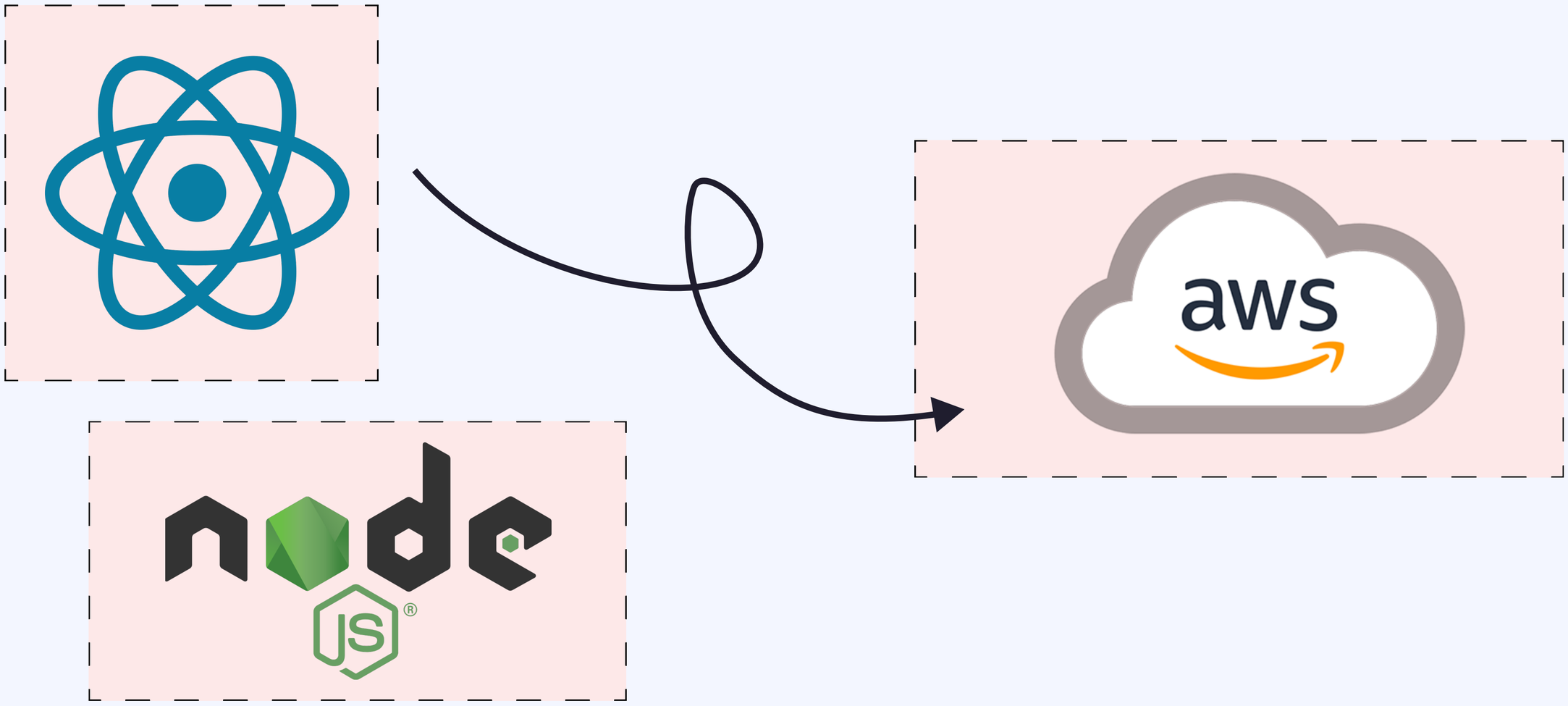
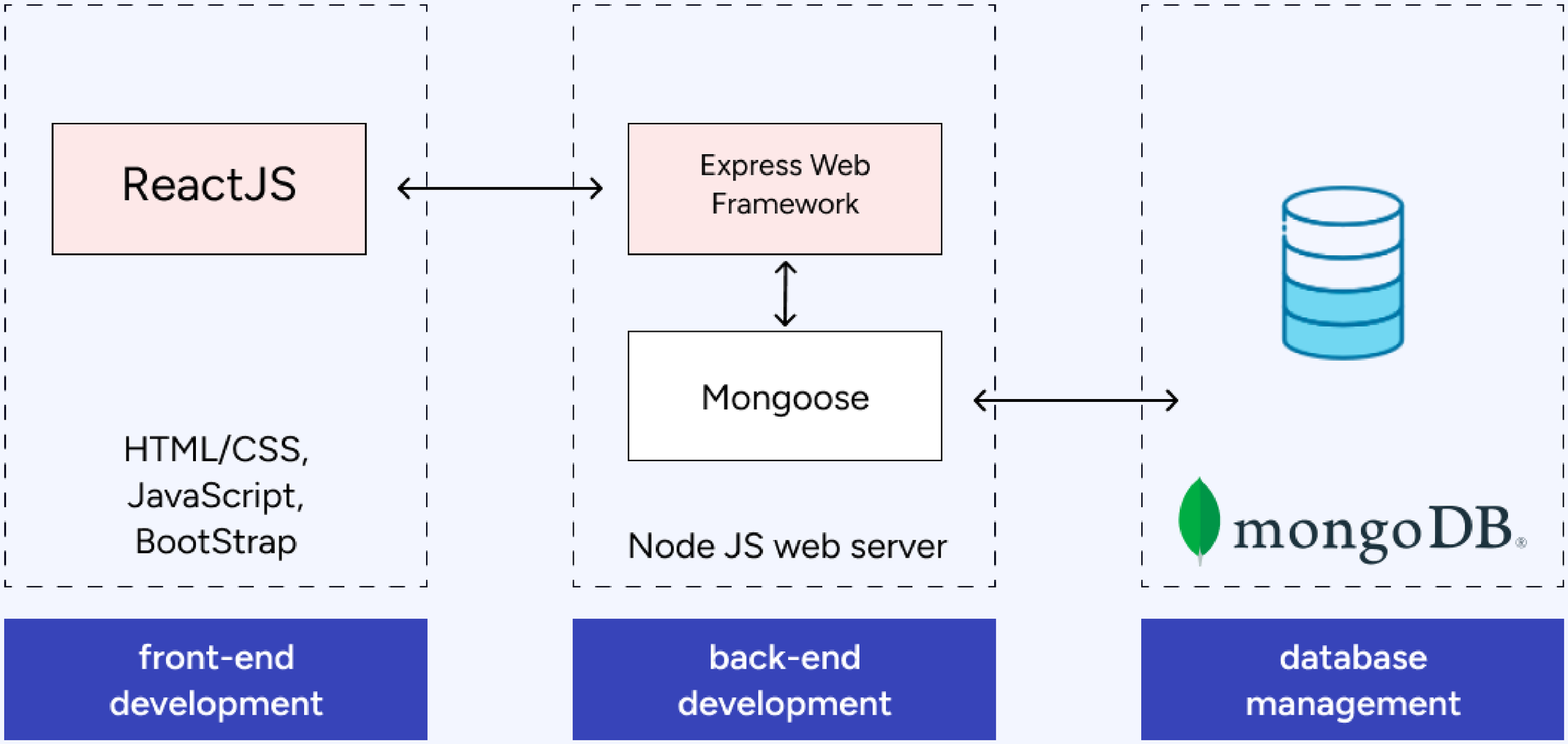
How to Deploy Node.js React App to AWS



