```
2024-05-29T16:49:16 Welcome, you are now connected to log-streaming
service.Starting Log Tail -n 10 of existing logs ----
/appsvctmp/volatile/logs/runtime/container.log
2024-05-29T16:18:34.7804331Z As of "05/29/2024 16:18:31 +00:00", the
heartbeat has been running for "00:00:03.1849740" which is longer than
"00:00:01". This could be caused by thread pool starvation.
2024-05-29T16:41:15.4110451Z  [40m [1m [33mwarn [39m [22m [49m:
Microsoft.AspNetCore.Server.Kestrel[22]
2024-05-29T16:41:15.4238881Z As of "05/29/2024 16:41:12 +00:00", the
heartbeat has been running for "00:00:02.7675978" which is longer than
"00:00:01". This could be caused by thread pool starvation.
2024-05-29T16:42:21.8657532Z  [40m [1m [33mwarn [39m [22m [49m:
Microsoft.AspNetCore.Server.Kestrel[22]
2024-05-29T16:42:21.8681997Z As of "05/29/2024 16:42:19 +00:00", the
heartbeat has been running for "00:00:02.4450998" which is longer than
"00:00:01". This could be caused by thread pool starvation.
2024-05-29T16:42:54.1538186Z  [40m [1m [33mwarn [39m [22m [49m:
Microsoft.AspNetCore.Server.Kestrel[22]
2024-05-29T16:42:54.1558907Z As of "05/29/2024 16:42:51 +00:00", the
heartbeat has been running for "00:00:02.2874069" which is longer than
"00:00:01". This could be caused by thread pool starvation.
2024-05-29T16:47:16.9137683Z  [40m [1m [33mwarn [39m [22m [49m:
Microsoft.AspNetCore.Server.Kestrel[22]
2024-05-29T16:47:17.0796217Z As of "05/29/2024 16:47:14 +00:00", the
heartbeat has been running for "00:00:02.4238657" which is longer than
"00:00:01". This could be caused by thread pool starvation.
2024-05-29T16:48:51.2211266Z system test:::Ending Log Tail of existing logs -
--Starting Live Log Stream ---
2024-05-29T16:49:28.6098200Z system test:::
2024-05-29T16:49:29.7726535Z system test:::
2024-05-29T16:49:41.1488077Z system test:::
2024-05-29T16:49:44.7946698Z system test:::
```

Azure App Service troubleshooting documentation

Welcome to Azure App Service troubleshooting.

These articles explain how to determine, diagnose, and fix issues that you might encounter when you use Azure Monitor. In the navigation pane on the left, browse through the article list or use the search box to find issues and solutions.

Capture memory dumps on the Azure App Service platform

This article provides guidance about Microsoft Azure App Service debugging features for capturing memory dumps. The capture method that you use is dictated by the scenario in which you capture a memory dump for troubleshooting a performance or availability issue. For example, capturing a memory dump is different for a process that's experiencing excessive memory consumption than for a process that's throwing exceptions or responding slowly. The process in this context is the Internet Information Services (IIS) worker process (W3WP, which runs as w3wp.exe).

Mapping memory dump scenarios to Azure App Service debugging features

The following table provides recommendations about the commands that each App Service feature runs to generate a memory dump. There are so many approaches to capturing a memory dump that the process might be confusing. If you're already proficient in capturing a W3WP memory dump, this information isn't intended to change your approach. Instead, we hope to provide guidance for inexperienced users who haven't yet developed a preference.

Unresponsive or slow scenario

When a request is made to a web server, some code must usually be run. The code execution occurs within the w3wp.exe process on threads. Each thread has a stack that shows what's currently running. An unresponsive scenario can be either permanent (and likely to time out) or slow. Therefore, the unresponsive scenario is one in which a request takes longer than expected to run. What you might consider being slow depends on what the code is doing. For some people, a three-second delay is slow. For others, a 15-second delay is acceptable. Basically, if you see performance metrics that indicate slowness, or a super user states that the server is responding slower than normal, then you have an unresponsive or slow scenario.

Crash (handled exceptions) scenario

Although a software developer takes special care to determine all possible scenarios under which the code runs, something unexpected can occur. The following errors can trigger an exception: Unexpected null values Invalid casting A missing instantiated object It's a best practice to put code execution into try-catch code blocks. If a developer uses these blocks, the code has an opportunity to fail gracefully by specifically managing what follows the unexpected event. A handled exception is an exception that is thrown inside a try block and is caught in the corresponding catch block. In this case, the developer anticipated that an exception could occur and coded an appropriate try-catch block around that section of code. In the catch block, it's useful to capture enough information into a logging source so that the issue can be reproduced and, ultimately, resolved. Exceptions are expensive code paths in terms of performance. Therefore, having many exceptions affects performance.

API Properties issues

To fix the property issues, follow the below steps

- Go ahead and create a POST request to logic apps and send property name in the request.

- Url for this is “https:”
- Logic apps will trigger a function that will fix your issue and Bitbucket pipeline will redeploy your code.