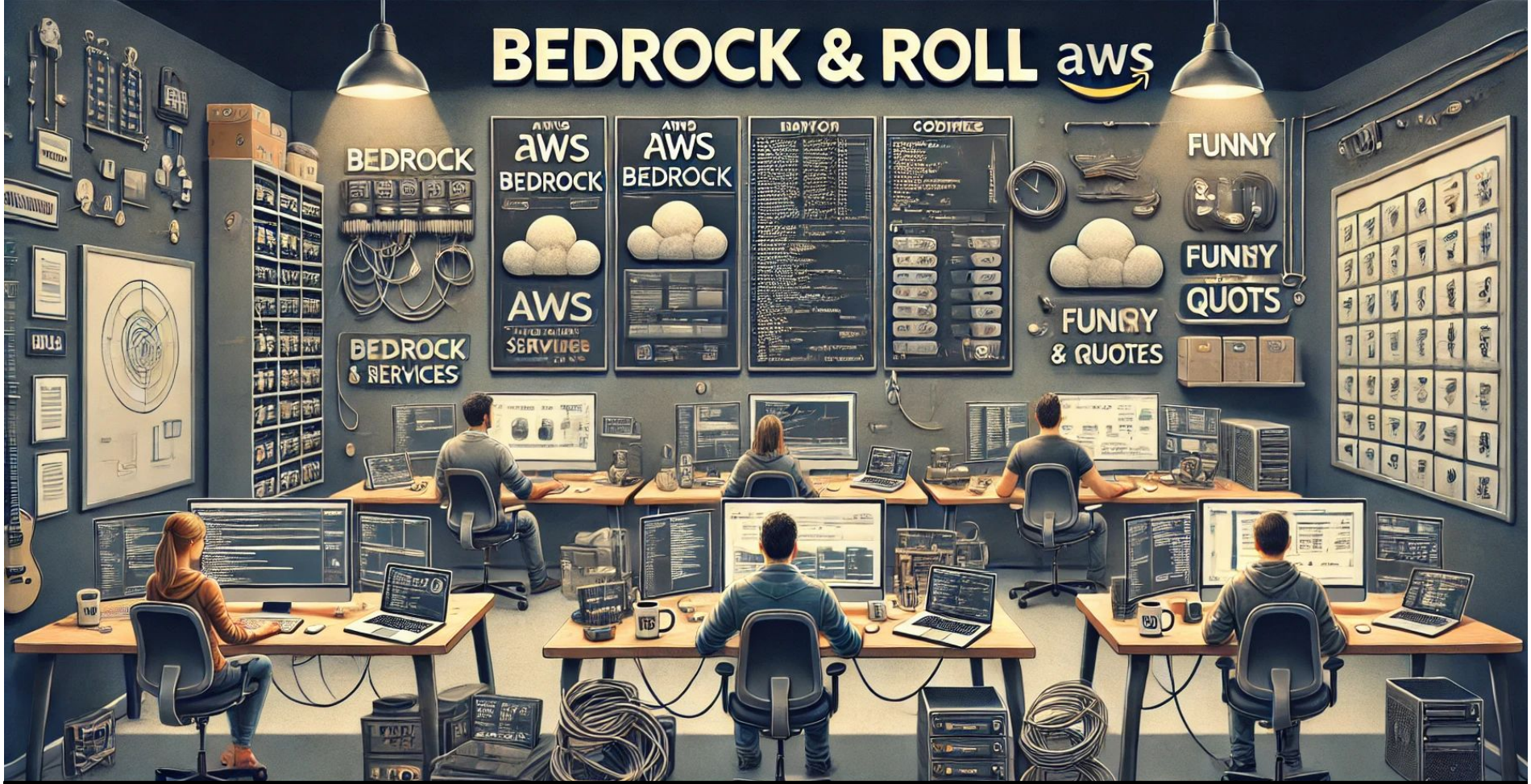


Building Mess-Erprise AI Systems That Actually Work

1. Il Contesto “Mess-Erprise”

- Abbiamo un progetto che richiede interazioni complesse con un sacco di dati sparsi. Diciamo pure un delirio di scartoffie, documenti e utonti.

