GENERIC KUBERNETES BASED LOAD TESTING FRAMEWORK

Client: Ilya Biryukov – Rapid7

Group 5

Second Years:

Holly Daly - 20331814

Declan Quinn - 20334565

Miguel Arrieta - 20332427

Abdelaziz Abushark- 20332134

Third Years:

Aaron Byrne - 19334098

Tomasz Bogun - 19335070

Vitali Borsak - 19335086

Table of Contents

1. The Project Planning Process 2

2. Project Goals and Objectives 3

3. Project Scope 4

3.1. 5

3.2. Product Backlog 5

4. Project Approach 7

4.1. Scrum Sprints 7

5. Project Organisation 8

5.1. Staff 9

5.2. Staff Chart 12

6. Risk Analysis 13

6.1. Risk Analysis 13

7. Project Controls 13

8. Communication 14

8.1. Client Communication 14

8.2. Project Team Meetings 14

8.3. Demonstrator and Team Meetings 14

1. The Project Planning Process

With regards to the planning of this project, our client laid out our requirements from the very beginning. At our first meeting as a group, each team member showcased their relevant skills and experience. When discussing the project description together, none of us were familiar with the concepts being asked of our client. After our initial meeting, we were still left quite confused but our client gave us great documentation breaking down the project into smaller concepts to tackle. This gave us a great starting point for creating teams.

We decided to split the team into 3 teams. 2 teams of two and 1 team of three.

1. Team One: Project Under Test
2. Team Two: Test runner and Configuration
3. Team Three: Docker and Kubernetes

For the first two weeks of the project, we set out and learned the technologies we would be using and researched new methods of development that we were not familiar with. Team members researched what technologies would be the best to implement the components. We had regular discussions with each other about our findings and how we thought it would be best to use what technologies and practices while keeping in mind the tight deadlines. We didn't have much time to be spending on learning complex technologies we had no experience with at all. When we had decided the languages and technologies we would use, we began developing.

Even though we were split into three teams, each team still had to collaborate with the other to make sure when the 3 components came together there would be minimal tweaking to the code.

Every week we had our own meetings to discuss the progress we have made and what was next on the agenda for every team member whether it be documentation or development. We recorded our scrum videos in these time slots. We also had meetings with our demonstrator to discuss any issues we had with our project or if we had any issues with our client. Everything went smoothly throughout the whole project and both our demonstrator and client were a great help to the team. They were quick to respond to any questions we had about topics relating to docker and Kubernetes and any other questions we had. We used Jira software to help manage the development cycles of our project. On Jira, we had the chance to report bugs and write up stories for upcoming sprints. This agile project management system gave us an insight into how large tech conglomerates would manage all their projects. Coming into the project you don't expect to learn about these ways of managing. But it shows these tools are useful and very important in meeting our goals.

We met with our client weekly unless there were any issues with meeting times or our client had other matters to attend to. These meetings usually entailed us demonstrating our progress and getting feedback on what we could do better or what we could add to the project.

As a team of Second and Third years that did not know each other before this module, we worked amazingly well together. Every member had a pivotal role in all aspects of the project. Our communication was one of the main factors that made this a smooth process. Our management and control process over the course of this semester was very efficient when managing our time and resources in the development and documentation work needed for this module.

2. Project Goals and Objectives

Our client Ilya Birukov is a lead developer at Rapid7. Ilya gave us deliverables when we first met him. We were asked to create a Kubernetes load testing framework that will run requests against HTTP APIs, time their responses, and obtain these performance results.

The following components were needed for the success of the project:

Component	**Description**
Test Runner	A configuration file driven executor of tests, as configured.
Test Runner configuration	A file of a defined format that will tell the test runner what actions to perform.
Project under test	A simple HTTP server that will simulate a database.
Docker Containers	Will be used to package both the test runner and project under test.
Kubernetes	Will be used to take the docker container packages and run them.
Results UI	Graph resource utilization of the project under test using a JS or Python UI.

The team worked very well together to develop the load testing framework. The framework works how our client wants it too and he was very impressed with the implementation. One goal we could not meet was integrating it with Kubernetes. Containerisation and the orchestration of services is a difficult topic to grasp and fully learn while our team members have busy schedules with college and jobs outside of college.

That being said, we did learn a great deal about the topic and with a few more weeks the project would have been a complete success for us and the client.

As well as the goals we did meet, we did integrate a CI/CD system using AWS and Jenkins to automatically build our code and test it when it is pushed to GitHub. Our goal for implementing the CI/CD was to use it to package our code into docker images and run them in an isolated Kubernetes server using AWS EKS (Amazon's Kubernetes Service).