gart.

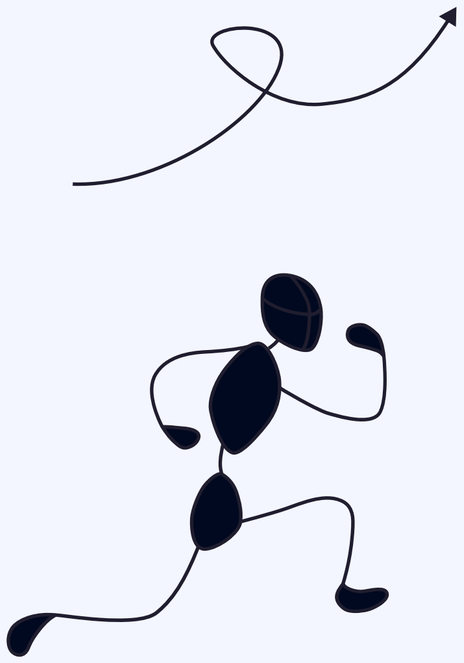
The MERN stack

- **M** stands for MongoDB - NoSQL Database
- **E** stands for Express.js - Web Application Framework for Node.js
- **R** stands for React - Frontend Library for Building User Interfaces
- **N** stands for Node.js - JavaScript Runtime for Server-Side Development



