Alternative Tools Research

1. Find a new CI tool to analyze. Find a new Real-Time Error Monitoring tool to analyze.
    a. CI Tool
        i. **Harness** - an integrated CI/CD tool
    b. RTEM Tool - **Dynatrace -** a unified monitoring tool
2. Analyze the value added by both tools - why should we use it?
    a. CI - **Harness**
        i. Integrated CI/CD product
            1. Since this is an integrated product, development teams need to only learn how to use one tool, reducing total training time
        ii. Visual and easy to navigate tool
            1. Many of the following features will highlight the usability of this tool coupled with how necessary information is readily available
        iii. Cost savings
            1. Save on costs throughout the CI/CD process, including cloud costs
        iv. Cloud Cost-Saving Management
            1. Cost Transparency - you will be able to see and easily find the following
                a. Deep Kubernetes visibility
                b. Cost Perspectives - receive periodic reports on cost and usage metrics that we define. Easily set up budgets and spending forecasts and then easily monitor them
        v. IDE autocomplete and schematization - provides an easier to use experience for developers, saving time while coding out YAML files, for instance
        vi. Test intelligence - utilize machine learning to test more efficiently, significantly reducing time to test
            1. Tests code only affected by the changes in the code, therefore only running necessary tests - can reduce test runs by over 90%
            2. Test more frequently - this gives developers the ability to test more often to receive the necessary feedback regarding code changes. Developers will be able to code more quickly, given the confidence that the changes made are effective and working
            3. Reduction in the overall deployment process - this one feature allows teams to move at an accelerated rate, from production to deployment, therefore greatly reducing the total time to deploy and the cost associated.
    b. RTEM - Dynatrace
        i. One solution so one tool for the DevOps team to learn
            1. Application Performance
                a. Cross-team Collaboration
                    i. An easy to use tool for all developers

            b. Utilize user experience feedback tools and perform business analytics with built-in customization
                i. Establish and track KPIs and make them visible to the entire team
                    1. Self-accountability
            c. Automatic monitoring - allow the AI to do the monitoring so the developers can focus on other priorities such as knocking down tech debt
        2. Infrastructure monitoring
            a. Intelligent Observability
                i. A wealth of options to improve observability - AI-driven
        3. AI Ops
        4. Digital Experience
        5. Digital Business Analytics

3. Evaluate documentation for both tools, specifically regarding set up and any “getting started” sections
    a. CI - Documentation is presented in a friendly, easy-to-use way. Not at all intimidating. It also appears to be thorough. There is a nice section even explaining how to use the documentation.
    b. RTEM - Documentation is not only informational, it is vast and well organized.
        i. Easy to navigate and very detailed, step-by-step implementation and usage instructions
4. Summarize the maturity and market share of each tool
    a. CI - Harness has been operational since 2012 - it is considered a mature product. Market share is hard to determine. However, Harness supports many notable brands and is seeing unprecedented growth as more and more dev teams seek out cloud-based SaaS CI/CD tools - a 400% revenue increase, year-over-year from 2019 to 2020, demonstrates this growth.
    b. RTEM - Dynatrace has an 8.8% market share - this is a significant presence. Further DT is growing.

Runtime Analysis

- extraLargeArray
    - DoublerAppend - 4.0389ms
    - DoublerInsert - 1.0422726s
- largeArray
    - DoublerAppend - 664.2 μs
    - DoublerInsert - 9.1534 ms
- mediumArray
    - DoublerAppend - 179 μs
    - DoublerInsert - 194.1 μs
- smallArray

    - DoublerAppend - 147.9 μs
    - DoublerInsert - 52.8 μs
- tinyArray
    - DoublerAppend - 106.3 μs
    - DoublerInsert - 39.3 μs

Size of Array	.Push time	.Unshift time
Extra Large - 100000	4.0389ms	1.0422726s
Large - 10000	664.2 μs	9.1534 ms
Medium - 1000	179 μs	194.1 μs
Small - 100	147.9 μs	52.8 μs
Tiny - 10	106.3 μs	39.3 μs

Units in Time:

- μs - one-millionth of a second
- ms - one-thousandth of a second
- s - one second

Results Summary:

- Scalability:
    - .push - is the better method to use from a scalability perspective as it significantly outperforms .unshift as the size of the array continues to grow.
        - .unshift performs better up to large when the time complexity for .unshift begins to demonstrate its poorer performance against .push.