

I'd like to build a chat bot web service using Bedrock. Can you make me a brief architecture for me?

Sure, I can provide you with a brief architecture for building a chatbot web service using Amazon Bedrock. Here's a high-level overview: [1]

1. **Frontend** : Develop a web application using a frontend framework like React, Angular, or Vue.js. This will serve as the user interface for your chatbot, where users can type in their queries and receive responses.
2. **Backend** : Set up a backend server using a language like Python, Node.js, or Java. This server will handle the communication between the frontend and Amazon Bedrock. [2]
3. **Amazon Bedrock** : Subscribe to the desired foundation model (FM) from Amazon Bedrock. You can choose from various FMs offered by leading AI companies like Anthropic, Cohere, and others. [3]
4. **Agents (Optional)** : If your chatbot requires complex reasoning, task orchestration, or integration with external data sources, you can create agents using Amazon Bedrock Agents. Agents automate prompt engineering and securely augment prompts with company-specific information.
5. **Knowledge Base (Optional)** : If your chatbot needs to retrieve information from external data sources, you can create a knowledge base using Amazon Bedrock Knowledge Base. This allows you to ingest and index your data for efficient retrieval.
6. **API Gateway** : Use AWS API Gateway to create a secure API endpoint that your backend server can call to interact with Amazon Bedrock.
7. **Lambda Functions (Optional)** : If you need to perform additional processing or integrate with other AWS services, you can use AWS Lambda functions triggered by API Gateway or other events.
8. **Authentication and Authorization** : Implement appropriate authentication and authorization mechanisms to secure your chatbot service and protect user data.
9. **Monitoring and Logging** : Set up monitoring and logging mechanisms to track the performance and usage of your chatbot service, and to troubleshoot any issues that may arise.
10. **Deployment** : Deploy your frontend application to a hosting service like AWS Amplify or S3, and your backend server to a service like AWS Elastic Beanstalk or ECS.

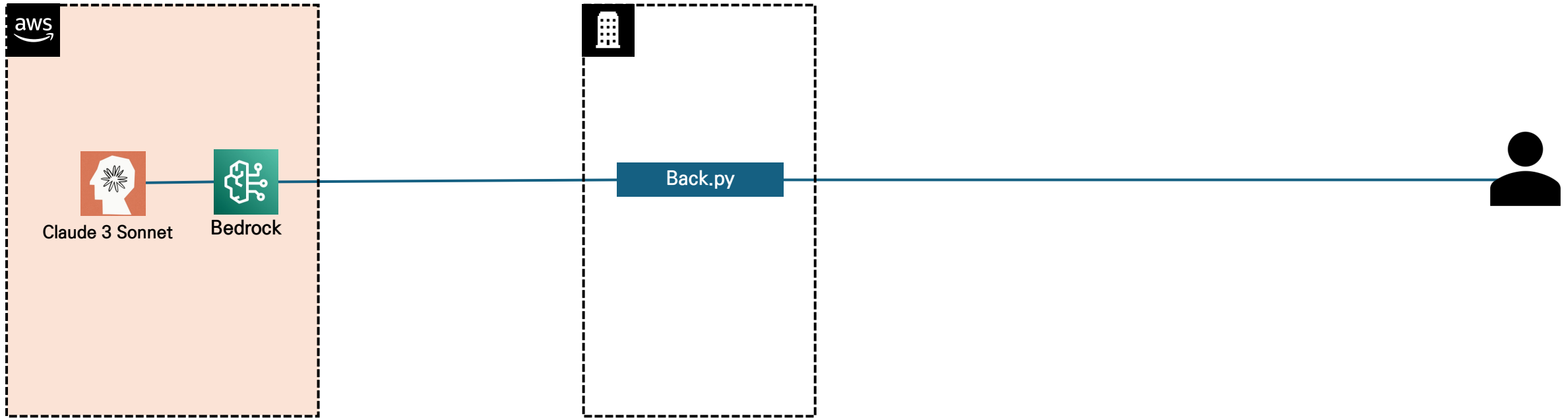
This is a high-level architecture, and you may need to adjust it based on your specific requirements, such as the complexity of your chatbot, the need for integration with other services, and the expected load and scalability requirements.

