**Internship Performance Review**

**Executive Summary**

**During my internship, I focused on critical DevOps infrastructure migration projects, successfully transitioning multiple workflows from Quick Build to GitHub Actions. I also developed specialized Docker solutions that significantly improved build efficiency, and demonstrated scripting expertise by migrating and adapting Groovy and PowerShell scripts. Additionally, I participated in innovate48 and am currently working on several initiatives to enhance team monitoring capabilities and infrastructure management for Virtual Machine .**

**My key contributions include:**

1. **Successfully migrating multiple workflows from Quick Build to GitHub Actions**
2. **Developing optimized Docker (Image) solutions that significantly reduced build times and improved consistency**
3. **Participating in (Innovate48 with Seniors ) innovation projects that enhance team capabilities and documentation access in AI**
4. **Reusing and migrating Groovy scripts from Quick Build to GitHub Actions, including create\_version\_file.groovy, post\_exe\_get\_artifactory.groovy, and revlog-git.groovy**
5. **Currently implementing monitoring solutions for improved resource management for Virtual Machine (Disk Storage)**
6. **Completed all assigned professional development courses on weLearn platform as required by HR**

**Technical Training Completed**

Throughout my internship, I invested in developing expertise in critical DevOps technologies:

* **GitHub Actions**: Mastered workflow automation, CI/CD pipeline configuration, and enterprise integration patterns
* **Docker**: Developed advanced containerization skills including image optimization, multi-stage builds, and enterprise deployment
* **DevOps**: Applied infrastructure-as-code principles, automated testing frameworks, and continuous deployment techniques
* **Quick Build**: Gained proficiency in legacy build system architecture which is currently Fis is using for workflows and Builds
* **GitHub**: Implemented enterprise-grade repository management and collaboration workflows
* **JFROG:** To store the antifactory or Builds
* **Scripting: Developed and migrated Groovy and Python scripts from Quick Build to GitHub Actions, including critical scripts like create\_version\_file.groovy, post\_exe\_get\_artifactory.groovy, and revlog-git.groovy**
* **Key Projects & Achievements**

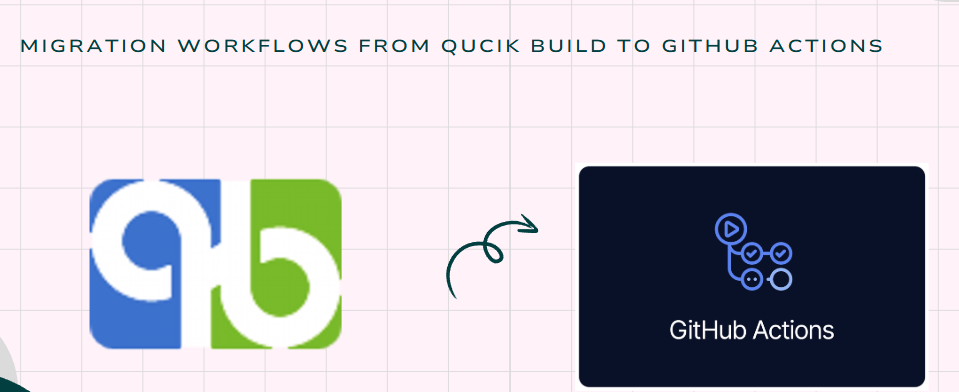
**1. GitHub Actions Migration**

**Led the strategic migration of critical build workflows from Quick Build to GitHub Actions, resulting in:**

* **Seamless Transitions: Successfully executed live migrations with minimal disruption to development teams**
* **Enhanced Automation: Implemented cloud-based automation replacing previously manual build processes**

**Successfully Migrated Components:**

* **ConvertFDB**
* **AMSC**
* **AIMS**
* **RAMSA**
* **BPR**

****

**Technical Implementation Highlights:**

**For each component, I carefully analyzed the existing Quick Build workflows and recreated them in GitHub Actions with appropriate optimizations.**

**For example:**

**QB workflow of ConvertFDB :**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Migrated to Github actions (ConvertFDB)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Builds successfully pushed to Artifactory (JFROG):**

**As you can see all builds of ConvertFDB is pushed in Artifactory**

**A screenshot of a computer

AI-generated content may be incorrect.**

**MEGA\_BPR Project:**

**This workflow consisted of 5 major workflows:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Component workflows: BPR, AMSC, RAMSA, AIMS**

