JEDD HOPKINS  
CAPGEMINI  
  
  
4. Store Test Results in S3

**\*\*Please note, many method, class and variable names have been changed to generic terms such as ‘thing’ to maintain anonymity\*\***

**Contents**

Page 2 – Starting the Ticket

Page 3 – Learning About Amazon S3

Page 4 – Writing the Guide

Page 5 – Determining the Method of Upload

Page 7 – Implementing the Upload Using Concourse CI/CD

Page 9 – Conclusion

**Please refer to submission:**

1. **Introduction & Selenium Testing**

**For an introduction to the company, how we operate and how I pick up tickets**

**Starting the Ticket**

The project I am working on currently houses approximately 15 apps, many of which utilise Concourse CI/CD to perform actions such as running tests, without the need for human interaction.

However, at the moment, the tests will run but the test results are not collected in any form. This would prove to be problematic for the test team so going forward we wanted a way to automatically capture the test results to be able to go back and look upon certain builds and the test that were passing at the time.

A screenshot of a social media post

Description automatically generatedThe Dev team needed to determine a way to save these test results in an organised manner which resulted in the creation of the ticket that I assigned myself to (Figure 1.1)

Figure 1.1 – Jira ticket

It had been agreed that I was going to capture the test results in an Amazon S3 bucket, kept behind our infrastructure. I initially spoke to my team lead about how to go about this and he mentioned that a member of the platform team knew how to do it. I then went to speak to them and asked if they would kindly walk me through the process with the thought that I would write up the process in a Confluence document. At this point I added the ‘Write up how to setup s3’ task to the ticket knowing that this information would be valuable when written up in a guide for other developers looking to replicate this task.

My team member agreed to help me later in the day.

**Learning About Amazon S3**

The next part required me to make changes to our Terraform, adding modules to the project infrastructure which holds the config files for everything from AWS to Jira, so I contacted my colleague to gauge when would be best to start our task

Figure 2.1 – Microsoft teams Chat Image

A screenshot of a cell phone

Description automatically generated

Figure 2.1 shows a Slack conversation between a senior member of the platform team and myself. During the call the senior developer walked me through all the steps required to add an Amazon S3 bucket to our infrastructure. The final message is my merge request which the senior developer then double checked, merged and then re-ran into Terraform to integrate it with our environment.

As I was being walked through the steps to create the bucket, I was also taking notes so that I could write up a Confluence page on how I accomplished this task. This proved useful not only by reinforcing what I had just learned but it will now act as a go-to guide for anyone in our team that wishes to create a new instance of an S3 bucket. It can now be found in our How-To section on our project Confluence.

**Writing the Guide**

Figure 3.1 – Confluence page

A screenshot of a cell phone

Description automatically generated

Figure 3.1 shows the Confluence page that I created which gives instructions for any developer if they would like to create their own Amazon S3 bucket. As mentioned, this guide was not part of the original ticket but as a team we try to document everything we do so I decided this would be a good opportunity to do so. This way, if we ever need to perform the task again there is a handy how-to guide and also it creates a documented trail and evidence that we are completing tickets and meeting key deliverables within our sprints and PI’s.

Documentation like this helps members of the client team, that may not have the technological skills as developers, to easily picture what work has been done. Ultimately leading to the sign off of our work and the maintenance of the confidence in our working relationship with the client.

**Determining the Method of Upload**

Now that the S3 bucket was set up, I had to find a way to get our apps to produce test results locally, in order for me to access them. I simply googled “capture maven test results” and found a plugin called ‘Surefire’. A maven plugin is much like a dependency in the way you add it to your project.

Figure 4.1 – Surefire dependency

Figure 4.1 shows the only difference is that you wrap the plugin in <build><plugins><plugin> tags, instead of <dependencies><dependency> ones.

A screenshot of a cell phone

Description automatically generated

Figure 4.2 – Surefire output

As you can see from Figure 4.2, the surefire plugin creates a ‘target’ folder in my project structure, which houses lots of information. However, I am only interested in the surefire reports as these hold the test classes and data that comes with them.

A screenshot of a cell phone

Description automatically generatedIn Figure 4.3 you can see that expanding one of the files will show me how many tests are in that class, along with how many failed and how long the tests took to run. All valuable information going forward.

Figure 4.3 – Individual test file

This proved that I could in fact capture the test results locally within my project. As I wanted to push these tests results to the S3 bucket automatically, I needed to find a way to include an action in our pipeline.yaml which would grab these test results after the build has run.

**Implementing the Upload Using Concourse CI/CD**

After some discussion with my team lead, about potential ways to achieve this task, we decided to follow the route that had previously been taken when performing Concourse tasks. This meant that we would create a new task file which would include all the instructions and environment variables needed for Concourse to upload the test results. Luckily for me we already had some of these created so it was a case of me following these and adapting the various information to achieve what I needed.

In Figure 5.1 you can see the upload-test-results.yaml file.



Figure 5.1 – Concourse script

This file looks complicated but it in essence takes the input of the source-code and mvn-build-output (the mvn-build-output is the output from a previous task), which holds the test results, and tells concourse where to find the results (the params) and eventually push to the S3 bucket in the final run phase.

Figure 5.2 is part of the pipeline.yaml. This is where I create the overview of the task and tell Concourse which file to find and use to find instructions for uploading the test results. As you can see, the way we create tasks is similar across the board. I give the task a name, specify the file it needs to look for to execute that particular task and provide it with a number of variables the task needs to pass through to concourse



Figure 5.2 – Individual test file

The code you see in the snippets above is the final, polished version of what I managed to write, whilst consulting my team for advice when I got stuck. There were many points at which what I had wrote didn’t work, whether that be a problem with grabbing the test results or a problem pushing them to the S3 bucket. To debug whilst undertaking this task I used Concourse itself. Similarly to debugging in an IDE, Concourse will spit out error messages relating to what went wrong and leave a trail of breadcrumbs. Deciphering these messages and getting to the root of problems is an invaluable skill that I have learnt whilst on this project and during my time at makers.

One of the final tasks on my ticket was now to replicate the uploading of test results for all our other apps. This proved to be an easy but meticulous task as the method of uploading results was almost identical in all our other apps. The only thing that needed to be changed for each one was the variable BUCKET\_PATH\_PREFIX. The BUCKET\_PATH\_PREFIX would be added to the s3 bucket to create a tree of all the apps, uploading all the results for the various projects underneath their relevant prefix name. This created an easily readable interface for anyone wishing to look up specific test results for a specific app.

**Conclusion**

Looking back on this piece of work now that it is finished, I can see that it helped me understand how to best use my time researching how to complete individual tasks. A fair bit of this work was unknown to me, for example, how to actually upload data to an Amazon S3 bucket. However, a combination of good documentation and trial and error, through debugging, helped me understand a lot more about what I was doing. It also taught me that whilst trying to learn myself is great, if it is something that is truly above my head then it is better to ask my team for some help rather than continue to bang my head against a wall, as a lot of the time a quick question and answer can take a few minutes, and they are always happy to help explain the problem to me while helping.

The business impact here was that now we had a standardised location for all test results to be saved to. This means that the testers can save a lot of time looking through individual projects and instead look in the s3 bucket for results.