C2 server outage incident report

Wednesday, August 10, 2016
By the C2 team

As we described in an earlier Slack post, C2 had an outage which we resolved on Monday. That post incorrectly stated that no customers were affected. Today we're providing an incident report that details the nature of the outage and our response. We apologize to everyone affected by this issue. We intend to do better than this, and are working to learn from this experience.

Issue Summary

From late August 4 to 9:47 PM ET August 8, visits to the C2 Dev and Staging servers resulted in either 500 error response pages, or silently failing worker processes. At its peak, the issue prevented a GSA customer from performing a scheduled demo for other GSA staff. The root cause of this outage was a loss of server-to-server connections that exposed weaknesses in our deploy process.

Timeline (all times Eastern Time)

August 4
Late in the day: Code deploy begins

August 5
12:19 AM: We discovered Staging outage
2:30 PM – 9:00 PM: Multiple failed manual restart attempts

August 8
9:52 AM: We discovered Dev outage
1:16 PM: Staging is partially operational
5:00 PM: Staging is fully operational
9:47 PM: Dev is fully operational

Root Cause

Some time late August 4, two of C2's three environments lost their bindings to their database and queue services, due to a code deployment. We don't know how the deployment caused this. With no network access to these services, the Dev and Staging web servers began failing. These failures exposed a weakness in our deployment scripts which caused re-starts and re-deploys to only duplicate the existing misconfigured environment. This in turn revealed that C2 is not controlling its deployment with manifest configuration files. At some point, the database server bindings were restored, but not the bindings to the queue service. This resulted in the appearance of full operation even though important functions were silently failing.

Resolution and Recovery

At 12:19 AM August 5 we discovered by manual observation that Staging was down, and posted it in the team Slack channel. The next morning in our weekly retrospective meeting we discussed the outage and set out to determine its extent and causes.

In the morning and afternoon we discussed the problem with cloud.gov support and engineers who had previously worked on C2. At 5:30 PM we learn that Staging is down because it could not connect to its database service. We attempted to restart the server multiple times and via different methods, all of which fail.

At 9:28 AM August 8, we learn that a GSA staff member was trying to use the Staging server unsuccessfully. He was scheduled to give a C2 demo to other GSA staff. We advised him to use the Dev server. At 9:52 AM we discovered that the Dev server was down, totally, or in part. We informed the customer and performed the demo by running C2 locally on a laptop.

At 1:16 PM, Staging is partially operational. We're not clear how this occurred. We report internally that it's fully operational. At 1:50 PM, we realize that Staging is not fully operational and correct our report.

At 4:30 PM, we connect back with cloud.gov support and together bring Staging fully online at 5:00 PM. We learn that our custom deploy scripts are not using the prescribed "manifest file" method of deploying to Cloud.gov — although they appear to be. Instead, the scripts auto-generate manifest files during the deploy process, based on the currently deployed system. They do this as a by-product of the particular "zero-downtime" deploy strategy we use. We apply the quick fixes to Dev and it becomes fully operational at 9:47 PM.

Corrective and Preventive Measures

In the past several days, we've talked with many people at 18F familiar with application deployment to Cloud.gov. We've arrived at a set of actions we will take to address the issues, help prevent them from recurring, and inform us if they do:

- Replace our current "zero-downtime" deploy process with a simple, "plain vanilla" manifest-based deploy setup.
- Create an architecture diagram showing the expected state of the deployed application.
- Document the expected output of diagnostic tools such as `cf services`.
- Configure monitoring services to report when outages like these occur, whether in full or in part, as we experienced.

We appreciate your patience and again apologize for the impact to our customers, and members of 18F.

Sincerely,

The C2 Team