**A screenshot of a computer

AI-generated content may be incorrect.**

* 1. **Original AMSC workflow:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Migrated : AMSC GitHub Actions workflow:**

**A screenshot of a computer

AI-generated content may be incorrect.**

* 1. **Original RAMSA workflow:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Migrated to GitHub actions : RAMSA workflow:**

**A screenshot of a computer

AI-generated content may be incorrect.**

* 1. **AIMS and BPR Status:**

**AIMS workflow (in progress): Its on process figuring out parallel Build in one time**

**QB:**

****

**GitHub actions workflow :**

**A screenshot of a computer

AI-generated content may be incorrect.**

**BPR workflow (pending access resolution):**

**BPR : it will get solved when we get access to copy the files   
  
A screenshot of a computer

AI-generated content may be incorrect.**

**Once we get the access to the file I will be able to migrate BPR successfully and it will initiate the xcopy command too.**

**Validation of Successful Migration:**

**All builds verified in Artifactory:**

**As you can see All builds are generated in Artifactory i.e. BPR , AMSC , RAMSA and ConvertFDB**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2. Script Migration Achievement:**

**A critical aspect of this migration involved adapting and migrating essential Groovy scripts from Quick Build to GitHub Actions:**

* **create\_version\_file.groovy: Modified to work with GitHub Actions environment variables and file system structure**
* **post\_exe\_get\_artifactory.groovy: Adapted to handle artifact processing in GitHub Actions context**
* **revlog-git.groovy: Updated to integrate with GitHub's version control environment**

**These script migrations ensured consistent functionality while leveraging the advantages of the GitHub Actions platform.**

**As you can see in image this is my Github Repo where all Scripts are stored and we give path of each script in the githubworkflows   
  
As you can see I have made 3 scripts which are used in my workflows**

* **create\_version\_file.groovy**
* **post\_exe\_get\_artifactory.groovy**
* **revlog-git.groovy**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Some Script which I have created to migrate in GitHub actions**

**For Example : post\_exe\_get\_artifactory.groovy**

**Made and migrated various Groovy Scripts**

**This Groovy script simulates a QuickBuild environment, checks for a code signing certificate, extracts and formats JSON data from a bom.json file, creates environment variables based on this data, and optionally executes a batch script (get\_artifacts\_FIS.bat) if it exists in the specified directory**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**3. Docker Optimization Project Saved (20 min)**

Developed specialized Docker solutions that delivered significant efficiency improvements:

* Created a customized Visual Studio Version Docker image that **reduced build times by 20 minutes per build**

**Technical Achievement:**

**Before optimization:** **Visual Studio installation process**

A screenshot of a computer

AI-generated content may be incorrect.

To streamline the build process, we first uninstall the latest version of Visual Studio and install the specific required version. Previously, this setup took around 20 minutes per build. By optimizing the workflow, I have successfully reduced this time, ensuring faster execution and greater efficiency.

**AFTER optimization:** **Visual Studio installation process**

A computer screen shot of a computer screen

AI-generated content may be incorrect.

After implementing the fix, I created a Docker image with the specific Visual Studio version needed. Now, instead of reinstalling the version each time, we can directly pull the pre-configured image into our workflow. This optimization has reduced build time from **20 minutes to just 5 minutes**, making the process more efficient and cost-effective.

**How cost effective ?**

**”If each minute costs $1, running a GitHub-hosted runner for 20 minutes would cost $20. If we trigger this workflow 10 times, the total cost would be $200. This approach is not cost-effective. So we reduced 20 min to 5 min which help to costcut for each build and workflow**

**Additionally, we can enhance this image by including essential tools like**

Python, Java, Ant, and more. When we pull the image, all required dependencies will be installed automatically, further minimizing setup time and ensuring a smoother workflow. This approach simplifies maintenance, accelerates execution, and optimizes resource usage.

