QBurst

CI-CD Pipeline Adoption Using

AZURE DEVOPS

Project Overview

The client's flagship Property Management application gives property owners the ability to initiate service and amenity requests with quick access to payments, title deeds, and buyer registration. However, their software development life cycle process incurred release cycles that took several months to complete. The client wanted to improve their code integration and build delivery mechanisms to optimize productivity.

Client Profile

Our client is a leading property development company based in the Middle East. The company provides property development and management services across the Middle East, Asia, Europe, and North America.

Business Challenges

- Conflict resolution due to delayed code integration cost the development team a significant amount of time
- Creating builds for different environments was another challenge
- The error-prone manual process involved multiple steps and led to mismatch in build number, environment, and certificate

QBurst Solutions

We proposed a continuous integration, delivery, and deployment strategy for the project using Azure DevOps. Collectively known as CI/CD, continuous integration, delivery, and deployment form an integral part of modern development intended to reduce errors during integration and deployment while stepping up project velocity. The CI/CD philosophy and practices are augmented by robust tooling capabilities that automate testing at each stage of the software pipeline. By incorporating these practices, we were able to

reduce time required to integrate changes and test each change before moving into production.

Azure DevOps was selected as CI/CD platform because of the following factors:

- Highly configurable pipeline
- Availability of different cloud-based Mac machines with multiple Xcode versions
- Availability of third-party plugins for integrations
- Azure DevOps was already being used for project management

Processes followed as part of each Pull Request (PR):

- Static Code Analysis – using Sonar Cloud (changes only)
- Code Compilation – on the incoming branch
- Unit Testing – on the incoming branch

Each merge to develop branch triggers the following:

- Static Code Analysis – using Sonar Cloud (full)
- Unit Testing – develop branch
- Build and sign – develop branch using Fastlane
- Upload artifacts – for developer testing using Fastlane

Each merge to QA branch triggers the following:

- Update QA configuration
- Build and sign – develop branch using Fastlane

- Share the build with QA team
- Deploy application to QA environment using Fastlane

Each merge to staging branch triggers the following:

- Update staging configuration
- Build and sign – develop branch using Fastlane
- Deploy application to staging environment using Fastlane
- Share the build with staging team for Regression Testing and UAT Testing

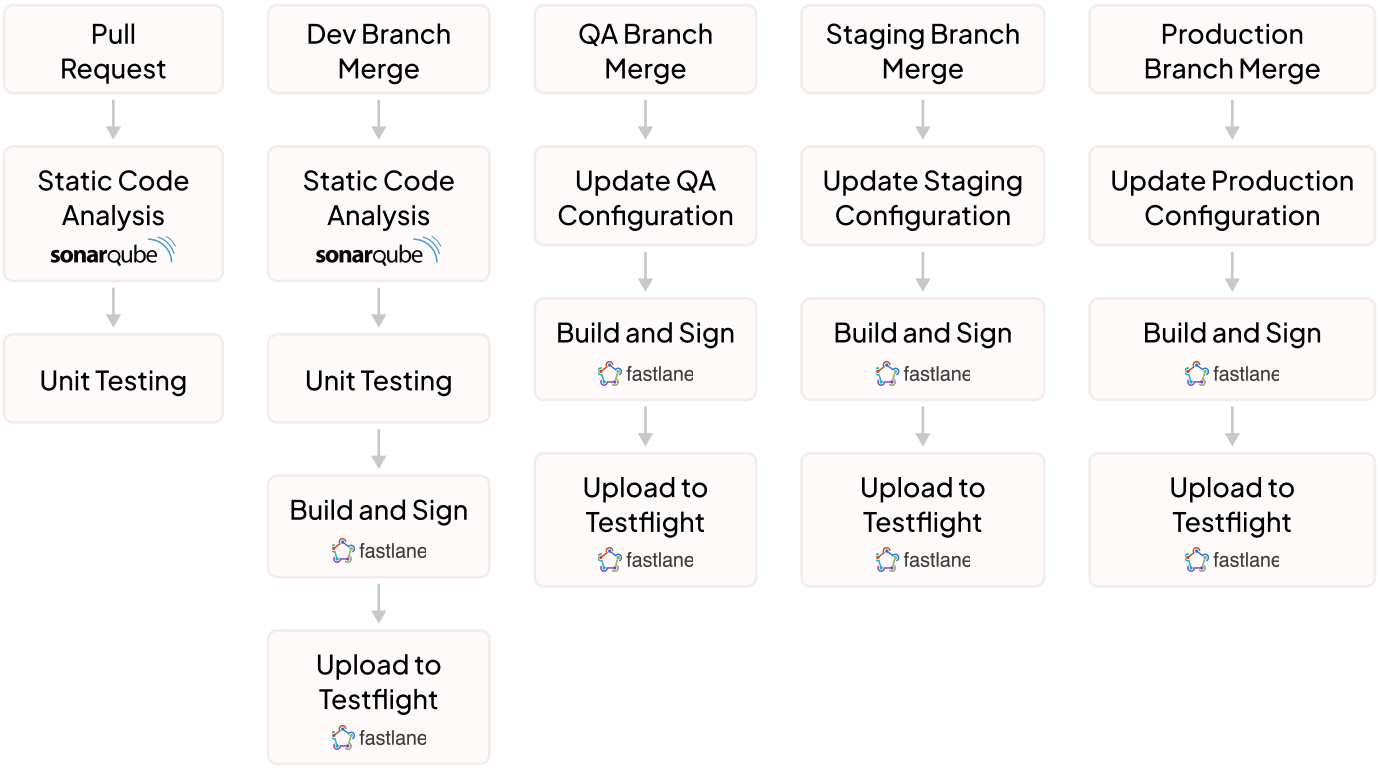
Each merge to production branch triggers the following:

- Update production configuration
- Build and sign – develop branch using Fastlane
- Deploy application to production environment using Fastlane
- Share the build with production team for Smoke testing

Implementing CI/CD pipeline ensured consistent and quality code. The Azure Pipeline also provided a quick, easy, and safe way to automate builds, making them ready for use.

Tools and Technologies

IOS SDK	SonarQube	Fastlane



Business Benefits

- Faster and reliable software delivery with value addition in terms of delivery quality and process efficiency
- Significantly reduced effort by the development team in generating builds
- Reduced dependency between QA team and development team with QA team also generating builds; hence reduced delay
- Reduced number of errors as a result of automation
- Faster deployment and configuration of resources in a reliable and repeatable manner