Our final project developed by the team included:

- A mock project under test, a HTTP server to search files. This mock project under test is a replication of the software the team at Rapid7 would be testing.
- A test runner that takes the configuration file and runs certain commands and tests the project under test. The test runner completes various different tests and times the results so our client can compare the software with previous versions.
- A structured file formatted so the test runner can read in the instructions and perform tests the client wants to run.
- Dockerfiles, so the components can be packaged up to docker images and run in an isolated kubernetes server.
- A results UI that pulls the data from the results file generated by the test runner and graphs the new results and previous results so our client can compare performance of their project under test.

Even though the project wasn't deemed a complete success we have learned more from this project than we would have from picking a project we all would have been familiar with.

3. Project Scope

At the start of this semester when we first met with the client we were given recommendations on what technologies to use. Ilya said it was entirely up to us on what we use. We began looking into these recommendations but after discussions, we used technologies that used languages we were more comfortable with. This decision made by the team was definitely the correct one. If we had any problems with these new technologies we would have found it very stressful focusing on fixes to the code rather than new features.

An example of what we changed was instead of using React for the UI we used Python and python libraries to display simple graphs with more information. Also, instead of using Dropwizard for the test runner, we used an apache server to manage the API requests to the project under test. These choices cut the learning times significantly.

From our initial scope defined in our project plan, no changes were made. We stuck to the plan we had from the beginning and our end goal was the same. Due to time, we unfortunately were unable to finish one of the smaller components of the project.

3.1. Product Backlog

From our project plan, our initial backlog was split into requirements, new features, changes to existing features, bug fixes, and infrastructure changes. We made out a table to highlight the importance of each task/backlog. The importance for each task can vary from low, to medium to high.

Initial Product Backlog		
Use description	**Type**	**Importance**
Allow the developer to load test their software with a list of HTTP API requests.	Requirement	High
Package the executable and the test runner up to docker containers and test it in a Kubernetes server.	Requirement	Medium
Record the results in a file. Results vary from performance times and computer utilization.	Requirement	High
Graph the results to compare the other software versions.	New features	High
Autoscaling Kubernetes. Create more instances of the project under test.	New features	Low
CI/CD implemented to automatically package and build the code in a dockers image and run in a Kubernetes server.	New features	Low
Using Jenkins on AWS to automatically build our code every time it is pushed to the GitHub repository.	Changes to existing features	Low
Changes to our mock project under test. Now has a more efficient search algorithm.	Changes to existing features	Medium
Jenkins was not set up correctly. Changes to that so GitHub was connected, and the pipeline was working in its intended way.	Bug fixes	Low
We found a bug in our search algorithm and would not count the number of searches correctly every time.	Bug fixes	High
Jenkins (CI/CD) was set up locally. Now has been set up in an EC2 with AWS Code Deploy.	Infrastructure changes	Low
MiniKube will be used for Kubernetes local testing, but we will migrate to AWS Elastic Kubernetes Service.	Infrastructure changes	Low

The following table shows our refined backlog. It shows when it was identified, what sprint it was dealt in and the impact it had.

Refined product backlog

Use description	**Identified**	**Handled**	**Impact made**	**Sprint**	**Importance**
General learning	Initial meetings with client	Some abandoned, some we used.	The learning brought the success of the project	Sprint 1	High
Using Java and apache instead of dropwizard.	When researching dropwizard.	Moved to a powerful technology that Miguel was familiar with.	Helped us get a great starting point on the project.	Sprint 2	High
Storing test file results with new file names	During planning for sprint 5.	Implemented a fix so user can specify software version	Can compare different versions performance	Sprint 4	High
Jenkins Implementation with AWS CodeDeply	Plan to implement from beginning	Setup to build code for a sprint or 2	Was a great tool to test the software upon pushing to Github	Sprint 3	Low
Jenkins CI/CD stopped	In sprint 4	Abandoned	Had to be stopped as AWS were looking for payment	sprint 4	Low
Kubernetes Autoscaling	Client said we could research	Abandoned	More time to spend on important components	Sprint 3	Medium
Used Python UI instead of React	React had steep learning curve	Used Python for familiarity	Aesthetic UI made before the presentation.	Sprint 5	High
Using helm for Kubernetes	Learning about Kubernetes technologies	Abandoned	Did not fully implement Kubernetes aspect. Did not fully impact core functionality	Sprint 4	Low
Documentation	Requirement	Completed	Final submission	Sprint 5	High
Created Dockerfiles to build images and push to Dockerhub	Wehn implementing Docker and Kubernetes	Docker images built	Learned a lot about Docker and how to build images and how powerful it is.	Sprint 3	Medium

4. Project Approach

We decided to change our Sprint structure from our initial one in the project plan as this was a better way of accomplishing our desired outcome.

