INTERNAL

HighWire Release & Deployment Management

FUNDS AXIS

Policy title:	HighWire Release & Deployment Management

Issue	1.2
Approved by:	Darren Burrows
Approval Date:	March 2024
Next Review Date:	March 2025

Scope:	The policy applies to Funds-Axis Group and all contractors and other people working on behalf of the company.
Responsibility for Implementation & Training:	Day to day responsibility for implementation: ISO Day to day responsibility for training: ISO

Distribution methods:	Methods used to communicate this policy: • Information Security Training Module

FUNDS AXIS

CONTENTS

1. Introduction

1.1 About HighWire

HighWire is a cloud-based technology solution comprising multiple modules accessible to users as a unified platform. Developed by Funds-Axis, it harnesses the capabilities of the AWS cloud platform to deliver a comprehensive, plug-and-play solution with rapid implementation capabilities.

1.2 Purpose

This document outlines the Release and Deployment process for the HighWire Product Platform. It describes the technology tools, architecture, and development processes used to ensure efficient, secure, and controlled engineering practices.

2. Release Management and Deployment Process

2.1 Overview

HighWire development adheres to an automated DevOps process for release and deployment. The application architecture follows an API-First approach, with each feature implemented as an API on the backend, loosely coupled with front-end modules to provide a cohesive solution.

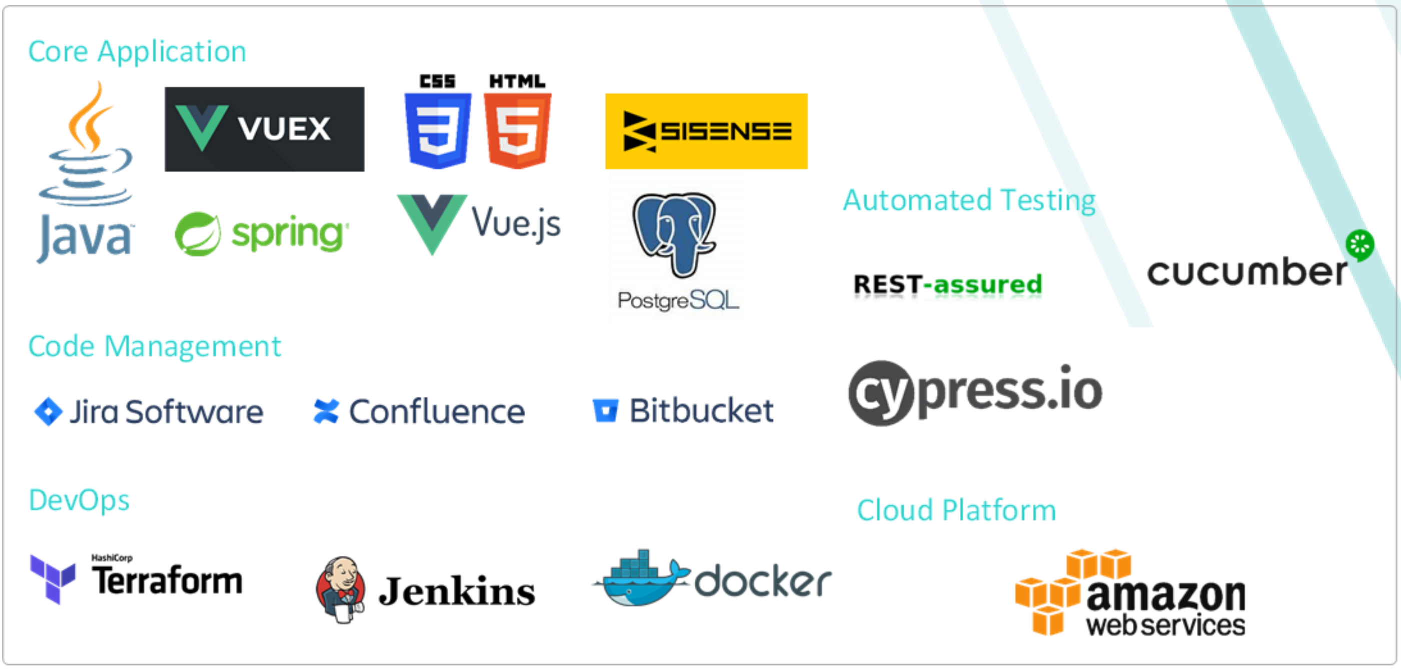
Infrastructure as code principles are strictly followed for application deployment and maintenance, with all AWS resources provisioned using Terraform scripts.

