Self-Healing Monitoring Platform - Abstract

Introduzione

Il presente progetto, sviluppato nell'ambito del percorso ITS Cloud Administrator, consiste nella progettazione e implementazione di una **piattaforma cloud-native di monitoring e self-healing automatizzato** su Amazon Web Services. L'obiettivo principale è realizzare un sistema in grado di **rilevare autonomamente anomalie e malfunzionamenti infrastrutturali e di applicare azioni correttive in tempo reale, senza necessità di intervento manuale da parte dell'operatore**.

Il progetto nasce dall'esigenza concreta, nel contesto delle moderne pratiche SRE (Site Reliability Engineering) e DevOps, di ridurre al minimo i tempi di inattività dei servizi e di automatizzare le risposte agli incidenti più comuni. La piattaforma dimostra come un'infrastruttura cloud possa essere resa **resiliente e auto-riparante** attraverso l'integrazione di strumenti di monitoring, alerting, automazione e - in chiave evolutiva - intelligenza artificiale.

Architettura del Sistema

L'architettura della piattaforma si basa su un modello a **due nodi** dislocati nella regione AWS eu-south-1 (Milano), ciascuno con un ruolo funzionale distinto, affiancati da un database relazionale managed.

Control Node ("Il Dottore")

Il primo nodo EC2 agisce come **centro di controllo e orchestrazione**. Ospita gli strumenti di automazione e visualizzazione che permettono di osservare lo stato dell'infrastruttura e di intervenire su di essa. Nello specifico, questo nodo esegue:

- **n8n**, la piattaforma di workflow automation che rappresenta il *cervello operativo* del sistema: riceve gli allarmi, valuta la situazione e orchestra le azioni correttive sui nodi target.
- **Grafana**, utilizzato come layer di visualizzazione per offrire all'operatore dashboard interattive con metriche in tempo reale provenienti da CloudWatch. È importante sottolineare che Grafana **non partecipa al flusso di self-healing**: il sistema è in grado di auto-ripararsi anche in assenza di Grafana, che serve esclusivamente per la supervisione umana.
- **Nginx**, configurato come reverse proxy con terminazione TLS (certificato self-signed), che espone n8n e Grafana su un unico punto di accesso HTTPS sicuro, proteggendo l'accesso dell'operatore ai pannelli di gestione.

Il Control Node è dotato di un **Elastic IP** (indirizzo IP statico) per garantire un punto di accesso stabile e persistente, indipendente dal ciclo di vita dell'istanza.

Worker Node ("Il Paziente")

Il secondo nodo EC2 rappresenta il **target monitorato**, ovvero il nodo che simula un'infrastruttura applicativa in produzione e sul quale vengono eseguite le azioni di remediation. Questo nodo ospita:

- **Nginx**, come web server applicativo esposto sulla porta 80, che simula un servizio web in produzione.

- **Redis**, come servizio di caching in-memory, rappresentativo di un componente applicativo critico il cui malfunzionamento richiede intervento automatizzato.
- **CloudWatch Agent**, installato e configurato per raccogliere metriche custom (utilizzo RAM e disco) che non sono disponibili nativamente tramite CloudWatch, e pubblicarle nel namespace custom per alimentare i relativi allarmi.

Il Worker Node è il principale soggetto delle operazioni di chaos testing e self-healing: è su di esso che vengono iniettati i guasti simulati e applicate le correzioni automatiche.

Database RDS

Un'istanza **RDS MySQL** managed completa l'architettura, fornendo un database relazionale persistente. Il database è predisposto per il logging strutturato degli eventi di self-healing (timestamp, tipo di incidente, azione intrapresa, esito), consentendo analisi storiche e alimentando dashboard Grafana con lo storico degli interventi automatici.

Provisioning e Infrastructure as Code

L'intera infrastruttura è provisionata e gestita tramite **Terraform**, seguendo rigorosamente il paradigma Infrastructure as Code (IaC). Ogni risorsa AWS - istanze EC2, security group, RDS, SNS topics, CloudWatch alarms, IAM roles, Elastic IP - è definita dichiarativamente in file Terraform organizzati per funzione.