Foundation Model 선정의 문제

- Partyrock 모델은 Claude 3 Sonnet 기반
- 개발 테스트를 위해서는 저렴한 모델 (Claude 3 Haiku) 사용
- 실제 구축/테스팅할 때는 Claude 3 Sonnet 사용
- Llama, Titan 모델도 테스트해보면 좋는데.. 복잡한 프롬프트 시나리오 만들어보기
- 요금 나오는 거 일단 지켜보자... 월 10달러 상한선 세팅된 상태
- 모델의 온도, 무작위성 등 튜닝은 솔루션 구축 후에 다듬으면 된다.

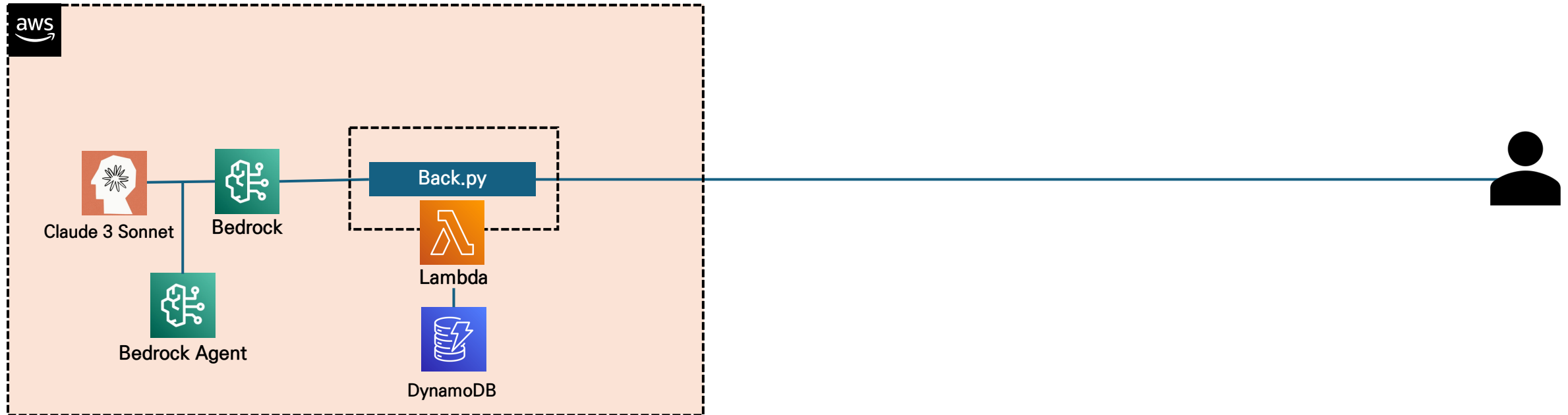
Phase 0: 개발 가능성 검토

- 챗봇에 질문을 던지면 응답이 가능한 기본 구조
- 로컬에서 백엔드 파일로 Bedrock API 접근
- 챗봇은 별도의 Agent가 붙어 있거나, 이전 대화를 기억하지는 못하는 상태