2.2 Architecture Overview and Technology Platform

HighWire is hosted on AWS, leveraging best practices outlined in the AWS Well-Architected Framework. Core application services are developed using Java and the Spring Boot framework for the backend, while Vue.js is utilised for rich, interactive front-end interfaces. PostgreSQL serves as the persistence layer.

Agile methodology drives product development, with Jira facilitating backlog management, sprint planning, and tracking. Integrated code management via Jira and Bitbucket enables traceability of code changes to corresponding stories.

Automated testing is integral to development, employing Rest-Assured for backend and API testing, and Cypress for UI testing. Infrastructure and deployments are managed through Terraform, with applications containerised using Docker and deployed on AWS. With this IaaS implementation, environment creation and deletion are automated.



2.3 Release Planning

Release planning and development follows Agile methodologies, with backlogs created as stories in Jira. Product owners create these stories based on various sources such as existing product enhancements, customer feedback during demos, and other factors. Once created, these are elaborated and refined by Product owners and development team. Stories are groomed and selected for development in each sprint.

2.4 Source Code Management

Bitbucket integrated with Jira and Confluence serves as the source code management platform. Development adopts a modular approach, with separate repositories and pipelines for various components.

Standard GitHub flow is followed for release management, with each requirement tracked as a story in Jira. Every requirement is created as a story for development in Jira along with acceptance criteria and scenarios (feature file).

For development of feature, developers create a feature branch from master branch. Junit and Mockito are used for unit testing. API testing implemented using Rest Assured and UI testing is implemented using Cypress. On completion of feature, it is merged with Master branch after feature implemented is test and the code reviewed.

2.5 Environment Management

Funds-Axis manages multiple environments throughout the product development lifecycle:

Environments	Purpose and Access
Development	• Environments used by development team. • Each developer carries out development in their local environment. • Docker images used for testing. • Development environment used as integration testing environment
Test	• Controlled environment • Automation Testing along with Regression testing. • Manual Sanity Checks • Development team has read-only access
Staging	• Environment to validate production issues / defects. • Used to carry out product demos. • No development or testing in this environment. • Maintained as replica of Production environment.
Production	• Production environment • No access to Funds-Axis development team

2.6 Product testing

HighWire development embraces Test-Driven Development (TDD) and Behaviour-Driven Development (BDD) approaches:

- Each story is developed along with Junit test cases for unit level testing.
- Mockito framework is used for mocking classes and objects.
- Product development follows “API-First” design approach.
- Each API developed along with end-to-end API automated test cases.
- Rest Assured framework is utilized to automate API testing.
- Feature files and scenarios form the input to test case implementation.
- For front-end (UI testing) automation, we use Cypress Framework.
- Apart from these automated testing, exploratory manual testing is carried out for each story.
- On completion automated and manual testing; stories are released to stage and production.

