Technische Hochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Informatik und Medien

Serverless Chat Application – Architektur & Delivery

Echtzeit-WebSocket-Chat auf AWS (Vue 3, Lambda, DynamoDB)

Oliver-René Goltsche · Matrikelnummer 20226965 · Technische Hochschule Brandenburg

Gliederung der Präsentation

1. Zweck und Features

2. Softwarearchitektur
3. AWS-Architektur
4. Datenmodell

5. CI/CD-Prozess
6. Live-Demo
7. Risiken & Verbesserungen

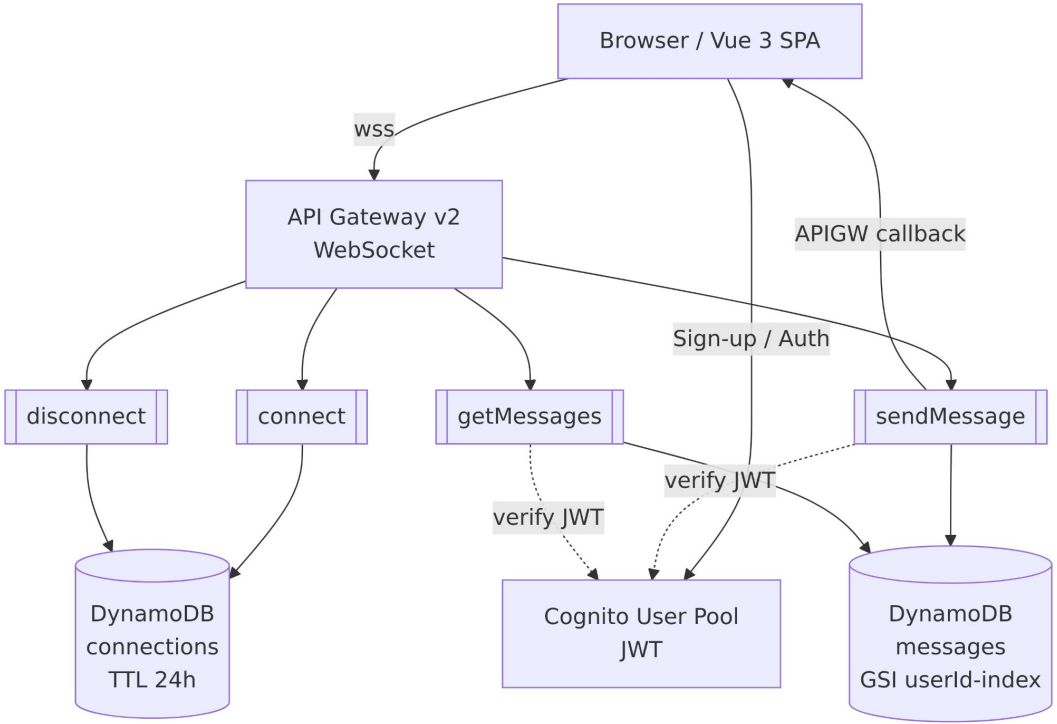
Zweck und Features

- **Zweck:** Serverloser Echtzeit-Textchat für mehrere Nutzer ohne eigenen Serverbetrieb
- **Nutzen:** Automatische Skalierung und nutzungsbasierte Kosten durch AWS-Serverless
- **Features:** Authentifizierung, Chat-Räume, Nachrichtenspeicherung und Verlauf mit Pagination, Automatisches Cleanup inaktiver Verbindungen per 24h-TTL
- **Scope:** Kein Datei- oder Bildtransfer (bewusst reduziert für Fokus und Sicherheit)

Softwarearchitektur

- **Frontend:** Vue 3 (Vite, Pinia)
- **Backend:** 4 Lambda-Handler (connect, disconnect, sendMessage, getMessages) hinter API Gateway WebSocket
- **Flow:** Login → WebSocket mit userId/username → Nachricht in DynamoDB speichern + Broadcast → History per Query; TTL räumt Connections.