Classic Deployment Methods on AWS

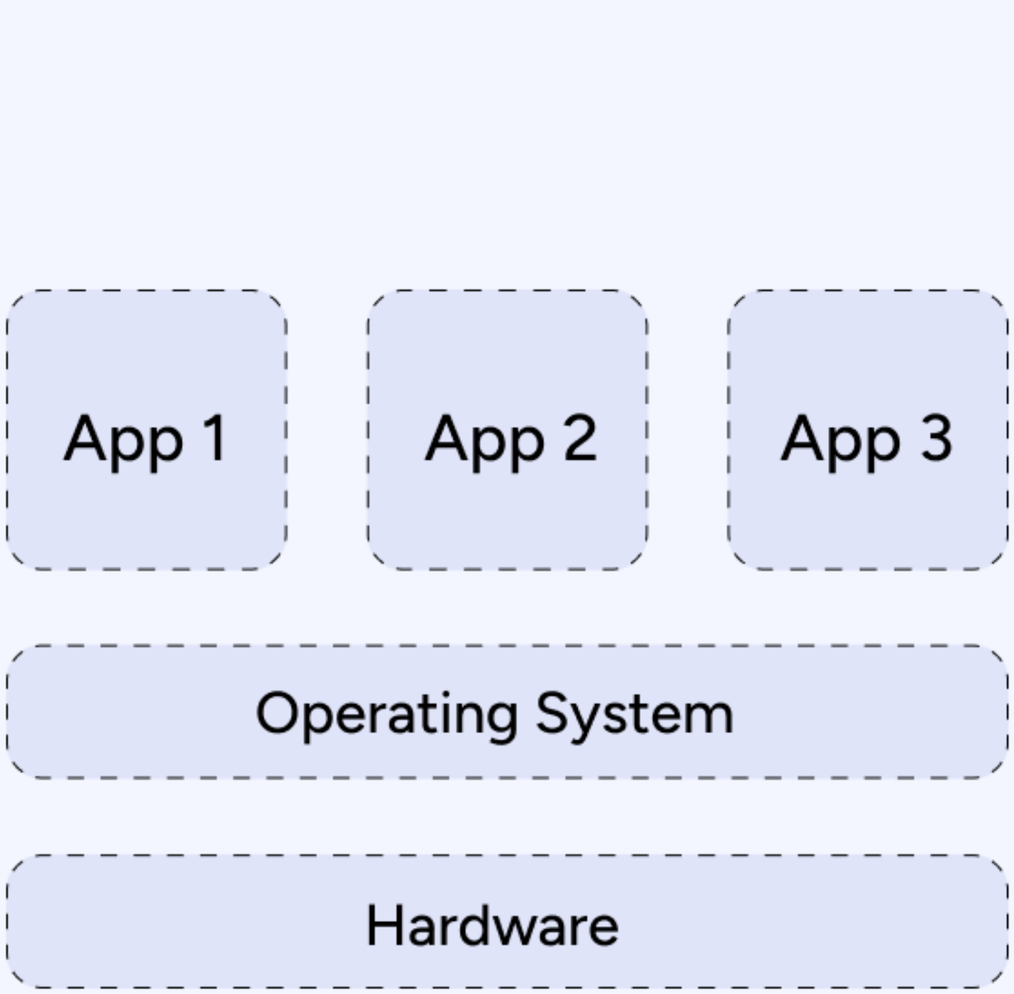
- Traditional VM Deployment
- Docker Containers
- ECS or EKS for Container Orchestration
- Serverless Approach with AWS Lambda
- AWS Amplify for Streamlined Deployment



