

Monitoring and Maintaining a Deployed Render Application

Learn how to monitor events, metrics, and logs of your deployed Render applications!

Introduction

Monitoring and maintenance are essential to ensure an application is performing optimally. Thoughtfully, Render provides convenient features for monitoring events and metrics related to our deployed application.

In this article, we will explore how to leverage Render's monitoring and maintenance features on deployed applications. Specifically, we will:

- [Use the "Metrics" and "Events" dashboards to monitor a deployed application](#)
- [Trigger automated re-deployments upon code commits](#)
- [Perform health checks on a deployed application](#)
- [Explore how to suspend or delete a deployed application web service](#)

Let's get started!

Note: This article assumes comfort with basic deployment with Render and uses a pre-deployed application as an example. If you are unfamiliar with deploying with Render, we recommend visiting the [Deploying a simple application with Render](#) article first!

Deployment Monitoring

Deployment monitoring is one of the core features Render offers. On the left-hand side of a deployed application's dashboard, we will find a few different features that Render provides for our deployment. We will explore the "Events" and "Metrics" tabs.

Events

Logs

Disks

Environment

Shell

Previews

Jobs

Metrics

Scaling

Settings

Events

The “Events” tab displays all events and their statuses related to our deployments. Each event will list the commit revision number and commit message along with a date timestamp of when the deployment was attempted. A few examples of events include:

- **First deploy:** The initial deployment.
- **Deploy live:** This indicates that a deployment was successfully deployed to a live running state.
- **Deploy started:** This is triggered when an automated or manual deployment occurs after a code commit.
- **Deploy canceled:** This occurs when a deployment is canceled before it has been completed.
- **Deploy failed:** If a deployment fails, this event will be shown. Some things that may cause a deployment to fail may be missing dependencies or errors in the application code.

Another useful feature of the “Events” tab is that we can re-deploy a previously successful deployment by clicking “Rollback to this deploy” next to the deployment event that we want to rollback to. This option can be helpful, for instance, if we find our current deployment has a bug or broken

functionality and we want to return the deployment to a previously successful version.

Events

Logs

Disks

Environment

Shell

PRs

Jobs

Metrics

Scaling

Settings

Deploy live for e873da5: Update README.md

May 8, 2023 at 9:23 PM

Deploy started for e873da5: Update README.md

New commit

May 8, 2023 at 9:21 PM

Deploy live for e7a5c56: Create `.gitignore`, upgrade `express` to latest (#25)

May 8, 2023 at 8:56 PM

Rollback to this deploy

First deploy started for e7a5c56: Create `.gitignore`, upgrade `express` to latest (#25)

May 8, 2023 at 8:54 PM

Metrics

The "Metrics" tab is helpful for tracking data about our application. Specifically, Render tracks metrics like "Usage" and "Bandwidth" metrics. The "Usage" graph is specific to those Render services running with the "Free" plan and will show how many hours our application has been running. It can also help with tracking both total and average usage. The "Bandwidth" graph will show the total amount of data that our application is sending.



Clicking the “View breakdown” link underneath the usage graph, we can see exactly the amount of build minutes and bandwidth that we have consumed within our instance type tier. Another important metric that can be tracked here is our available “Free Instance Hours”, which represents how many free hours are left to run all of our deployed, free instance web services. Free web services will consume these hours as long as they are actively running, but not if they are [spun down](#). In the event that we run out of “Free Instance Hours”, “Free Bandwidth”, or “Free Build Minutes”, our deployed applications will become unavailable until the first day of the following month, when the hours are reset. If we choose to buy additional hours, we can also set monthly spending limits to cap how many additional hours are purchased.

Free Instance Hours 0.67 hours / 750 hours ⓘ	Free Bandwidth 0 GB / 100 GB ⓘ	Free Build Minutes 3 min / 500 min ⓘ
--	--	--

Build Minutes Spend Limit Edit

Once you've used your free build minutes, additional minutes are charged at a rate of \$5 per 1,000 minutes.

☒ **No limit**
Allow unlimited additional build minutes to be charged according to your needs.

☐ **Maximum spend limit**
Specify a monthly amount you're willing to spend. Builds will not trigger once the spend limit is reached.

\$

Deployment Maintenance

After a service is initially configured, we may need to modify settings. To do so, we can return back to the service dashboard and visit the “Settings” menu option. Here, we can modify any of the previously set configuration fields. There are also a few new options we haven't explored yet:

- **Repository:** This setting allows us to update the link to the code repository that hosts the application. This is helpful if we ever move application code to another repository location or even another platform (e.g. GitHub to GitLab).
- **Auto Deploy:** This setting allows Render to automatically re-deploy the application whenever code change commits are made to the branch. Enabling this setting helps us quickly view deployed changes on the live

website. Turning this setting off will require us to do manual deployments in order for code-commit changes to be deployed to the live website.