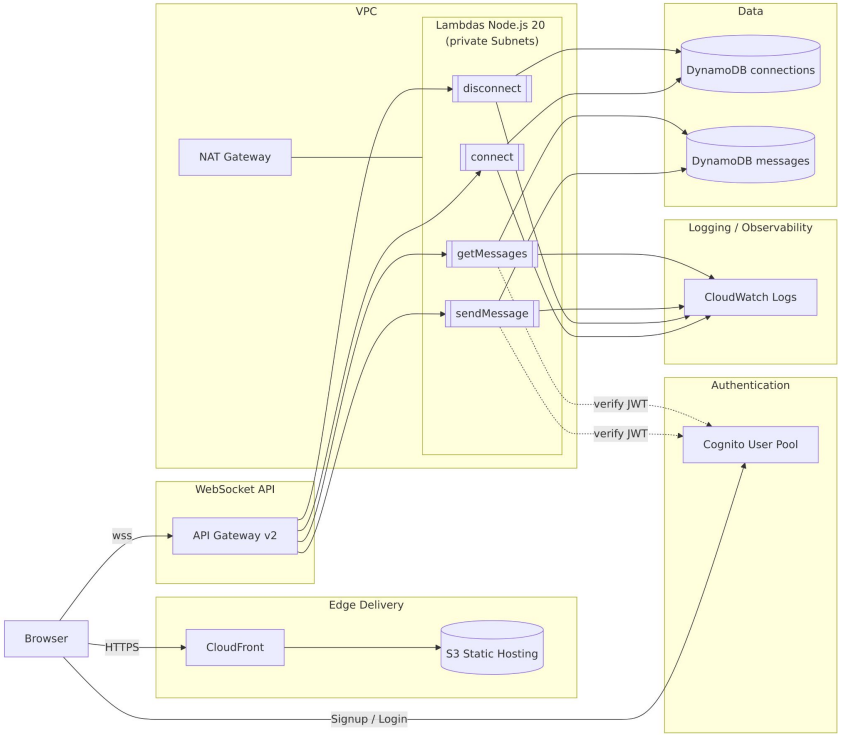
Softwarearchitektur



AWS-Architektur

- **Edge & Entry:** CloudFront + S3 (SPA), API Gateway WebSocket als einziger Echtzeit-Einstieg.
- **Compute & Auth:** Lambdas (Node 20) in privaten Subnetzen hinter NAT; Cognito User Pool als Auth-Provider.
- **Daten & Ops:** DynamoDB connections (TTL) + messages; CloudWatch Logs (14 Tage), Terraform verwaltet alle Ressourcen.

AWS-Architektur



Datenmodell (DynamoDB)

- connections: PK connectionId; Felder userId, username, connectedAt, ttl (24h Autocleanup)
- messages: PK roomId, SK timestamp#messageId; Felder userId, username, content, createdAt
- GSI userId-index: gezielte User-Historie; On-Demand Billing für elastische Last

CI/CD-Prozess (GitHub Actions)

- Änderungsfluss bis Produktion: PR, Review, Merge, danach automatisierte Pipeline
- Qualitäts-Gates vor Release: Linting, Typprüfung, Tests sowie IaC- und Security-Checks
- Deployment-Reihenfolge mit geringerem Risiko: Infrastruktur, dann Backend, dann Frontend
- Reproduzierbare lokale Entwicklung über standardisierte Container-Umgebung

Live-Demo

Risiken & Verbesserungen

- Skalierung: Broadcast bei tausenden gleichzeitigen Nutzern
- Zuverlässige Zustellung unter Last
- Observability und Incident-Response

Technische Hochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Informatik und Medien

Vielen Dank für Ihre Aufmerksamkeit

Echtzeit-WebSocket-Chat auf AWS (Vue 3, Lambda, DynamoDB)

Oliver-René Goltsche · Matrikelnummer 20226965 · Technische Hochschule Brandenburg