**DOCKER Image pushed to docker hub:**

A screenshot of a computer

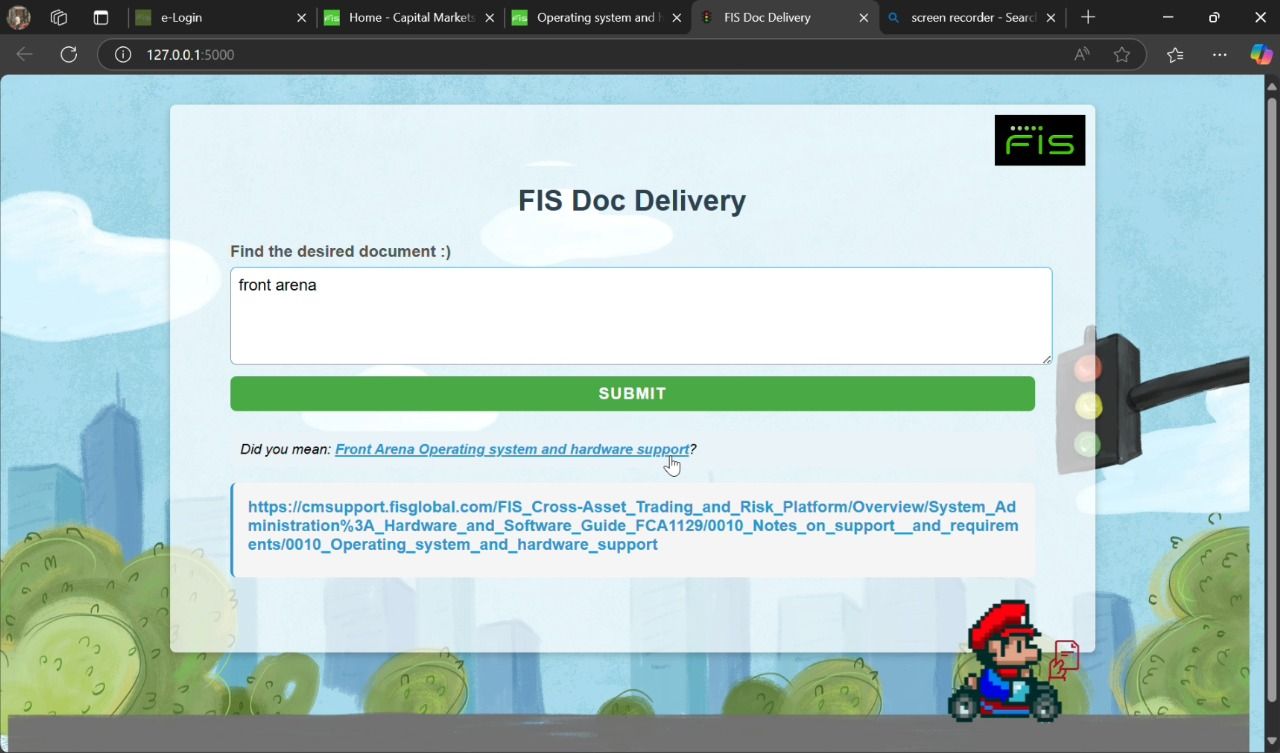
AI-generated content may be incorrect.

The Docker image has been successfully pushed to Docker Hub, providing a convenient way to integrate it directly into our workflow. This eliminates the need for manual uploads and ensures a more efficient, streamlined process. By pulling the image directly from Docker Hub, we can improve deployment speed, maintain consistency, and simplify our overall workflow. This makes the entire process more seamless and effective

* 1. **Innovation48 Initiative**

Collaborated with **Senior engineers(Current Team)** to create a documentation chatbot for FIS that addressed a significant organizational challenge:

I built a "Documentation-Finding Chatbot" to solve a very real challenge:  
 At FIS, with thousands of internal documents scattered across systems, finding the exact one you need is often time-consuming.



* Dramatically reduced time to find exact documentation (from minutes to seconds)
* Improved developer productivity through instant access to technical information
* Leveraged modern AI techniques to enhance knowledge retrieval capabilities

**Technical Implementation:**

**Built using Python, Flask, HTML, CSS, and React, with advanced ML models including:**

* **NLP with TF-IDF (Term Frequency-Inverse Document Frequency)**
* **TextBlob for sentiment analysis**
* **FuzzyWuzzy for approximate string matching to handle typos**

**This solution allows users to input a single prompt and immediately access the needed documentation, eliminating time-consuming searches across thousands of internal documents**

* 1. **Current Projects Assigned :**

I am actively working on several initiatives to increase **team efficiency**:

**VM Resource Management Dashboard**

**My manager has assigned me full ownership of developing a centralized dashboard that will:**

* **Visualize all VMs and their remaining capacity**
* **Implement monitoring alerts to proactively identify resource constraints**
* **Create reporting features to support capacity planning**

***Will Use TechStack*** :  
Front end : React , html , CSS, TailwindCSS , Backend : nodejs or Spring board JAVA   
and powershell for scripts to give VM ware details or python

This solution will address the current challenge of monitoring space allocation across multiple virtual machines, providing a unified interface to view capacity and free space on each disk across all machines.