2.7 Release schedule

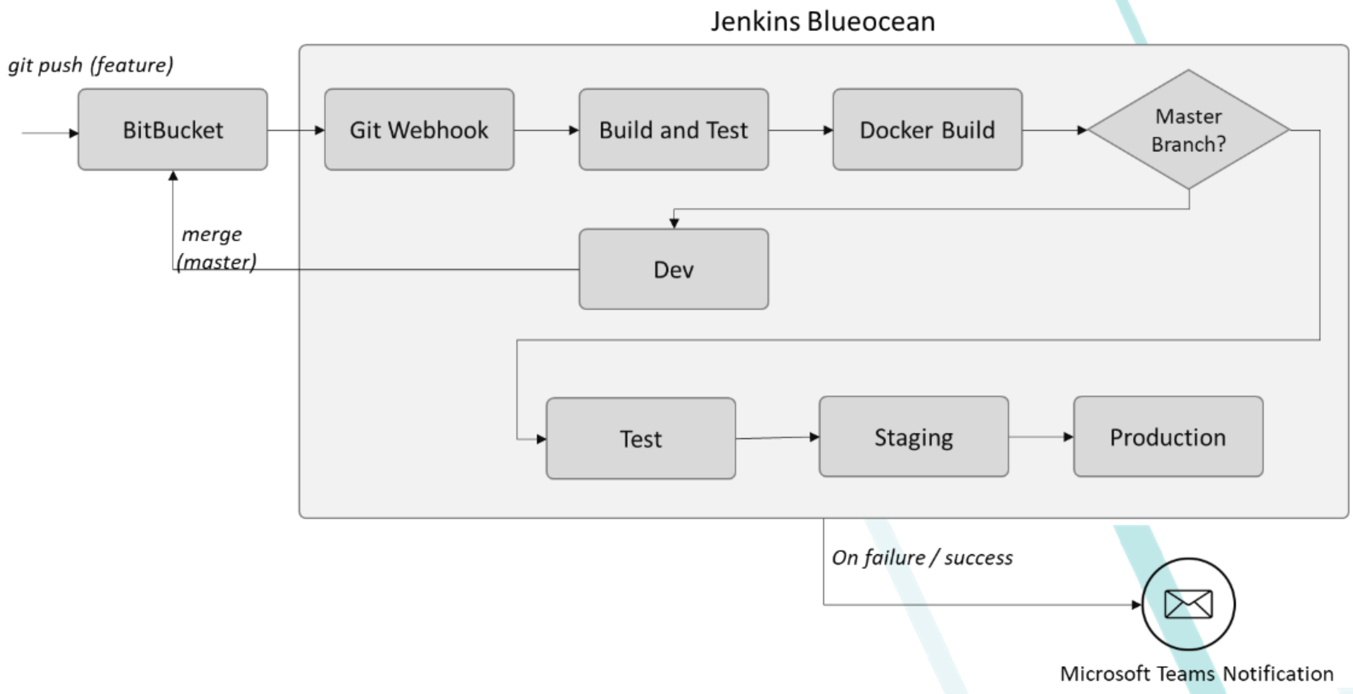
Continuous Integration and Continuous Deployment (CI/CD) practices are adopted for product releases:

- Once the story is approached in Testing stage, CI/CD pipeline is triggered to deploy story to staging and Production.
- While staging environment update is immediate, update to product environment is control.
- All release updates to production environment happens after business-hours. This is to avoid any disruption to live environments.
- Exception to above process is hot-fix required to production environment.