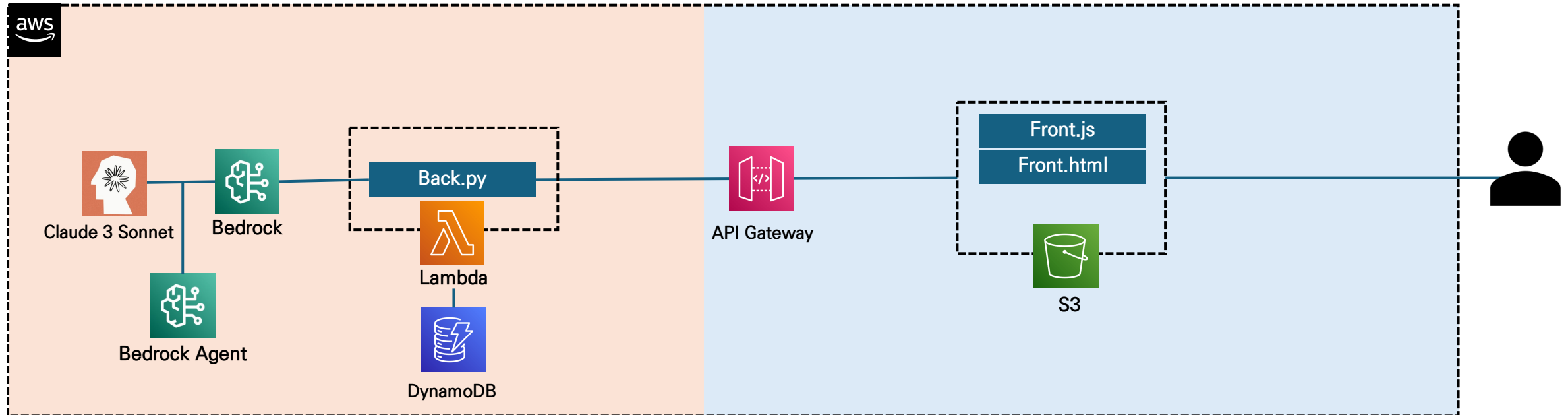
Phase 1: 백엔드 필수 요소 구축

- 챗봇에게 역할을 제시하는 Agent를 직접 구현
- 백엔드 파일은 서버리스로 Lambda 활용, python 외 다른 언어 써도 OK
- 단일 세션을 가정하고 메모리 구현 – 일단 레퍼런스 참고해서 DynamoDB 활용
- 인터랙션은 shell 등 개발 환경 사용



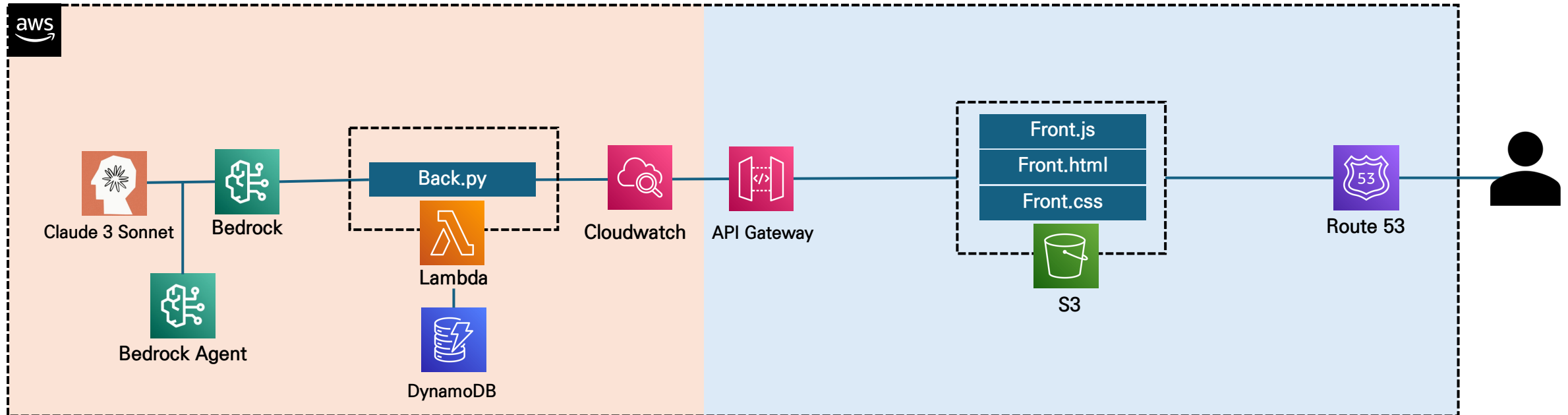
Phase 2: 프론트엔드 필수 요소 구축

- 백엔드 기본 구조를, 사용자 인터랙션이 가능하도록 API Gateway 구축
- 프론트는 정적 웹페이지로 S3에 적재
- 프론트에서 스크립트를 캐치해서 역으로 전달하는 경로 필요
- 실행 시 웹페이지 직접 접근 필요한 상태



Phase 3: 솔루션 완성

- 모니터링 구현 (Cloudwatch)
- 웹 디자인 + CSS 구현
- Route 53 등으로 도메인 접속 구현



Phase 4: 솔루션 고도화

- CloudFront 캐싱하여 속도 향상
- Cognito로 로그인 기능 구현 + 프론트 상에도 로그인 페이지 구현
- Lambda 및 DynamoDB 최적화, 추가 보안 조치 검토

