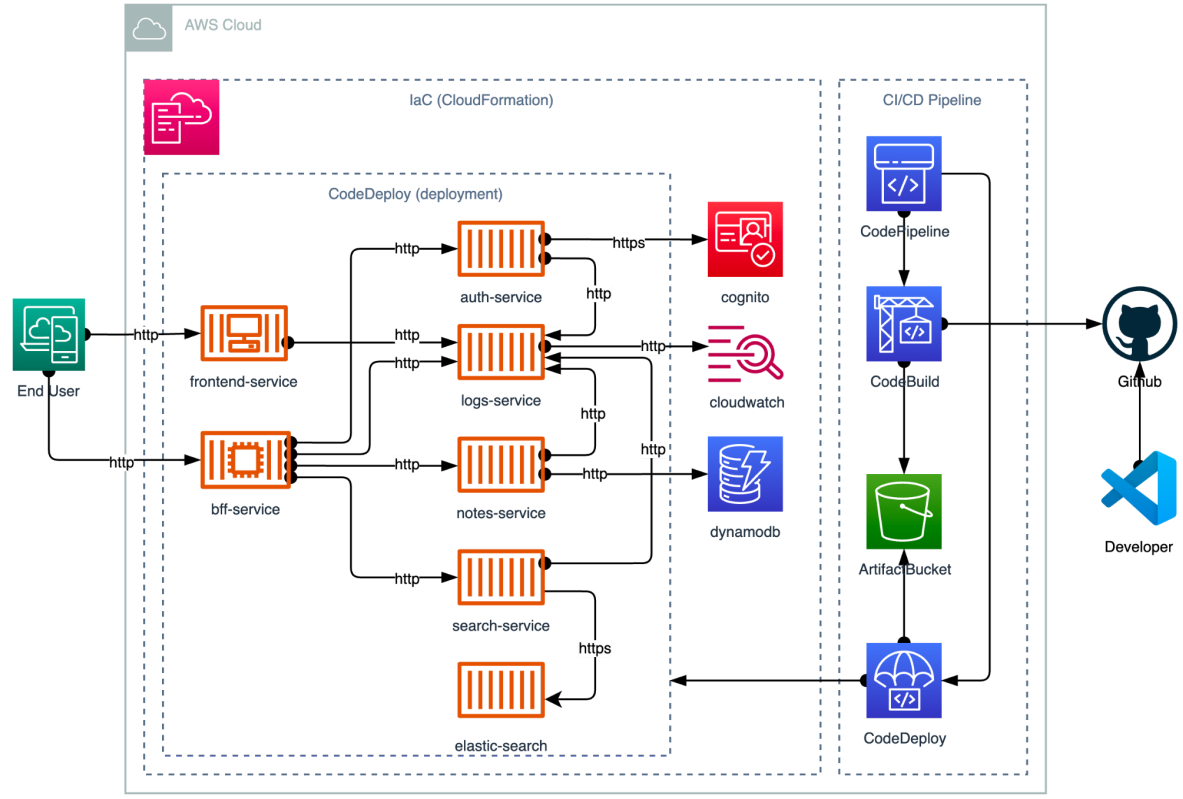
Complex Web Application - High-level architecture - Don Athalage



This is the high-level architecture of the application. I have made the following key architecture decisions about the application to progress with the application development. Please note that this depiction is

Key Decision	Choice	Alternatives	Rational
Programming Language	Golang	Javascript	As a learning experience as it would help me to progress at my workplace
Cloud Platform	AWS	GCP, Azure	I have extensive experience in AWS and can create advanced cloud-native applications with the expertise
IaC language	CloudFormation	Terraform	Simplicity and advanced platform features
CI/CD Tooling	Github, CodePipeline, CodeBuild, CodeDeploy	Jenkins, CodeCommit	Simplicity, Advanced platform features, Cost savings and ease of implementation
Compute	EKS or ECS	EKS, ECS, Fargate, EC2	This decision will be finalised on cost concerns and implementation details.

Development Steps and projected durations

- Planning: **DONE**
- Setup CICD Pipelines: 1 Week
- Setup IaC Code: 1 Week
- Deploy Infrastructure: 1 Day
- Application Development: 5 Weeks
- Deployment Testing: 1 Week

Services

1. Frontend Service (frontend-service) - Simple React notes application that exposes the following functionality
    a. User Signup/Login
    b. Create/Read/Update/Delete Notes
    c. Search Notes
2. Backend for Frontend Service (bff-service) - Backend Service that processes the frontend requests by sending subsequent requests to other backend services
3. Authentication Service (auth-service) - Exposes API endpoints to be used by bff-service to,
    a. api/create-user/ :POST
    b. api/login-user/ :POST
    c. api/validate-user/ :POST
4. Notes Service (notes-service) - Exposes APIs to CRUD notes once users are authenticated. Logs are sent to logs-service. DynamoDB is used as the database
    a. api/list-notes/ :GET
    b. api/create-note/ :POST
    c. api/read-note/ :GET
    d. api/update-note/ :PUT
    e. api/delete-note/ :DELETE
5. Search Service (search-service) - Exposes API to query some keywords in notes. This loads all the notes from the user to elastic-search service and query for keywords.
    a. api/search: POST
6. Logs Service (logs-service) - Receive logs from all the services and submits them to CloudWatch Logs for audit purposes.
    a. api/log: POST
7. Elastic Search Service (elastic-search) - search-service loads notes into elastic search service when there is a search request and searches for notes. Returns relevant notes when there are matches.