- **Custom Domains:** By clicking “Add Custom Domain”, we are able to point the application to a domain that we own. The deployed application can then be accessible via both the custom domain and the Render-provided URL.
- **PR Previews:** By enabling this setting, we are able to access a preview URL that contains all of the changes present in any pull request (PR) that is opened within our connected code repository. This is helpful to visually preview the changes within our pull requests before they are merged into our main branches. Note that every running PR preview does count against our total free instance hours. We can enable this setting by clicking **Edit** and selecting **Enabled** from the dropdown menu.
- **Health & Alerts:** There are additional settings that can notify us when a web service or the deployment process has failed. Also in this section is an option to provide an endpoint that can be called within the application that will check the health of the application. We will cover this concept of a **Health Check Path** more in just a bit.
- **Delete or Suspend:** At the far bottom of the **Settings** page are red buttons for either deleting the web service or suspending it. While the web service is suspended, we will not be billed for any resources. Once a web service has been deleted, all deployment history and events will be deleted and the live website will be terminated. It’s important to note that a deleted web service cannot be recovered, however, this action doesn’t affect the code repository itself.

You can read more about these settings and how they work at the [Render docs](#).

Health Check Path

We discussed an option in the **Settings** page that allows Render to call an endpoint from the web service to check on the application’s health. This endpoint should always return a **200 OK** response, indicating that the application is in a healthy, responsive state. Render will periodically call this endpoint as a means to monitor the health of your application, which helps prevent application downtime.

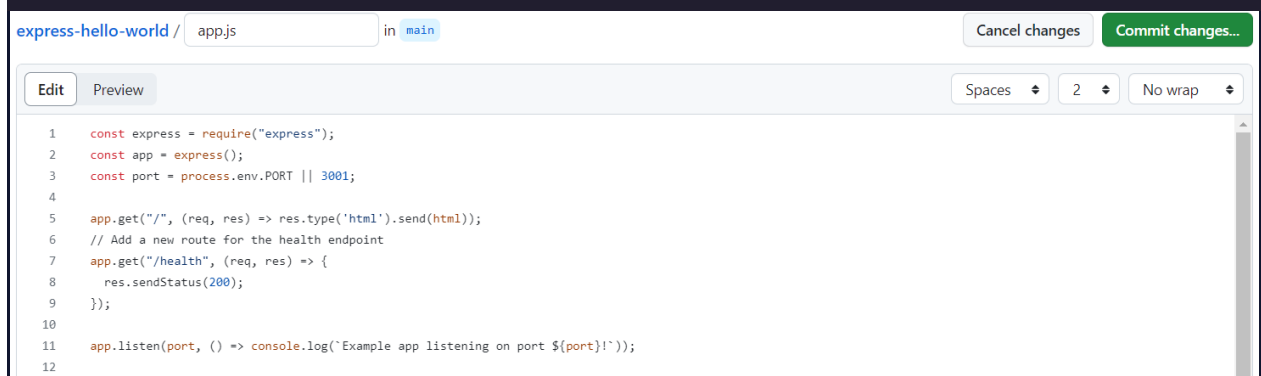
With the **Auto Deploy** setting turned on, let’s try modifying our code and committing a change to trigger a re-deployment of our application. Navigate

to the code repository and open the `app.js` file. Click the pencil icon to edit the file.



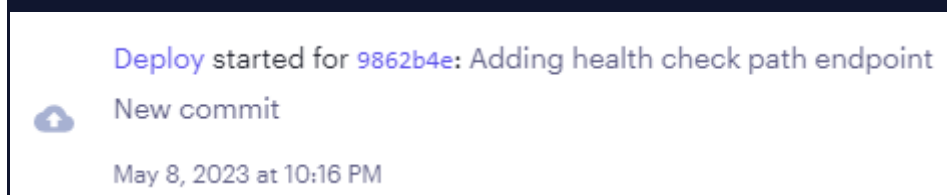
Let's say we want to add a new endpoint that Render can use as the health check path. Within our `app.js` file, let's add the following lines of code directly underneath line 5. Add the code where the comment "Add a new route for the health endpoint" begins:

```
app.get("/", (req, res) => res.type('html').send(html));`:  
// Add a new route for the health endpoint  
app.get("/health", (req, res) => {  
    res.sendStatus(200);  
});
```



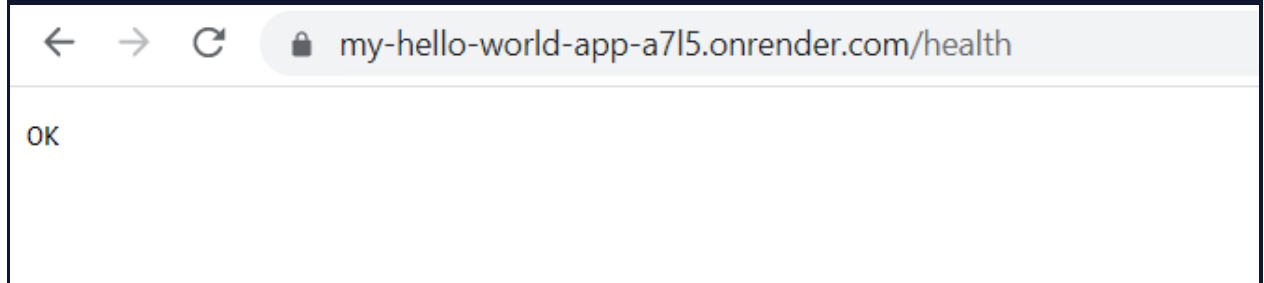
Commit the changes and provide a commit message, for example: "Adding health check path endpoint". This change can be committed directly to the `main` branch.

