Datadog + NodeJS

**To** [**install the chart**](https://docs.datadoghq.com/containers/kubernetes/installation/?tab=helm) **with a custom release name, <RELEASE\_NAME> (for example, datadog-agent):**

1. [Install Helm](https://v3.helm.sh/docs/intro/install/).
2. Using the [Datadog values.yaml configuration file](https://github.com/DataDog/helm-charts/blob/master/charts/datadog/values.yaml) as a reference, create your values.yaml. Datadog recommends that your values.yaml only contain values that need to be overridden, as it allows a smooth experience when upgrading chart versions.

If this is a fresh install, add the Helm Datadog repo:  
helm repo add datadog https://helm.datadoghq.com

helm repo update

1. Retrieve your Datadog API key from your [Agent installation instructions](https://app.datadoghq.com/organization-settings/api-keys) and run:

Helm v3+  
helm install <RELEASE\_NAME> -f values.yaml --set datadog.apiKey=<DATADOG\_API\_KEY> datadog/datadog --set targetSystem=<TARGET\_SYSTEM>

* Replace <TARGET\_SYSTEM> with the name of your OS: linux or windows.

This chart adds the Datadog Agent to all nodes in your cluster with a DaemonSet. It also optionally deploys the [kube-state-metrics chart](https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-state-metrics) and uses it as an additional source of metrics about the cluster. A few minutes after installation, Datadog begins to report hosts and metrics.

Next, enable the Datadog features that you’d like to use: [APM](https://docs.datadoghq.com/agent/kubernetes/apm?tab=helm), [Logs](https://docs.datadoghq.com/agent/kubernetes/log?tab=helm)

Notes:

* For a full list of the Datadog chart’s configurable parameters and their default values, see the [Datadog Helm repository README](https://github.com/DataDog/helm-charts/blob/master/charts/datadog).

### [**Container registries**](https://docs.datadoghq.com/containers/kubernetes/installation/?tab=helm#container-registries)

* If Google Container Registry ([gcr.io/datadoghq](https://gcr.io/datadoghq)) is not accessible in your deployment region, use another registry with the following configuration in the values.yaml file:
* For the public AWS ECR registry ([public.ecr.aws/datadog](https://gallery.ecr.aws/datadog/)), use the following:  
  registry: public.ecr.aws/datadog

Note:

* It is recommended to use the public AWS ECR registry ([public.ecr.aws/datadog](https://gallery.ecr.aws/datadog/)) when the Datadog chart is deployed in an AWS environment.

## [**Log collection**](https://docs.datadoghq.com/containers/kubernetes/log/?tab=helm#log-collection)

* In order to start collecting your application logs you must be [running the Datadog Agent in your Kubernetes cluster](https://docs.datadoghq.com/agent/kubernetes/). To enable log collection with your Agent, follow the instructions below:

To [enable log collection with Helm](https://docs.datadoghq.com/containers/kubernetes/log/?tab=helm), update your [datadog-values.yaml](https://github.com/DataDog/helm-charts/blob/master/charts/datadog/values.yaml) file with the following log collection configuration, then upgrade your Datadog Helm chart:

datadog:

## @param logs - object - required

## Enable logs agent and provide custom configs

#

logs:

## @param enabled - boolean - optional - default: false

## Enables this to activate Datadog Agent log collection.

#

enabled: true

## @param containerCollectAll - boolean - optional - default: false

## Enable this to allow log collection for all containers.

#

containerCollectAll: true

# **Kubernetes Trace Collection**

#### [**OPTIONAL - CONFIGURE THE DATADOG AGENT TO ACCEPT TRACES OVER TCP**](https://docs.datadoghq.com/containers/kubernetes/apm/?tab=helm#optional---configure-the-datadog-agent-to-accept-traces-over-tcp)

The Datadog Agent can also be [configured to receive traces](https://docs.datadoghq.com/containers/kubernetes/apm/?tab=helm) over TCP. To enable this feature:

Update your **values.yaml** file with the following APM configuration:  
datadog:

## Enable apm agent and provide custom configs

apm:

# datadog.apm.portEnabled -- Enable APM over TCP communication (port 8126 by default)

## ref: https://docs.datadoghq.com/agent/kubernetes/apm/

portEnabled: true

Then, upgrade your Datadog Helm chart using the following command: helm upgrade -f values.yaml <RELEASE NAME> datadog/datadog. If you did not set your operating system in values.yaml, add --set targetSystem=linux or --set targetSystem=windows to this command.

