Step 1: **Alternative Tools Research**

1. Do some internet research and find two new tools to explore - one for Continuous Integration, and one for Real Time Error Monitoring. You cannot use the ones you used earlier this week.

- Jenkins in several reviews is one of the top open-source projects for automation with thousands of plugins to choose from.
- Raygun Error Monitoring is a powerful, user-centric tool that gives actional insights into errors and crashes impacting users. Monitors full-tech stack in real-time.

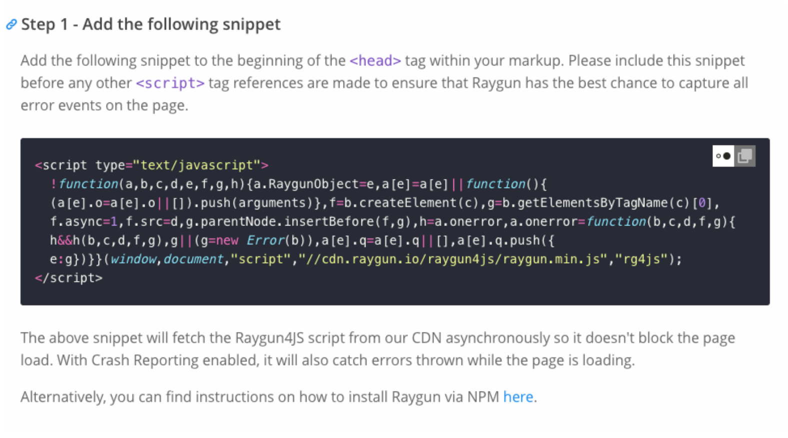
2. Jenkins Continuous Integration and Raygun Error Monitoring

- Jenkins
    - Works as a stand-alone CI server, or you can turn it into a continuous delivery platform for virtually any of your projects
    - Pre-built packages for Unix, Windows and OS X ensures an easy installation process
    - A web interface that can be used to quickly configure your server.
    - Customer plugins for build and source code management, administrative tasks, user interface, and platforms
    - Deployable across a network of machines, improving performance of builds and tests
    - Large community with leading software brands involved in development

- Raygun
    - Complete visibility throughout your entire tech stack
    - Support for all major programming languages and frameworks including .NET, Javascript, PHP, Ruby and more
    - Simple on-demand pricing from $4 per month with unlimited users
    - Monitor every deployment in real-time to quickly identify errors before they impact your users
    - See the full stack trace, environment, browser, version, class name, host and more
    - Connect to source code repositories using native integrations with Github, Bitbucket, and GitLab with support for source maps decoding

3.
   Getting Started Summary

- Jenkins
    - Download Jenkins on jenkins.io
    - Open up a terminal in the download directory
    - Run java -jar jenkins.war --httpPort=8080

    - Browse to http://localhost:8080
    - Follow the instructions to complete the installation
- Raygun
    - After signing up, create an application in Raygun
    - 

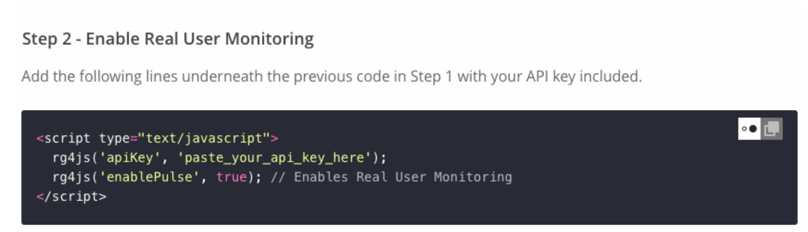
Step 1 - Add the following snippet

Add the following snippet to the beginning of the <head> tag within your markup. Please include this snippet before any other <script> tag references are made to ensure that Raygun has the best chance to capture all error events on the page.

```
<script type="text/javascript">
  !function(a,b,c,d,e,f,g,h){a.RaygunObject=e,a[e]=a[e]||function(){
  (a[e].o=a[e].o||[]).push(arguments)},f=b.createElement(c),g=b.getElementsByTagName(c)[0],
  f.async=1,f.src=d,g.parentNode.insertBefore(f,g),h=a.onerror,a.onerror=function(b,c,d,f,g){
  h&&h(b,c,d,f,g),g||(g=new Error(b)),a[e].q=a[e].q||[],a[e].q.push({
  e:g})}}(window,document,"script","//cdn.raygun.io/raygun4js/raygun.min.js","rg4js");
</script>
```

The above snippet will fetch the Raygun4JS script from our CDN asynchronously so it doesn't block the page load. With Crash Reporting enabled, it will also catch errors thrown while the page is loading.

Alternatively, you can find instructions on how to install Raygun via NPM here.



Step 2 - Enable Real User Monitoring

Add the following lines underneath the previous code in Step 1 with your API key included.

```
<script type="text/javascript">
  rg4js('apiKey', 'paste_your_api_key_here');
  rg4js('enablePulse', true); // Enables Real User Monitoring
</script>
```

    - After installing these two keys, Raygun is good to go!

4. Jenkins and Raygun History

- Jenkins
    - Started in 2004 by Kohsuke Kawaguchi
    - Jenkins is the oldest player in the industry and commands a market share of 71%
    - Jenkins is very active with over 3 commits in the past hour.

- Raygun
    - Started in 2013 by John-Daniel Trask
    - No market share to be found for Raygun
    - Updated recently 2 days ago as of the date this document was made

Step 2: **Runtime Analysis**

	Insert	Append
tinyArray	30.959 μs	80.875 μs
smallArray	48.375 μs	80.5 μs
mediumArray	201.75 μs	131.75 μs
largeArray	9.249375 ms	581.25 μs
extraLargeArray	1.077183041 s	8.609083 ms

The pattern I see is that insert is far better for certain situations. In my opinion, I think each function has its own situational use. Seeing the graphs side by side, you could say Insert is O(n) and Append is O(1). Append is pushing the next number to the end of the array without having to change any of the following arrays inside the index. Unshift on the other hand is having to add one or more elements at the start of the array. Overall though, if I were to choose between the two functions that scales better, it would have to be Insert O(n) because a constant factor with O(1) would be significantly slower if the number were to be like n < 10000000. It's not truly seen in the table, but if the numbers were to be a larger array, Insert O(n) would be more viable. Although depending on the use, both functions could be deemed useful for different circumstances.