- Ci sono questionari, form e noiosissimi annessi e connessi da orchestrare in tempo (più o meno) reale.

- "Mess-Erprise" come mentalità: riconoscere che i clienti cambiano idea ogni mezz’ora e i sistemi si sfasano ad ogni release.

Quando il cliente ha cambiato idea 6 volte in un giorno, abbiamo capito che ‘Mess-Erprise’ era una descrizione ottimistica della situazione... ma alla fine quale progetto non è così?

2. Focus su Amazon Bedrock

- **LLM as a Service:** Per gestire modelli foundation e versioni custom (fine-tuning) senza sclerare con server GPU e code davanti al frigo.

- **Gestione Sicurezza & Compliance:** Infrastruttura AWS + IAM, VPC, e controlli vari.

- **Scalabilità & Pay-as-you-go:** Cresce (o si riduce) quando arriva l'onda di richieste, e tu paghi in base a quanta caffeina consumi.

- **Custom Fine-Tuning:** Prendere un modello base e strapazzarlo per i propri scopi, senza dover invocare rituali demonia

Quando mi chiedono 'Ma non basta ChatGPT?', rispondo: 'Ma ChatGPT è un coltellino svizzero di default, Bedrock è un arsenale di coltelli di vario tipo – e noi stiamo imparando a usarli senza tagliarci.

3. Architettura a Grandi Linee

- **Frontend (React)** – con WebSocket per la chat in real-time
- **Backend (FastAPI)** – gestisce sessioni di interazione, autenticazione (AWS Cognito), e coordina i LLM.
- **AI:**
  a. Bedrock – lo zoccolo duro dell'AI, che fornisce l'accesso a vari modelli (foundation + custom).
  b. altri llm saas sparsi
- **Storage** – S3 per file e log, DynamoDB per i dati dinamici, MySQL per altre entità standard.
- **Compliance** – per dormire sereni la notte, confinando i sistemi in reti private e usando IAM, ruoli, cognito e magia.

Flow Tipico di un'Interazione:

- L'utente chiede qualcosa a voce → STT con web api o AWS Transcribe -> Il modello su Bedrock elabora → generiamo una bella risposta → la comunichiamo in modo opportuno (testuale, vocale, etc)

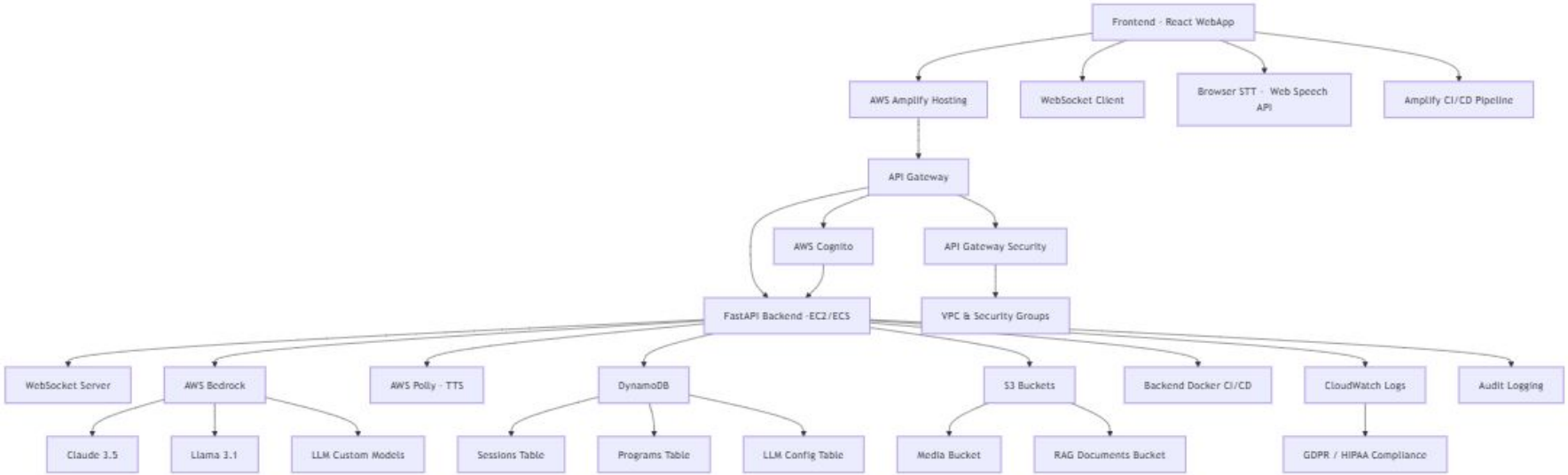
A breve sperimenteremo la comunicazione telepatica su Bedrock... se AWS ce la implementa entro venerdì!