Lo **state di Terraform** è conservato remotamente su un bucket **S3**, garantendo la condivisione dello stato tra i membri del team e la protezione contro la perdita accidentale dello state locale. La struttura del codice Terraform è modulare, organizzata per ambienti (`environments/dev/`), e progettata per essere **idempotente**: ogni `terraform apply` produce risultati predicibili e ripetibili senza effetti collaterali.

Il deployment iniziale dei servizi applicativi sui nodi avviene tramite **cloud-init** (`user_data`), che automatizza l'installazione di Docker, Docker Compose e l'avvio degli stack containerizzati al primo boot dell'istanza, eliminando la necessità di configurazione manuale post-provisioning.

Pipeline di Monitoring e Alerting

Il sistema di monitoring si articola su due livelli di raccolta metriche:

1. **Metriche native CloudWatch**: AWS raccoglie automaticamente metriche fondamentali come CPU Utilization e Status Check delle istanze EC2, senza necessità di agenti aggiuntivi.

2. **Metriche custom via CloudWatch Agent**: per indicatori non disponibili nativamente - in particolare l'utilizzo della memoria RAM e lo spazio disco - il CloudWatch Agent installato sul Worker Node raccoglie e pubblica queste metriche in un namespace custom su CloudWatch.

Su queste metriche sono configurati **quattro CloudWatch Alarms**, ciascuno associato a una soglia specifica:

- **CPU Utilization**: rileva situazioni di sovraccarico computazionale.
- **Memory Usage**: identifica memory leak o consumo eccessivo di RAM.
- **Disk Usage**: segnala il riempimento anomalo del filesystem.
- **Status Check Failed**: intercetta failure a livello di istanza EC2.

Quando un allarme transita nello stato ALARM, pubblica automaticamente un messaggio sul **SNS Topic** dedicato. Da qui il messaggio viene inoltrato simultaneamente via email (per audit e backup) e tramite **HTTP POST** al webhook di n8n, innescando il flusso di self-healing.

Self-Healing Automatizzato

Il cuore della piattaforma è il **motore di self-healing** implementato su n8n, che trasforma gli allarmi ricevuti in azioni correttive concrete eseguite automaticamente sul Worker Node. La piattaforma gestisce quattro scenari di failure:

Scenario 1 - Application Crash (Container Down)

Quando un container applicativo (ad esempio Nginx) crasha o si interrompe, il sistema lo rileva e procede al riavvio automatico del container, verificando successivamente che il servizio sia nuovamente raggiungibile e funzionante tramite health check HTTP.

Scenario 2 - CPU Overload

In caso di sovraccarico della CPU causato da processi anomali, il sistema identifica e termina il processo responsabile. Se l'intervento non risolve la situazione, il sistema scala ad azioni più aggressive fino al riavvio completo dell'istanza.

Scenario 3 - Memory Leak

Quando il consumo di memoria supera la soglia critica, il sistema interviene terminando i processi che consumano risorse eccessive o riavviando i container problematici, con escalation progressiva fino al reboot dell'istanza in caso di persistenza dell'anomalia.

Scenario 4 - Disk Full

Il riempimento anomalo del disco viene risolto attraverso operazioni di pulizia automatica: rimozione di immagini e container Docker inutilizzati, rotazione e compressione dei log, eliminazione di file temporanei. Il sistema verifica che lo spazio disco liberato sia sufficiente a riportare l'utilizzo sotto la soglia di sicurezza.

Ciascuno di questi scenari può essere implementato come workflow n8n dedicato oppure - nella configurazione più avanzata - consolidato in un unico workflow intelligente governato da un agente AI.

Escalation Multi-Livello

Per massimizzare l'efficacia dell'auto-riparazione, i workflow di self-healing implementano un meccanismo di **escalation a tre livelli**:

- **Livello 1 - Azione Diretta**: il sistema applica la correzione più mirata e meno invasiva (es. riavvio del container specifico). Dopo l'intervento, viene eseguita una verifica automatica per confermare la risoluzione dell'incidente.
- **Livello 2 - Azione Aggressiva**: se la verifica post-Livello 1 fallisce, il sistema scala ad un'azione più drastica (es. riavvio completo dell'istanza EC2). Una nuova verifica viene eseguita per validare l'intervento.

