Node.js Runtime Environment

Node.js 20 is now generally available.

The Node.js runtime is the software stack responsible for installing your web service's code and its dependencies and running your service.

The Node.js runtime for App Engine in the standard environment is declared in the `app.yaml` (/appengine/docs/standard/nodejs/configuring-your-app-with-app-yaml) file:

```
runtime: nodejsVERSION
```

Where ***VERSION*** is the Node.js major version number (#version), for example `20`.

Node.js version

The Node.js runtime supports Node.js 10, Node.js 12, Node.js 16, Node.js 18, and Node.js 20 . The runtime uses the latest stable release of the version that is specified in your `app.yaml` file. App Engine automatically updates to new patch and minor release versions, but it will not automatically update the major version.

For example, your application might be deployed at Node.js 10.9.4 and later automatically updated to version 10.10.0, but it will not be automatically updated to Node.js 12.x.x.

Because minor and patch versions are automatically updated, if present, the `engines.node` (https://docs.npmjs.com/files/package.json#engines) property in your `package.json` (https://docs.npmjs.com/files/package.json) file can only specify the major version and be compatible with the Node.js version specified in your `app.yaml` file.

For example for Node.js 20 :

- `20.x.x`
- `^20.0.0`
- `~20`
- `>=6`

If you specify an incompatible Node.js version in your `package.json` file, your deployment will fail with an error message.

Dependencies

During deployment, the runtime installs your dependencies using the `npm install` (https://docs.npmjs.com/cli/install) command. The runtime also supports Yarn (`yarn.lock`) and Pnpm (`pnpm-lock.yaml`) package managers. For more information, see Specifying Dependencies (/appengine/docs/standard/nodejs/specifying-dependencies). Because the runtime performs a fresh install, you do not need to upload your `node_modules` folder.

Note: By default, the gcloud CLI does not upload your `node_modules` folder. You can change which files are ignored using the `.gcloudignore` (/sdk/gcloud/reference/topic/gcloudignore) file.

To support Node.js packages that require native extensions, the runtime includes system packages enabling you to use tools such as ImageMagick (https://www.imagemagick.org), FFmpeg (https://www.ffmpeg.org/), and Chrome headless (https://developers.google.com/web/updates/2017/04/headless-chrome). See the full list of packages at Included System Packages (/appengine/docs/standard/nodejs/reference/system-packages). To request a package, file an issue in the issue tracker (https://issuetracker.google.com/issues/new?component=441254&template=1123398).

NPM build script

By default, the Node.js runtime executes `npm run build` if a `build` script is detected in `package.json`. If you require additional control over your build steps before starting your application, you can provide a custom build step (/docs/buildpacks/nodejs#executing_custom_build_steps_during_deployment) by adding a `gcp-build` script to your `package.json` file.

To prevent your build from running the `npm run build` script, you must either:

- Add a `gcp-build` script with an empty value in your `package.json` file: `"gcp-build":""`. For details about configuring the `package.json`, see Node.js buildpacks configurations (/appengine/docs/standard/nodejs/running-custom-build-step).
- Add the `GOOGLE_NODE_RUN_SCRIPTS` (/docs/buildpacks/nodejs#google_node_run_scripts) build environment variable with an empty value in your `app.yaml` file.

```
build_env_variables:
  GOOGLE_NODE_RUN_SCRIPTS: ''
```

For details about specifying build environment variables see build_env_variables (/appengine/docs/standard/reference/app-yaml?tab=node.js#runtime_and_app_elements).

Application startup

By default, the runtime starts your application by running `node server.js`. If you specify a `start` script in your `package.json` file, the runtime runs the specified start script instead. For example:

```
"scripts": {
  "start": "node app.js"
}
```

For your app to receive HTTP requests, your `start` script should start a web server that listens on host `0.0.0.0` and the port specified by the `PORT` environment variable (#environment_variables), which is accessible in Node.js as `process.env.PORT`.

For the best performance, the `start` script should be lightweight and exclude build steps, because it runs whenever a new instance of your application is created.

You can override this behavior by specifying a script in the `entrypoint` field (/appengine/docs/standard/reference/app-yaml#entrypoint) in `app.yaml`. Instead of running `node server.js` or a start script, the runtime starts your application with the command you specify in `entrypoint`.

Environment variables

The following environment variables are set by the runtime:

Environment variable	Description
`GAE_APPLICATION`	The ID of your App Engine application. This ID is prefixed with '***region code***~' such as 'e~' for applications deployed in Europe.
`GAE_DEPLOYMENT_ID`	The ID of the current deployment.
`GAE_ENV`	The App Engine environment. Set to `standard`.
`GAE_INSTANCE`	The ID of the instance on which your service is currently running.
`GAE_MEMORY_MB`	The amount of memory available to the application process, in MB.
`GAE_RUNTIME`	The runtime specified in your `app.yaml` file.
`GAE_SERVICE`	The service name specified in your `app.yaml` file. If no service name is specified, it is set to `default`.
`GAE_VERSION`	The current version label of your service.
`GOOGLE_CLOUD_PROJECT`	The Google Cloud project ID associated with your application.
`PORT`	The port that receives HTTP requests.
`NODE_ENV` (Only available in the Node.js runtime)	Set to `production` when your service is deployed.

You can define additional environment variables in your `app.yaml` file (/appengine/docs/standard/reference/app-yaml#environment_variables), but the above values cannot be overridden, except for `NODE_ENV`.

HTTPS and forwarding proxies

App Engine terminates HTTPS connections at the load balancer and forwards requests to your application. Some applications need to determine the original request IP and protocol. The user's IP address is available in the standard `X-Forwarded-For` header. Applications that require this information should configure their web framework to trust the proxy.

With Express.js (https://expressjs.com/), use the `trust proxy` setting:

```
app.set('trust proxy', true);
```

Filesystem

The runtime includes a full filesystem. The filesystem is read-only except for the location `/tmp`, which is a virtual disk storing data in your App Engine instance's RAM.

Metadata server

Each instance of your application can use the App Engine metadata server to query information about the instance and your project.

Note: Custom metadata is not supported in the standard environment.

You can access the metadata server through the following endpoints:

- `http://metadata`
- `http://metadata.google.internal`

Requests sent to the metadata server must include the request header `Metadata-Flavor: Google`. This header indicates that the request was sent with the intention of retrieving metadata values.

The following table lists the endpoints where you can make HTTP requests for specific metadata:

Metadata endpoint	Description
`/computeMetadata/v1/project/numeric-project-id`	The project number assigned to your project.
`/computeMetadata/v1/project/project-id`	The project ID assigned to your project.
`/computeMetadata/v1/instance/region`	The region the instance is running in.
`/computeMetadata/v1/instance/service-accounts/default/aliases`	
`/computeMetadata/v1/instance/service-accounts/default/email`	The default service account email assigned to your project.
`/computeMetadata/v1/instance/service-accounts/default/`	Lists all the default service accounts for your project.
`/computeMetadata/v1/instance/service-accounts/default/scopes`	Lists all the supported scopes for the default service accounts.
`/computeMetadata/v1/instance/service-accounts/default/token`	Returns the auth token that can be used to authenticate your application to other Google Cloud APIs.

For example, to retrieve your project ID, send a request to `http://metadata.google.internal/computeMetadata/v1/project/project-id`.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see the Google Developers Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2023-08-30 UTC.