Memory-related errors

# Memory-related errors#

n8n doesn't restrict the amount of data each node can fetch and process. While this gives you freedom, it can lead to errors when workflow executions require more memory than available. This page explains how to identify and avoid these errors.

Only for self-hosted n8n

This page describes memory-related errors when self-hosting n8n. Visit Cloud data management to learn about memory limits for n8n Cloud.

## Identifying out of memory situations#

n8n provides error messages that warn you in some out of memory situations. For example, messages such as Execution stopped at this node (n8n may have run out of memory while executing it).

Error messages including Problem running workflow, Connection Lost, or 503 Service Temporarily Unavailable suggest that an n8n instance has become unavailable.

When self-hosting n8n, you may also see error messages such as Allocation failed - JavaScript heap out of memory in your server logs.

On n8n Cloud, or when using n8n's Docker image, n8n restarts automatically when encountering such an issue. However, when running n8n with npm you might need to restart it manually.

## Typical causes#

Such problems occur when a workflow execution requires more memory than available to an n8n instance. Factors increasing the memory usage for a workflow execution include:

• Amount of JSON data.

• Size of binary data.

• Number of nodes in a workflow.

• Some nodes are memory-heavy: the Code node and the older Function node can increase memory consumption significantly.

• Manual or automatic workflow executions: manual executions increase memory consumption as n8n makes a copy of the data for the frontend.

• Additional workflows running at the same time.

## Avoiding out of memory situations#

When encountering an out of memory situation, there are two options: either increase the amount of memory available to n8n or reduce the memory consumption.

### Increase available memory#

When self-hosting n8n, increasing the amount of memory available to n8n means provisioning your n8n instance with more memory. This may incur additional costs with your hosting provider.

On n8n cloud you need to upgrade to a larger plan.

### Reduce memory consumption#

This approach is more complex and means re-building the workflows causing the issue. This section provides some guidelines on how to reduce memory consumption. Not all suggestions are applicable to all workflows.

• Split the data processed into smaller chunks. For example, instead of fetching 10,000 rows with each execution, process 200 rows with each execution.

• Avoid using the Code node where possible.

• Avoid manual executions when processing larger amounts of data.

• Split the workflow up into sub-workflows and ensure each sub-workflow returns a limited amount of data to its parent workflow.

Splitting the workflow might seem counter-intuitive at first as it usually requires adding at least two more nodes: the Loop Over Items node to split up the items into smaller batches and the Execute Workflow node to start the sub-workflow.

However, as long as your sub-workflow does the heavy lifting for each batch and then returns only a small result set to the main workflow, this reduces memory consumption. This is because the sub-workflow only holds the data for the current batch in memory, after which the memory is free again.

### Increase old memory#

This applies to self-hosting n8n. When encountering JavaScript heap out of memory errors, it's often useful to allocate additional memory to the old memory section of the V8 JavaScript engine. To do this, set the appropriate V8 option --max-old-space-size=SIZE either through the CLI or through the NODE\_OPTIONS environment variable.

--max-old-space-size=SIZE

NODE\_OPTIONS