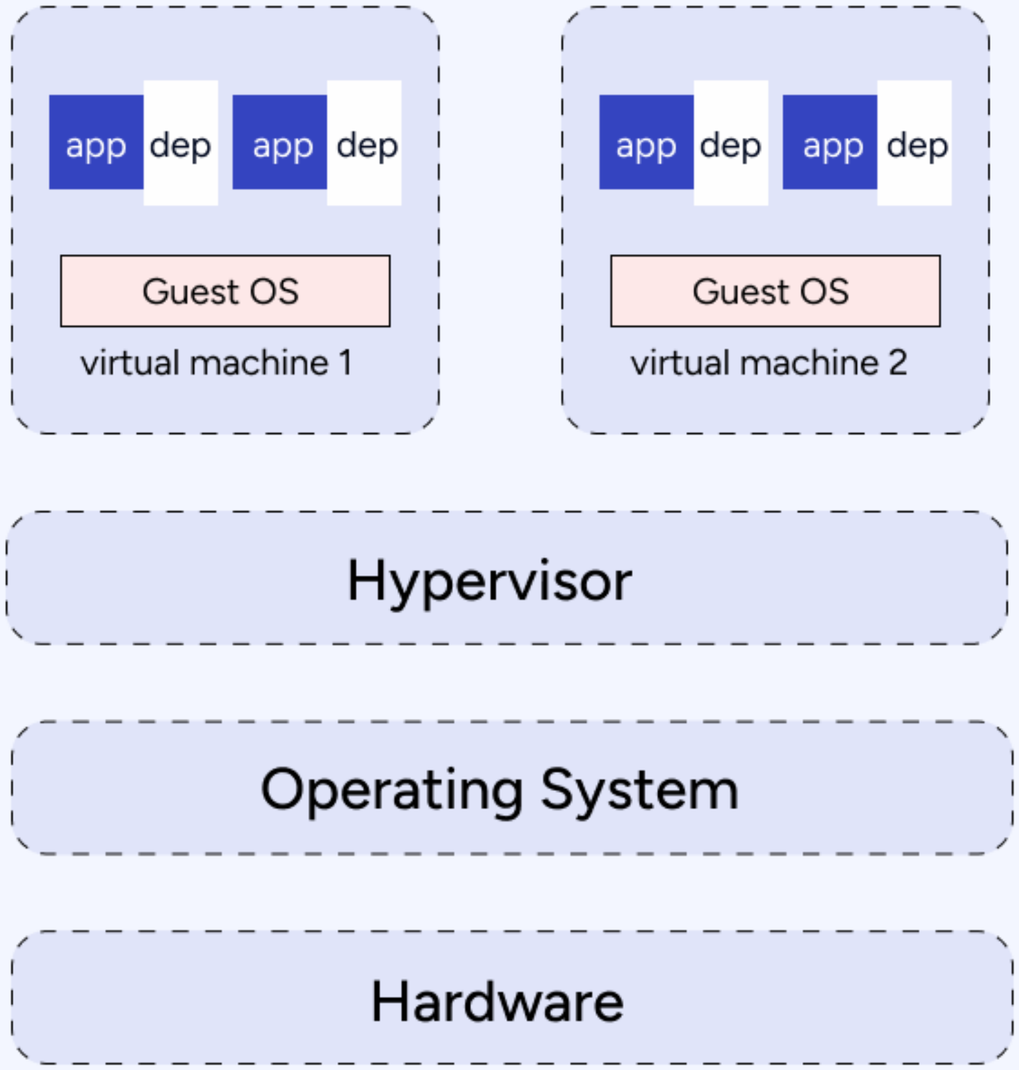
gart.

Traditional VM Deployment

- Pros: Full control over infrastructure.
- Cons: Manual scaling, limited flexibility.