- **Livello 3 - Notifica Urgente all'Operatore**: se anche il Livello 2 non risolve il problema, il sistema riconosce che il malfunzionamento richiede intervento umano e invia una notifica urgente all'operatore tramite i canali configurati, fornendo tutti i dettagli diagnostici raccolti durante i tentativi precedenti.

Tra un livello e il successivo è previsto un timeout per consentire alla correzione di produrre i propri effetti prima della verifica.

Notifiche e Comunicazione

La piattaforma integra un sistema di notifiche multi-canale per mantenere l'operatore costantemente informato sullo stato dell'infrastruttura e sulle azioni intraprese:

- **Slack**: notifiche in tempo reale inviate tramite Slack Webhook, integrate direttamente nei workflow n8n. Ogni azione di healing genera un messaggio strutturato contenente il tipo di incidente rilevato, l'azione correttiva eseguita e il relativo esito.
- **Email (via SNS)**: notifiche email come canale di backup e audit trail, garantendo che ogni evento sia tracciato anche in caso di indisponibilità temporanea di Slack.

AI-Driven Healing Agent

In chiave evolutiva, la piattaforma prevede l'integrazione di un **agente AI diagnostico** all'interno di n8n. Anziché utilizzare quattro workflow separati con logica condizionale statica, un singolo workflow equipaggiato con un nodo AI Agent riceve tutti gli allarmi e utilizza un Large Language Model (LLM) per:

- **Analizzare** il payload dell'allarme (metrica coinvolta, valore rilevato, istanza target, contesto temporale).
- **Diagnosticare** la causa più probabile del malfunzionamento.
- **Selezionare** l'azione correttiva più appropriata tra i tools a sua disposizione: riavvio container, terminazione di processi anomali, pulizia disco, riavvio istanza, o escalation all'operatore.
- **Verificare** l'esito dell'intervento e generare un report dettagliato.

Questo approccio rende il sistema più **flessibile e adattivo**: l'agente AI può gestire scenari non esplicitamente previsti, combinare più azioni correttive e fornire una diagnostica più ricca e contestuale rispetto ai workflow rule-based tradizionali. La scelta del provider AI (AWS Bedrock, OpenAI, OpenRouter) verrà finalizzata durante l'implementazione in base a criteri di integrazione, costo e affidabilità.

Osservabilità e Dashboard

Grafana fornisce all'operatore un pannello di osservabilità completo attraverso dashboard interattive che attingono direttamente a CloudWatch come datasource. Le dashboard presentano:

- Metriche in tempo reale del Worker Node: utilizzo CPU, memoria RAM, spazio disco, traffico di rete, stato dell'istanza.
- Stato corrente dei CloudWatch Alarms (OK/ALARM/INSUFFICIENT_DATA) per avere una vista immediata della salute dell'infrastruttura.
- Storico degli eventi di self-healing, alimentato dai log memorizzati nel database RDS, che consente all'operatore di analizzare trend, frequenza degli incidenti e tempi di recovery.

L'autenticazione di Grafana verso CloudWatch avviene tramite il **IAM Role** assegnato all'istanza EC2 del Control Node, senza necessità di configurare credenziali statiche.

Sicurezza

La piattaforma adotta diverse misure di sicurezza:

- **Security Group** dedicati per ciascun tier (Control, Worker, Database), configurati con il principio del minimo privilegio per limitare il traffico di rete alle sole comunicazioni necessarie.
- **Reverse proxy Nginx con TLS** (certificato self-signed) sul Control Node, che cifra il traffico tra l'operatore e i pannelli di gestione n8n/Grafana.
- **IAM Role** con policy restrittive assegnati alle istanze EC2, che evitano l'uso di credenziali statiche e garantiscono il principio del minimo privilegio per l'accesso ai servizi AWS (SSM, CloudWatch).
- **Webhook n8n accessibile solo internamente** alla VPC per le comunicazioni machine-to-machine con SNS, mentre l'accesso dell'operatore avviene esclusivamente tramite HTTPS.