Return to the Render web service dashboard and refresh the **Events** page. You should see the deployment now running on the commit revision that you just committed (you will also see your commit message).

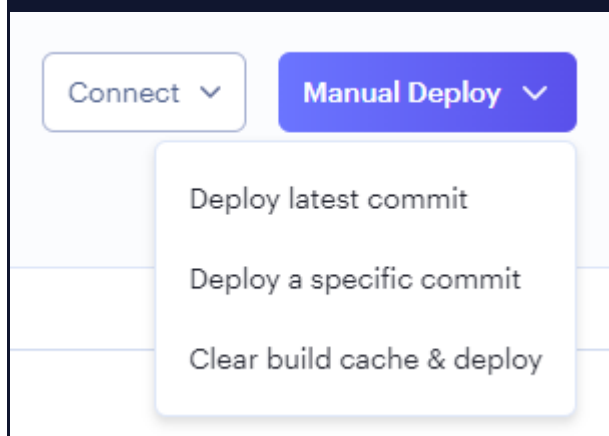


Wait for the deployment to succeed.

Click the Render-provided URL to navigate to the deployed website. In the browser, append “/health” to the URL to navigate to your newly added endpoint. You should see “OK” displayed on the page since our endpoint simply returns a **200 OK** response.



If we want to disable **Auto Deploy**, we can manually deploy our changes by navigating to the web service Dashboard and selecting the **Manual Deploy** button in the top right corner. We can deploy either the latest commit or choose a specific commit revision to deploy.



Let's briefly review the “Auto Deploy” feature:

Free response

What are some benefits to having **Auto Deploy** enabled?

Your response

Auto Deploy offers the developer the possibility of saving time without the need of setting manual deployment.

Our answer

Using the “Auto Deploy” feature allows us to be able to see our new code changes and features quicker since the changes are deployed immediately upon being committed.

Next, we'll try linking our new endpoint to the Render Health Check Path.


Return to the **Settings** page and navigate to the **Health Check Path** configuration. In the field box, select **Edit**, type `/health` and then click Save. This value will match the newly added `/health` endpoint that we added to our `app.js` file in the previous steps.

Health Check Path

If you're running a server, enter the path where your server will always return a **200 OK** response. We use it to monitor your app and for zero downtime deploys.

[Cancel](#)[Save changes](#)

Render will re-deploy the application and there will be a message that it is performing the health check using the provided health check endpoint.

 Waiting for internal [health check](#) to return a successful response code at: `my-hello-world-app-a715.onrender.com:10000/health`

Once the health check and deployment succeeds, the application will move to a green, "Live" state again.

Deployment Troubleshooting

Back on our web service dashboard, we have a menu option to view **Logs**. Log files list out messages related to the build and deployment processes as well as messages that occur during the application runtime — which can be very helpful when troubleshooting issues or bugs within our code. Logs are also useful for displaying regular informational messages, such as messages that we include in our application code. In the sample logfile, there are messages that show the webserver starting and the application running successfully:

Events

Logs

Disks

Environment

Shell

PRs

Jobs

Metrics

Scaling

Settings

Search logs

Search

Maximize

Scroll to top

```
May 8 08:56:33 PM ==> Starting service with 'node app.js'
May 8 08:56:34 PM Example app listening on port 10000!
May 8 08:56:52 PM ==> Detected Node version 20.1.0
May 8 08:56:52 PM ==> Starting service with 'node app.js'
May 8 08:56:53 PM Example app listening on port 10000!
May 8 09:23:20 PM ==> Detected Node version 20.1.0
May 8 09:23:20 PM ==> Starting service with 'node app.js'
May 8 09:23:22 PM Example app listening on port 10000!
May 8 09:23:33 PM ==> Detected Node version 20.1.0
May 8 09:23:33 PM ==> Starting service with 'node app.js'
May 8 09:23:35 PM Example app listening on port 10000!
```

Looking for more logs? Try [Log Streams](#).

Scroll to bottom

Wrap up

- How to monitor events, metrics, and logs for a deployed application
- How to update and modify settings for a deployed application
- How to suspend or delete a deployed application
- How to auto deploy committed changes within the code repository
- How to add a health check path for Render to monitor application health

With the ease of configuring, creating, and maintaining a web service using Render, getting applications into the hands of the end-users is much faster, done more efficiently, all while upholding near-zero application downtimes.