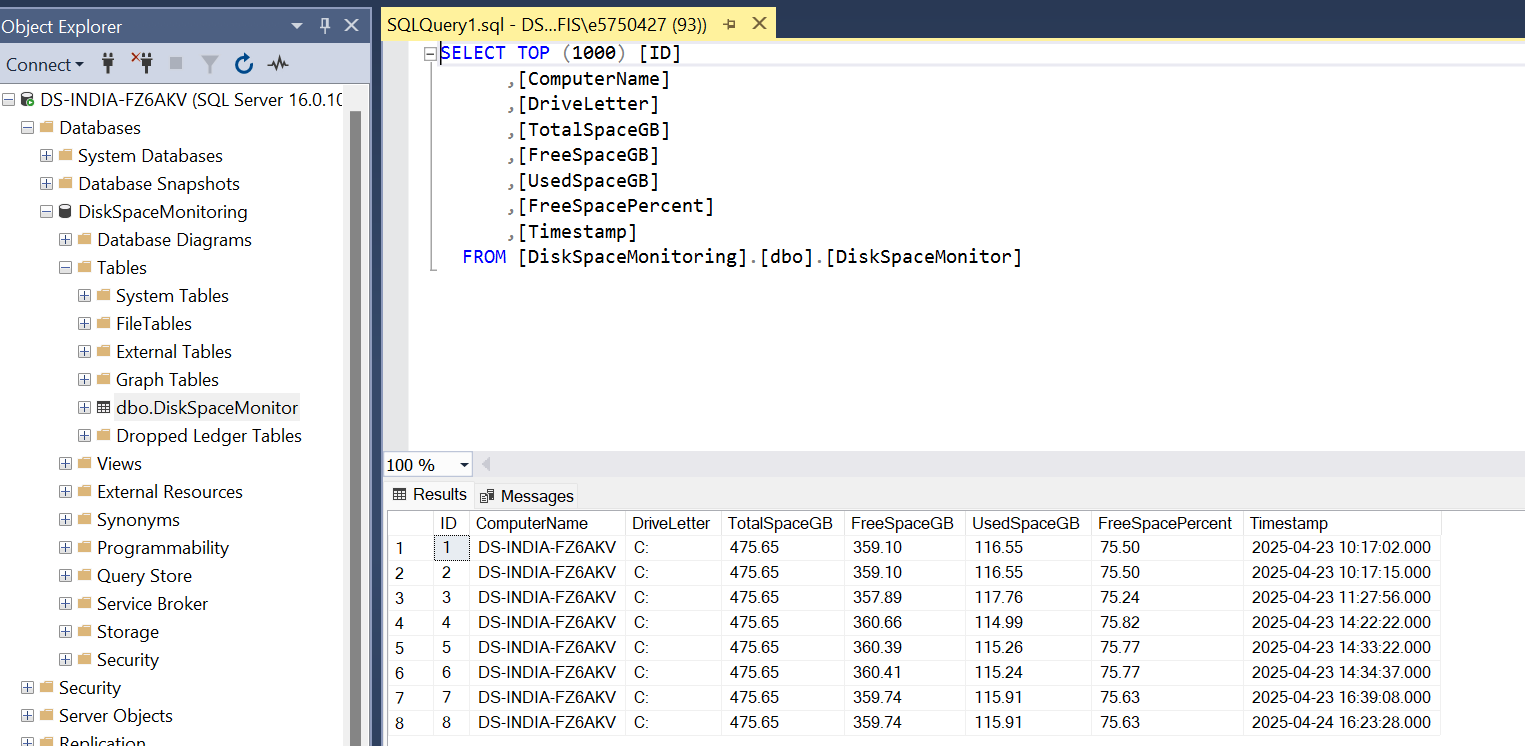
**Infrastructure Management Improvements**

Additionally, I am working on:

* Addressing certificate management issues to improve system security
* Streamlining ticket management processes through the Snow Ticket

A screenshot of a computer

AI-generated content may be incorrect.



6 . Done with HR Assigned WeLearn Courses

A screenshot of a computer

AI-generated content may be incorrect.