Chaos Testing e Validazione

Per dimostrare l'efficacia del sistema di self-healing, la piattaforma include una suite di **script di chaos testing** che simulano scenari di failure realistici sul Worker Node:

- **Crash applicativo**: arresto forzato di un container Docker per simulare un crash del servizio.
- **Sovraccarico CPU**: generazione di carico computazionale artificiale tramite stress-ng per saturare le risorse di calcolo.
- **Memory leak**: allocazione massiva di memoria per simulare un leak applicativo.
- **Riempimento disco**: creazione di file di grandi dimensioni per riempire il filesystem.

Ciascun test viene eseguito in condizioni controllate e l'intero ciclo - dall'iniezione del guasto, alla detection, alla remediation, alla verifica finale - viene monitorato e documentato per validare i tempi di recovery e l'efficacia delle azioni correttive.

Stack Tecnologico

Il progetto integra un ampio ventaglio di tecnologie cloud e DevOps:

Area	Tecnologia	Ruolo
Infrastructure as Code	Terraform	Provisioning dichiarativo di tutte le risorse AWS
Compute	EC2 (Amazon Linux 2023)	Hosting dei nodi Control e Worker
Database	RDS MySQL	Storage persistente managed per logging eventi
Containerizzazione	Docker + Docker Compose	Runtime isolato per tutti i servizi applicativi
Automazione	n8n	Workflow engine per self-healing e orchestrazione

Area	Tecnologia	Ruolo
Monitoring	CloudWatch + CloudWatch Agent	Raccolta metriche native e custom, gestione allarmi
Visualizzazione	Grafana	Dashboard operatore con datasource CloudWatch
Alerting	SNS (Simple Notification Service)	Routing degli allarmi verso webhook e email
Reverse Proxy	Nginx	Terminazione TLS e routing verso n8n/Grafana
Notifiche	Slack + Email	Comunicazione real-time e audit trail
State Management	S3	Backend remoto per lo state di Terraform
Permessi	IAM Roles	Accesso sicuro ai servizi AWS senza credenziali statiche
AI (Evolutiva)	LLM (Bedrock/OpenAI/OpenRouter)	Diagnostica intelligente e scelta azioni correttive

Competenze Dimostrate

Il progetto è progettato per dimostrare in modo pratico un ampio insieme di competenze trasversali nell'ambito del Cloud Administration:

- **AWS Cloud Services**: utilizzo integrato di EC2, RDS, S3, SNS, CloudWatch, IAM, VPC e Security Groups.
- **Infrastructure as Code**: provisioning completo tramite Terraform con state remoto, struttura modulare e best practice di idempotenza.
- **Versionamento**: utilizzo di Git per il controllo di versione del codice sorgente e della configurazione dell'infrastruttura.
- **Containerizzazione**: deployment di applicazioni multi-container con Docker Compose, gestione volumi persistenti e networking Docker.
- **Automazione e Orchestrazione**: progettazione di workflow di automazione complessi con escalation, verifica post-azione e integrazione multi-servizio.
- **Monitoring e Osservabilità**: implementazione di una pipeline completa di raccolta metriche, alerting e visualizzazione.
- **Sicurezza**: applicazione del principio del minimo privilegio, cifratura del traffico, segregazione di rete.
- **Pratiche SRE/DevOps**: self-healing, chaos engineering, incident response automatizzata, riduzione del MTTR (Mean Time To Recovery).
- **Intelligenza Artificiale applicata alle Operations**: integrazione di modelli LLM per diagnostica e decision-making automatizzato nell'ambito delle operazioni infrastrutturali (AIOps).

Deliverable

Il progetto si concretizza in tre deliverable principali:

1. **Repository Git** contenente il codice Terraform, i workflow n8n esportati, le dashboard Grafana, gli script di chaos testing e la documentazione tecnica completa.
2. **Live Demo su AWS** con dimostrazione in tempo reale del ciclo di self-healing: iniezione di un guasto sul Worker Node, rilevazione automatica tramite CloudWatch, esecuzione della remediation tramite n8n, verifica del recovery e notifica all'operatore.
3. **Documento di Progetto** con descrizione dell'architettura, motivazione delle scelte tecniche, mappatura delle competenze ITS dimostrate e analisi dettagliata degli scenari di incidente gestiti.