Traditional software deployment model

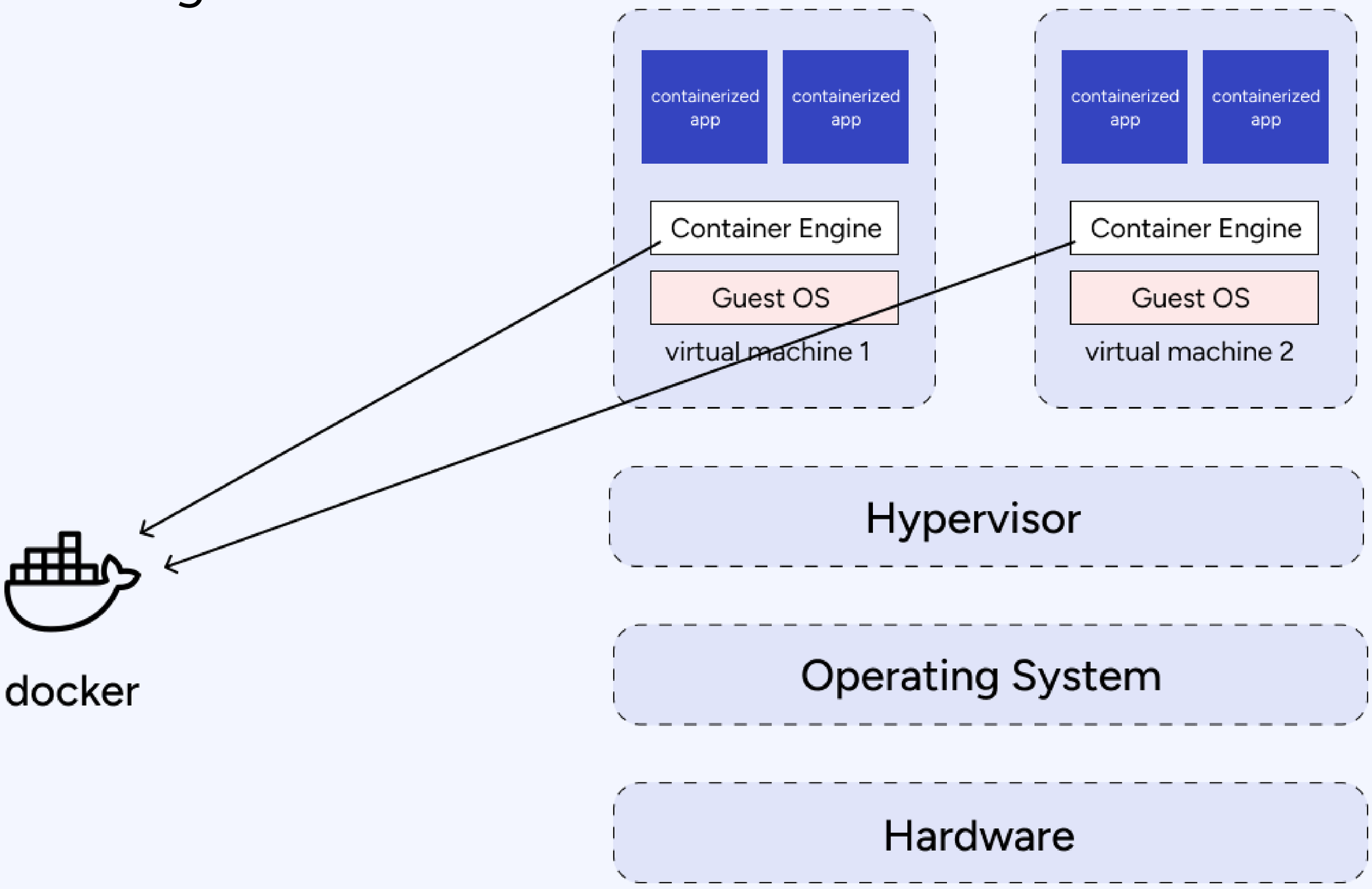


Virtualized software deployment model

gart.

