Codelepant MVP

Context

Codelephant is a product to help developers measure their code quality! (trivia: elephants have the best sense of smell so we strive to be as good at sniffing code as them)

Product requirements

Functional requirements

F1. User can login to our system

F2. User can add/delete/modify repositories

F3. User can trigger code analysis on changes to repo

F4. User can view analysis results on FE

Non-functional requirements

NF1. Should be scalable up to 100,000 users

NF2. Communications should be done via a central API gateway interfacing the front-end

NF3. Completed jobs should have their downloaded code cleaned up

System design

- The list of services required are:
    - Frontend service
        - Serves frontend application to user
    - User service
        - Authenticates users
        - Stores user information
    - Repo service
        - Stores repository information
    - Code quality service
        - Triggers code analysis for a chosen repo, branch, commit
        - Stores analysis results
    - Webhook service
        - Receives github/gitlab webhook and creates a new code quality job
    - Repopuller service
        - Pulls source code given a repo URL, branch and commit and stores it
        - Uses a queue to schedule asynchronous task of pulling source code
        - Stores source code metadata
    - Code analysis worker service
        - Performs analysis on a codebase and return results
        - Uses a queue to schedule asynchronous task of analysing code

Milestones

Start date: 22 May 2023

Targeted launch date: 17 Jul 2023

Total sprints required: 4

Assumptions:

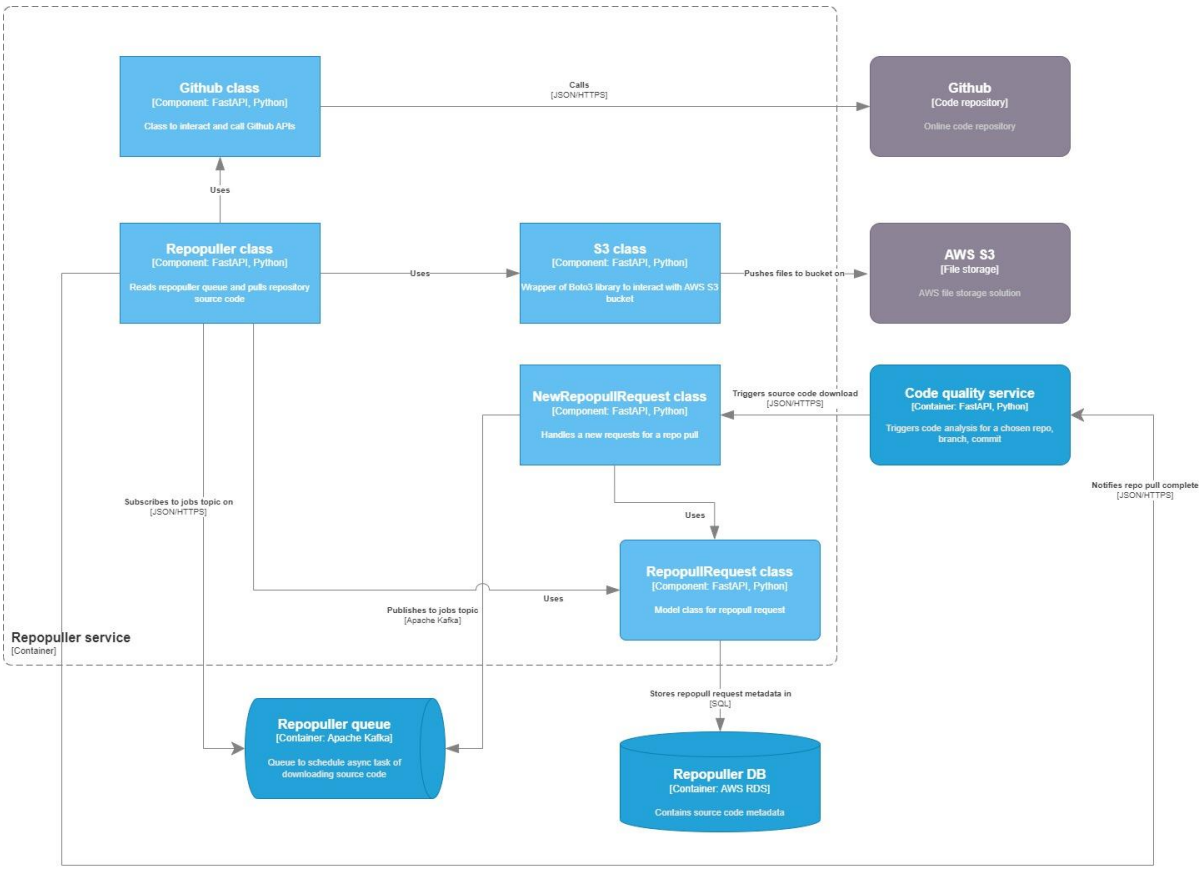
1. Each service requires 2 weeks of effort from 1 team (a sprint)
2. Each sprint covers the planning, execution, QA
3. A buffer of 1 sprint is allocated for high-priority fixes and QA before official launch
4. Each service is independent and there are no dependencies (unlikely but I'm assuming this to simplify things)

Sprints:

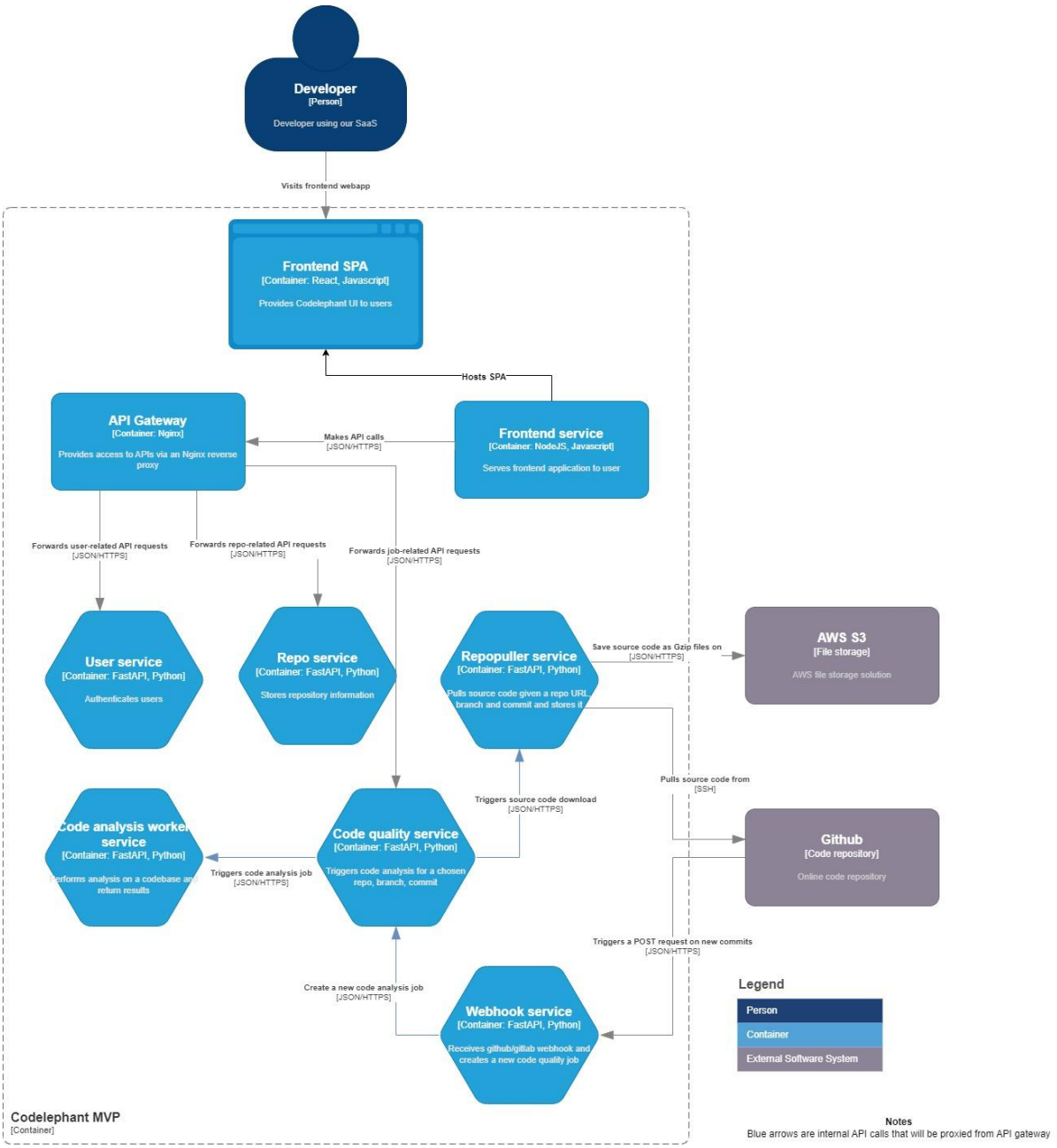
Team	Sprint	Goal	Start date	End date	Milestones
Team A	1	• Setup API gateway • Refactor POC backend and frontend to communicate using API gateway	2023-05-22	2023-06-02	NF2 achieved
Team B	1	• Setup User service	2023-05-22	2023-06-02	F1 achieved
Team A	2	• Setup Repo service	2023-06-05	2023-06-16	F2 achieved
Team B	2	• Setup Webhook service	2023-06-05	2023-06-16	F3 achieved
Team A	3	• Setup Repopuller service	2023-06-19	2023-06-30	
Team B	3	• Setup Code quality service	2023-06-19	2023-06-30	
Team A	4	• Setup Code analysis worker service	2023-07-03	2023-07-14	F4 achieved
Team B	4	• Buffer sprint for fixes	2023-07-03	2023-07-14	

Component diagram

Use case: Code quality service triggers a new repo pull request to download source code


Github class
[Component: FastAPI, Python]
Class to interact and call Github APIs
Calls
[JSON/HTTPS]
Github
[Code repository]
Online code repository
Uses
Repopuller class
[Component: FastAPI, Python]
Reads repopuller queue and pulls repository source code
Uses
S3 class
[Component: FastAPI, Python]
Wrapper of Boto3 library to interact with AWS S3 bucket
Pushes files to bucket on
AWS S3
[File storage]
AWS file storage solution
NewRepopullRequest class
[Component: FastAPI, Python]
Handles a new requests for a repo pull
Triggers source code download
[JSON/HTTPS]
Code quality service
[Container: FastAPI, Python]
Triggers code analysis for a chosen repo, branch, commit
Notifies repo pull complete
[JSON/HTTPS]
Subscribes to jobs topic on
[JSON/HTTPS]
Uses
RepopullRequest class
[Component: FastAPI, Python]
Model class for repopull request
Uses
Publishes to jobs topic
[Apache Kafka]
Repopuller service
[Container]
Stores repopull request metadata in
[SQL]
Repopuller queue
[Container: Apache Kafka]
Queue to schedule async task of downloading source code
Repopuller DB
[Container: AWS RDS]
Contains source code metadata

Container diagram



Architectural diagram