4.1. Scrum Sprints

Sprint 1 - 27 Jan to 13 Feb

Length: 2.5 Weeks (17 days)

Scrum Master: Holly Daly

Start: Thursday the 27th of January

End: Sunday the 13th of February

Scheduled multiple meetings with the client for a better understanding of client's desired system.

Decided what software we were going to utilize in our project.

Split the project and the groups into 3 parts.

Sprint 2 - 14 Feb to 6 Mar

Length: 2.8 Weeks (20 days)

Scrum Master: Declan Quinn

Start: Monday the 14th of February

End: Sunday the 6th of March

Created the project under test.

Showed the project under test to the client to make sure that it was what they were looking for.

Learned about testing framework and Kubernetes Cluster.

Sprint 3 - 7 Mar to 27 Mar

Length: 2.8 Weeks (20 days)

Scrum Master: Miguel Arrieta

Start: Monday the 7th of March

End: Sunday the 27th of March

Finished working on the test runner.

Showed the test runner to the client to make sure that it was what they were looking for.

Began looking into UI and React.

Sprint 4 - 28 Mar to 14 April

Length: 2.5 Weeks (17 days)

Scrum Master: Abdelaziz Abushark

Start: Monday the 28th of March

End: Thursday the 14th of April

Displayed results of test and also comparison graphs to the performance of previous iterations of the software.

Presented work to the client.

Finished all documentation.

5. Project Organisation

The project managers (3rd years) make the vast majority of decisions thaking the 2nd years' opinions and wishes into account. If such a case arises where a decision cannot be reached by consensus, the product owner has the final decision making say. This power has not been exercised as we luckily have not had a decision that was voted for unanimously. The team managers look after the wellbeing and happiness of the team, if any issues are to arise, a meeting would be held to see what can be done to resolve the issue, such as switching around or lowering the workload.

Our group was split into 3 teams, one was in charge of the project under test, consisting of one 3rd year and one 2nd year, another team in charge of the test runner, consisting of one 3rd year and two 2nd years, and a team in charge of Kubernetes consisting of one 3rd year and one 2nd year. Each 2nd year team member served as the scrum master for one of the four sprints, overseeing each one is the 3rd year in their respective team. Finally all of the 2nd years contributed to the making of the UI for our project which the 3rd years oversaw.

Throughout the entire project we aim to implement Agile principles in leadership. Our project managers are servant

leaders whose primary job is to enable progress and remove obstacles for the team. With the varying experience
levels on the team, everyone is a teacher and everyone is a student.

5.1. Staff

Aaron Byrne

Role: Team Lead

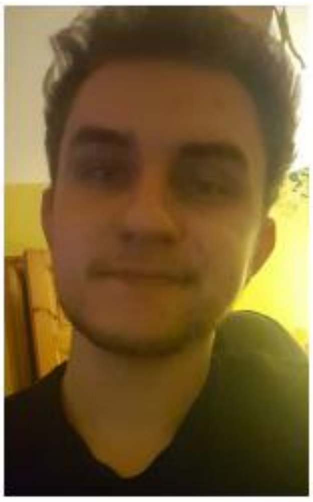
Aaron is a 3rd Year Computer Science and Business Student. From course modules and outside work, Aaron has gained experience and skills in Java, C++, Python, React and other NodeJS technologies.
Aaron was the team lead for this project. This consisted of contacting our demonstrator Jiawen Lyn with any questions the group had or any problems we may have run into during the course of the project. Aaron also was responsible for organising the meetings with the client.
He also had to keep track of the deadlines for the documents throughout the semester.

Tomasz Bogun
Role: Lead engineer

Tomasz is a 3rd year integrated computer science student. He has experience in Java, C++, React from college work as well as python and mobile development technologies like swift and flutter from his personal experience.
Tomasz loves to learn new things and take on exciting challenges. He enjoyed the responsibility that was associated with making a piece of software for a real company that will be used by other people.
He was the lead engineer that oversaw the progress and the quality of the software engineering involved in the project. This was his first time being involved in the management of a software engineering project which he found exciting.

Vitali Borsak

Role: Project Manager



Vitali is currently in his third year of integrated computer science. Having gained vast experience throughout his time in the course, he has learned Python, C, Java, and C#. Some of Vitali's best skills are communication and learning new concepts and ideas in order to better his work. In this project Vitali was the project manager in charge of overseeing the team consisting of two 2nd years, that was tasked with creating the test runner.

During the course of this project Vitali has learned about how a test runner would work and how applicable it is in a real-world software company, how agile development works, but most importantly Vitali has improved his communication and teamwork skills which will be extremely beneficial once he goes out into the working world.