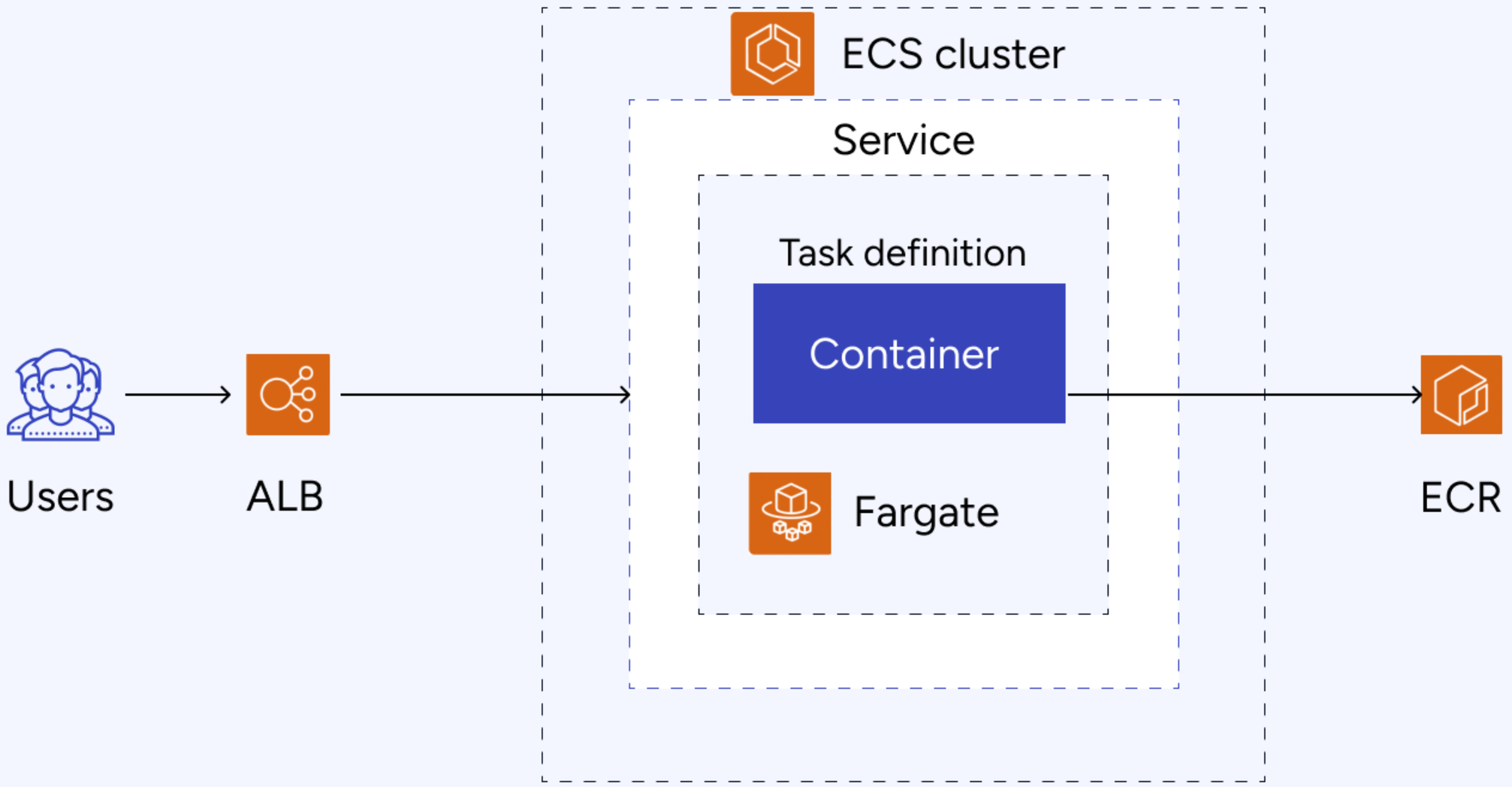
Containerized Deployment with Docker

- Pros: Isolation, easy scalability, and flexibility.
- Cons: Requires Docker knowledge, manual management.



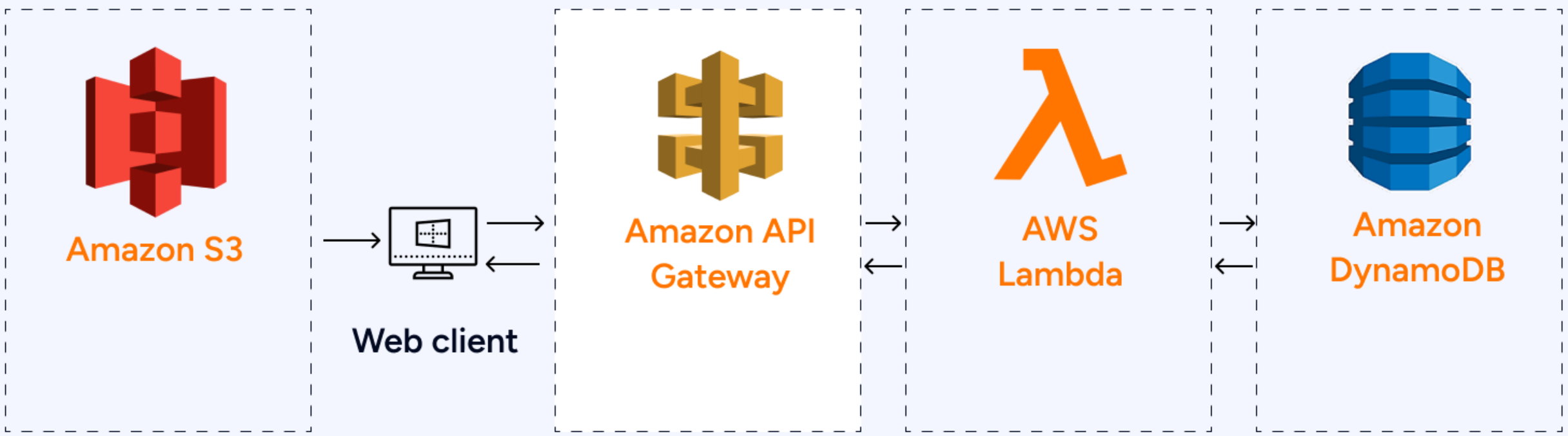
Deploying with AWS ECS or EKS

- Pros: AWS-managed container orchestration.
- Cons: Learning curve, potential complexity.