**Note**: On minikube, you may receive an Unable to detect the kubelet URL automatically error. In this case, set DD\_KUBELET\_TLS\_VERIFY=false.

# **Tracing Node.js Applications**

## [**Compatibility requirements**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#compatibility-requirements)

The latest Node.js Tracer supports versions >=14. For a full list of Datadog’s Node.js version and framework support (including legacy and maintenance versions), see the [Compatibility Requirements](https://docs.datadoghq.com/tracing/compatibility_requirements/nodejs) page.

### [**Instrument your application**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#instrument-your-application)

Install the Datadog Tracing library using npm for Node.js 14+:  
npm install dd-trace --save

If you need to trace end-of-life Node.js version 12, install version 2.x of dd-trace by running:  
npm install dd-trace@latest-node12

1. For more information on our distribution tags and Node.js runtime version support, see the [Compatibility Requirements](https://docs.datadoghq.com/tracing/compatibility_requirements/nodejs) page.
2. Import and initialize the tracer either in code or via command line arguments. The Node.js tracing library needs to be imported and initialized before any other module.  
   Once you have completed setup, if you are not receiving complete traces, including missing URL routes for web requests, or disconnected or missing spans, confirm step 2 has been correctly done. The tracing library being initialized first is necessary for the tracer to properly patch all of the required libraries for automatic instrumentation.  
   When using a transpiler such as TypeScript, Webpack, Babel, or others, import and initialize the tracer library in an external file and then import that file as a whole when building your application.

### 

### [**Adding the tracer in code**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#adding-the-tracer-in-code)

#### [**JAVASCRIPT**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#javascript)

// This line must come before importing any instrumented module.

const tracer = require('dd-trace').init();

#### [**TYPESCRIPT AND BUNDLERS**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#typescript-and-bundlers)

For TypeScript and bundlers that support EcmaScript Module syntax, initialize the tracer in a separate file in order to maintain correct load order.

// server.ts

import './tracer'; // must come before importing any instrumented module.

// tracer.ts

import tracer from 'dd-trace';

tracer.init(); // initialized in a different file to avoid hoisting.

export default tracer;

If the default config is sufficient, or all configuration is done via environment variables, you can also use dd-trace/init, which loads and initializes in one step.

import 'dd-trace/init';

### [**Adding the tracer via command line arguments**](https://docs.datadoghq.com/tracing/trace_collection/dd_libraries/nodejs/?tab=containers#adding-the-tracer-via-command-line-arguments)

Use the --require option to Node.js to load and initialize the tracer in one step.

node --require dd-trace/init app.js

Note: This approach requires using environment variables for all configuration of the tracer.

Ref: <https://docs.datadoghq.com/tracing/trace_collection/library_config/nodejs/>

# **Enabling the Node.js Profiler**

## [**Requirements**](https://docs.datadoghq.com/profiler/enabling/nodejs?tab=environmentvariables#requirements)

The Datadog Profiler requires at least Node.js 14, but Node.js 16 or higher is recommended. If you use a version of Node.js earlier than 16, some applications see tail latency spikes every minute when starting the next profile.

Continuous Profiler is not supported on serverless platforms, such as AWS Lambda.

## [**Installation**](https://docs.datadoghq.com/profiler/enabling/nodejs?tab=environmentvariables#installation)

To begin profiling applications:

* If you are already using Datadog, upgrade your Agent to version [7.20.2](https://app.datadoghq.com/account/settings#agent/overview)+ or [6.20.2](https://app.datadoghq.com/account/settings?agent_version=6#agent)+.
* Run npm install --save dd-trace@latest to add a dependency on the dd-trace module which includes the profiler.
* Enable the profiler:
  + [Environment variables](https://docs.datadoghq.com/profiler/enabling/nodejs?tab=environmentvariables#)

export DD\_PROFILING\_ENABLED=true

export DD\_ENV=prod

export DD\_SERVICE=my-web-app

export DD\_VERSION=1.0.3

* + [In code](https://docs.datadoghq.com/profiler/enabling/nodejs?tab=incode#)

**const tracer = require('dd-trace').init({**

**profiling: true,**

**env: 'prod',**

**service: 'my-web-app',**

**version: '1.0.3'})**

Note: If you’re already using Datadog APM, you are already calling init and don’t need to do so again. If you are not, ensure the tracer and the profiler are loaded together:  
node -r dd-trace/init app.js

* A minute or two after starting your Node.js application, your profiles will show up on the [APM > Profiler page](https://app.datadoghq.com/profiling).