8s per l'esecuzione della toolchain pesante e sottoposto a provisioning tramite Terragrunt e Terraform. L'intero progetto, inoltre, è gestito attraverso un processo CI/CD.





ARCHITETTURA IBRIDA AWS: SERVERLESS + KUBERNETES PER GENERAZIONE FIRMWARE

L'architettura realizza un sistema cloud-native su AWS per la generazione automatica di firmware motor control. Gli utenti accedono agli endpoint di generazione tramite domini custom gestiti da Amazon Route 53, con Amazon CloudFront come layer di edge e TLS termination che instrada il traffico verso il backend, migliorando latenza e sicurezza per utenti distribuiti globalmente. L'autenticazione è gestita da Amazon Cognito, integrato con API Gateway e una AWS Lambda di custom authorization che applica le policy di accesso specifiche della piattaforma.

Livello Serverless: il backend applicativo è organizzato attorno a un insieme di AWS Lambda specializzate: una per il connection algorithm (riscritto dall'originale Python desktop), una per la gestione dei metadata dell'hardware, una per la generazione di summary di progetto e una dedicata all'invio di notifiche.

Livello Kubernetes: le richieste di generazione firmware sono instradate, attraverso un Application Load Balancer gestito da AWS Load Balancer Controller, a due container specializzati: CubeMX, che configura l'MCU e integra il middleware Motor Control, e CubeIDE, che compila il binario finale. L'esecuzione headless è resa possibile da Xvfb e da un wrapper custom che intercetta i dialog GUI interattivi convertendoli in risposte programmatiche. La separazione in due container, inoltre, garantisce isolamento dei guasti, scalabilità indipendente e allocazione mirata delle risorse. Il passaggio CubeMX→CubeIDE avviene tramite un pattern di polling asincrono su Amazon DynamoDB, evitando di mantenere connessioni HTTP prolungate attraverso l'ALB su build che possono durare anche 6-7 minuti e abilitando retry idempotenti in scenari di rete instabili.

Storage e dati: S3 ospita sia gli asset generati (zip dei progetti utente) — la cui distribuzione agli utenti sfrutta S3 Transfer Acceleration per ridurre drasticamente i tempi di download in region distanti — sia le librerie sorgente MCSDK/STM32Cube FW, sincronizzate verso Amazon EFS tramite AWS DataSync. EFS funge da shared volume tra i pod CubeMX e CubeIDE, consentendo accesso concorrente alle librerie firmware senza doverle includere nelle immagini Docker. DynamoDB traccia lo stato di ogni generazione garantendo coerenza tra i due stadi della toolchain. Tutti gli asset sensibili — log, progetti utente e librerie firmware — sono cifrati at-rest tramite chiavi AWS KMS dedicate.

Piattaforma e operations: il cluster Kubernetes è equipaggiato con Karpenter per il provisioning automatico dei nodi in base al carico e con external-dns che automatizza la registrazione dei record DNS per i servizi esposti dal cluster. L'osservabilità è garantita da uno stack basato su Grafana, Loki e Alloy, quest'ultimo deployato su ogni namespace come collector. L'intera pipeline di build, test e deploy è orchestrata da AWS CodePipeline, assicurando integrazione continua e rilasci rapidi su dev, qa e prod.

Benefici architetturali: eliminazione delle installazioni locali, scalabilità automatica, parallelizzazione delle build, user experience semplificata e una piattaforma pronta ad accogliere nuovi modelli, nuove board e integrazioni intelligenti basate su GenAI.