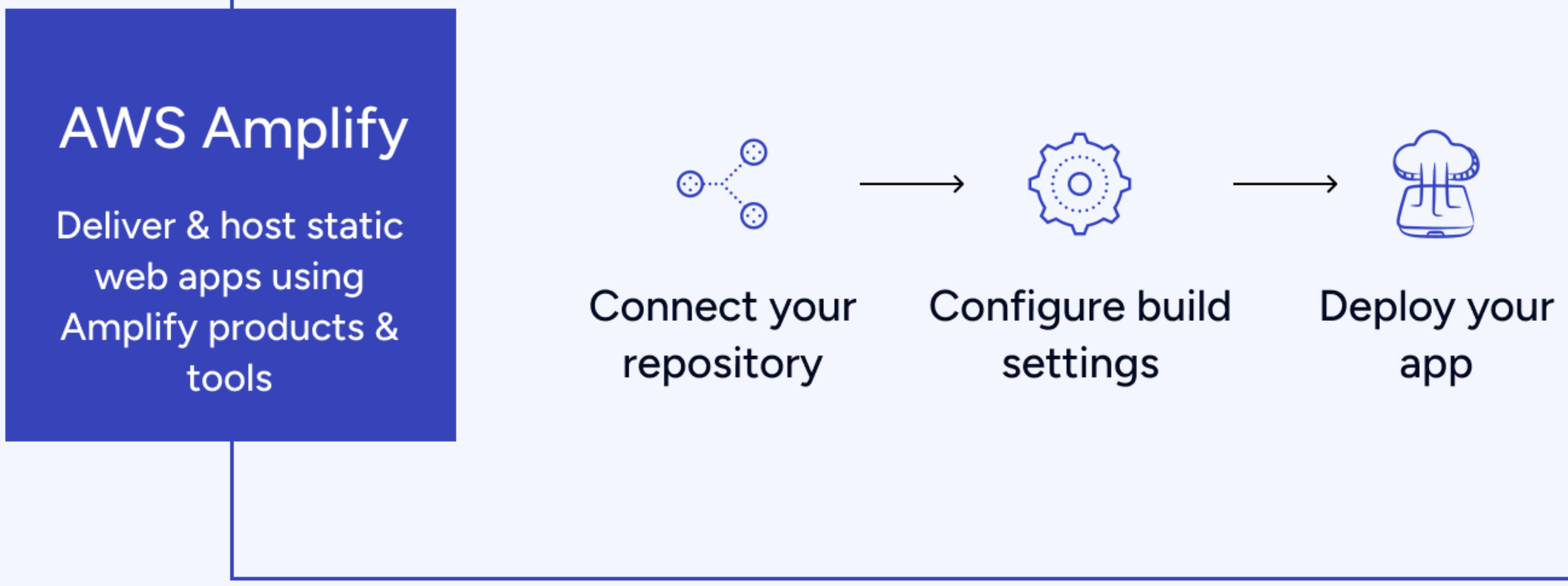
Serverless Approach with AWS Lambda

- Pros: Pay-per-use, automatic scaling.
- Cons: Limited execution time, cold start latency.



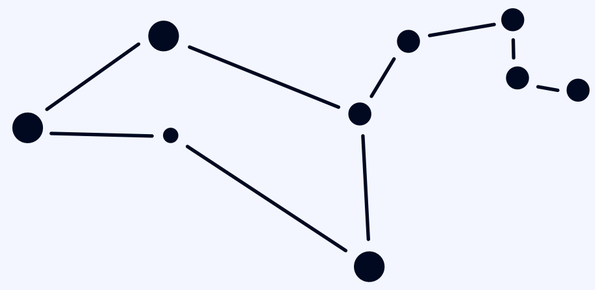
AWS Amplify for Simplified Deployment

- Pros: Simplified deployment for startups and MVPs.
- Cons: May not be suitable for complex projects.



Best Practices:

- Use AWS IAM roles for security.
- Implement CI/CD pipelines for automation.
- Leverage AWS CloudFormation for infrastructure as code.
- Monitor performance using AWS CloudWatch.
- Implement HTTPS and security measures.



gart.

If you like the content

Share
Like
Comment

gart.