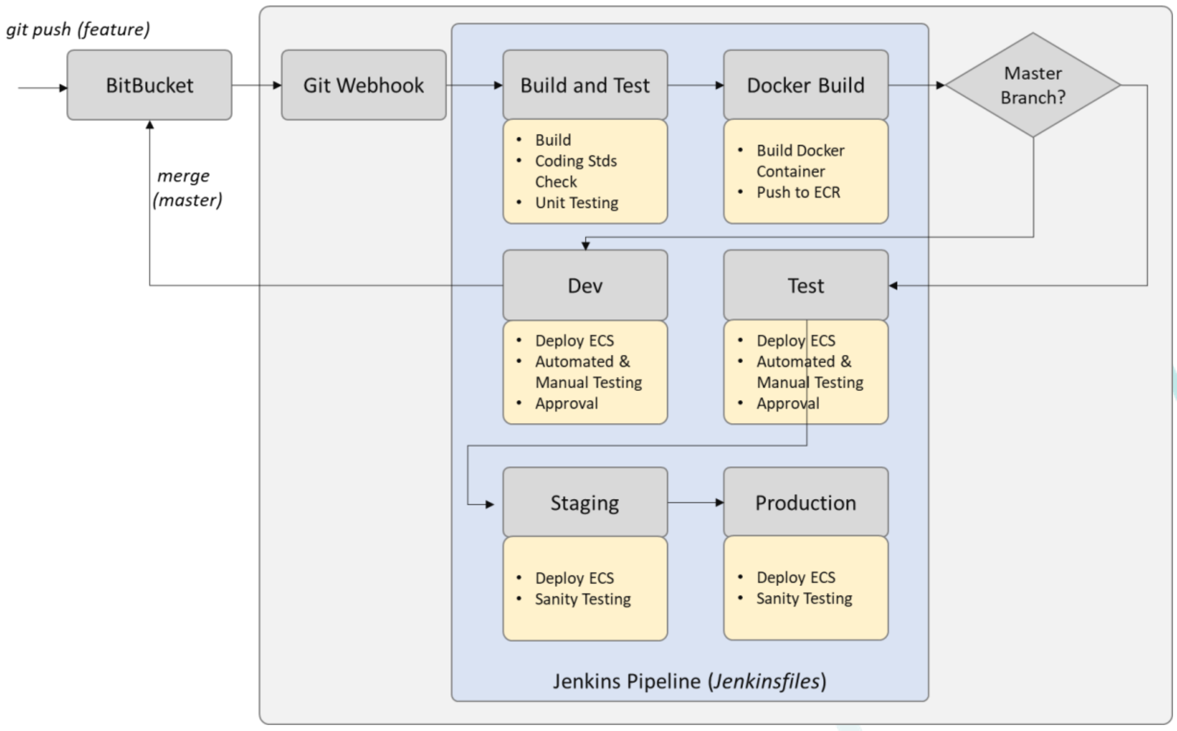
3. Continuous Delivery

3.1 Pipeline Overview

Delivery pipelines are triggered by Git events, initiating automated build, testing, and deployment processes. Jenkins orchestrates the deployment pipeline, conducting code style verifications, unit testing, and packaging of Docker images:



- Delivery pipeline is triggered on following events:
  - Git push to feature branch.
  - Merge to master branch.
- On receipt of Git events, Jenkins trigger deployment process.
- Build and automation tests are carried out:
  - The application is built using the "mvn compile" command.
  - Checkstyle is used to perform code style verifications.
  - The mvn test command is utilized for automated unit testing of the application.
- This is followed by packaging docker images by initiating Dockers:
  - Build docker images using docker build.
  - Push docker image to repository. HighWire utilises AWS ECR as image repository.
- On merge to master branch:
  - Build is deployed to Test environment, where all the automated testing including regression testing is carried out.
  - Build is then deployed to Stage and Production environment.
- On failure in any stage of the process; notification is sent using Microsoft teams.



3.2 Deployment Process using Jenkins.

- Process for deployment from the master branch.
- Login to Jenkins to initiate deployment process. It is also triggered on merged to Master Branch
- Click on 'Ops' Tab and select 'semaprod' from the list. We can view 'Build History' with the latest build details.
- Once we click on the 'Build with Parameters' option, screen will be navigated to the Project semaprod page.
- Build with Parameters has two modes:
- GO - This Mode will deploy all the builds till Production Environment.
- NOPROD - This Mode will deploy the code till Stage Environment only. For deploying the code till Production, the mode must be set as GO and then the build must be started.
- HighWire Application Relevant jobs
- Backend: HighWire
- Frontend: HighWire-client
- 'Master' branch is used for Production deployment.
- Click on the 'Open Blue Ocean' to get the list of Feature Branches.
- Click on the 'Master' Branch to view the current build status.
- Pipelines consist of all the test and deployment status and shows the currently running tasks or the deployed status.
- Frontend Pipeline with the deployment status can be view by clicking on the HighWire-client Job and selecting the Master Branch.

END