4. Motivi (seri) per cui usiamo Bedrock

- **Modelli Foundation di Qualità:** Non ci affidiamo a qualcosa di sperimentale e super-beta: abbiamo a disposizione i modelli ben testati di AWS e partner.
- **Custom Fine-Tuning Integrato:** Possiamo pasticciare (ehm, 'personalizzare') i modelli con i nostri dati senza dover affittare GPU in parallelo.
- **Sicurezza e Compliance by Default:** Stare su AWS significa avere IAM, VPC, e tutte le scartoffie pronte per le normative di turno.
    - Compliance 2: "Gestiamo utenti e ruoli con Cognito, così quando qualcuno chiede 'Hai i permessi?', possiamo tirare fuori i token come se fossero caramelle."
- **Pay-as-you-go:** "Se un giorno non vogliamo più far girare i modelli, stacchiamo la spina. Fine delle spese (salvo la caffeina)."
- **Integrazione con l'Ecosistema AWS:** "Tiriamo in ballo Cognito, S3, DynamoDB, e tutto il resto della 'scatola dei balocchi' in modo nativo."
- **Easy Deploy:** perché complicarci la vita?
- **Scalabilità:** il nostro target è quello di dover sostenere fino a 1000 sessioni concorrenti (ad oggi, ma il futuro è sempre incerto), con gente che bombarda il sistema di richieste

molte cose sono sotto i famigerati NDA, il tempo maggiore per scrivere questa presentazione è stato controllare di non aver violato questi patti con i vari diavoli

4. grafici? (se ho tempo di farli)



li disegno a mano? :P o trovo la giusta AI?

Marco Gaion @ Walit

6. Le Sfide Principali

- **Latency**: se c’è un ritardo nel rispondere, la gente si lamenta subito.

- **Normative/compliance**: chiamiamole come vogliamo, ci sono regolamenti e leggi che non possiamo ignorare.

- **Requisiti in Evoluzione**: un giorno vogliono 10 campi di input, il giorno dopo 50, poi 200. E noi dietro col retino.

Una volta un cliente ci ha chiesto di aggiungere la compatibilità con la macchinetta del caffè, é stato uno dei requisiti implementati con maggiore gioia

7. Best Practices

1. **Modularità:** meglio a microservizi, così se uno schianta, gli altri fingono di non conoscerlo.

2. **Test Iterativi:** piccoli Proof of Concept (PoC), sennò rischiamo di scoprire i guai a fine progetto.

3. **Security by Design:** Cognito, encryption, permission e paranoia costruttiva.

4. **Monitoraggio & Alerting:** se non controlli i costi e l'uso, finisci a chiedere finanziamenti in lacrime.

9. Q&A (domande [im]possibili)

Come gestite il budget LLM? Ci affidiamo a "stime" (script e test di carico, analisi tramite strumenti forniti) e soprattutto a un microchip nel cervello che urla quando superiamo i limiti.

E se cambio modello? Bedrock permette di swappare e testare LLaMA e vari cugini. Scegli tu in base all'umore.

E la privacy? AWS compliance + incrociare le dita, perché i regolamenti si aggiornano più velocemente di noi.