Holly Daly

Role: Test Runner Developer/ Front-end Developer

Holly is currently in her second year of integrated computer science. From past dprojects within and outside of college she has gained a knowledge of Java, C and Python. From doing this project Holly has improved on her team working skills significantly by working on better communication. She also found working with an actual software company to be very beneficial and learned about new aspects of coding. Overall she enjoyed the experience and thinks it will be beneficial in her future projects and career.

Miguel Arrieta

Role: Test Runner Developer/ Front-end Developer

Miguel is currently in his second year of integrated computer science. He has worked on personal projects in technologies like Java, Scala, Python and MongoDB. He has improved on his collaboration, communication and coding skills. He enjoyed the experience of collaborating in a group for a company and believes it will be beneficial for his future career.

Declan Quinn

Role: Project Under Test Developer/ Front-end Developer

Declan is in second year integrated computer science. He has past experience in Java, C, HTML and Python. Over the course of the project he has improved his collaboration and coding skills. Working for Rapid7 has broadened his insight into industry practices and skills. His new experiences will prove to be of benefit after graduation having experience interacting with the workforce.

Abdelaziz Abushark
Role: Kubernetes Integration Developer/ Front-end Developer

2nd year Computer science and business student . I have previous experience in python and Java .I really enjoyed working on this project as it was my first real life experience group project with a company and their software engineer. I gained skills such as communication, time management and research. I learnt how to use new technologies during these modules such as dockers and kubernetes.

5.2. Staff Chart

Scrum Sprints **27/01/2022 - 13/02/2022**

Product Owner: Aaron Byrne

Scrum Master: Holly Daly

Changes: N/A

Scrum Sprints **14/02/2022 - 06/03/2022**

Product Owner: Tomasz Bogun

Scrum Master: Declan Quinn

Changes: Decided to take more time to learn about how the test runner and Kubernetes work.

Scrum Sprints **07/03/2022 - 27/03/2022**

Product Owner: Vitali Borsak

Scrum Master: Miguel Arrieta

Changes: N/A

Scrum Sprints **28/03/2022 - 14/04/2022**

Product Owner: Aaron Byrne

Scrum Master: Abdelaziz Abushark

Changes: Decided to leave out the implementation of Kubernetes into our project as we could not figure how to make it work in time.

6. Risk Analysis

6.1. Risk Analysis

Luckily we did not run into any of the risks outlined in the project plan, however, we discovered and ran into some risks and issues that were not listed there:

Covid-19 Sickness: Thankfully, none of our team members became ill, but it was a risk that we thought after submitting our project plan that could have affected us greatly as members would be ill and be in no shape to work which could have led to us falling behind schedule.

Interference from Other College Work: Something we did not think of in our project plan was the interference from other modules, especially during reading week and closer to the end of term. Luckily through good delegation of work and communication, we were able to distribute adaptable amounts of work to all members as to not hinder the workflow.

Difficult Topics: The final issue that we ran into was that we were tasked with running our testing framework on a Kubernetes cluster, which we thought would be doable with a little bit of research, however, it was quite a difficult task that required more time than we thought. Sadly in the end we did not have enough time to implement this feature into our project.

7. Project Controls

Factors	**Control Method**
Execution	We controlled the execution of our project by continually testing it and making sure there were no bugs or errors, and that the functionality of the project met the clients requirements specification. We ran and tested the execution of the project using various IDEs (Integrated Development Environments) along with external tools such as Postman.
Progress	We controlled the progress of our project by communicating with our teammates regularly in order to see if they are running into any difficulties that could slow down the progress. We aimed to complete each of our individual tasks before our team deadlines, so that in the event of a bug or other development issue that resulted in a delay, there was some additional time to finish things up.
Quality	We controlled the quality of our code by reviewing and giving feedback about any code that was contributed. Whenever any of us were uncertain about their piece of work, we would also ask for suggestions from our teammates before committing it to Github. We found this highly effective because multiple people are more likely to come up with improvements, find bugs, and offer great suggestions than just one person working alone on a task.

Deliverables	We controlled the deliverables by using teamwork and delegation. When we were assigned a new piece of work, we met up and planned it out as soon as possible. We split up the work into smaller chunks that could be worked on separately by everyone in the team. We delegated each subtask to the people that were the most comfortable with completing it. This was effective at reducing the challenges of learning new technologies.
Deadlines	We controlled the deadlines by aiming to start our work as soon as possible. We created soft internal deadlines and schedules when we were assigned tasks. This is so that we had some safe padding in the case of a disruption in the development. This also left us with time to review other people's work to give suggestions and feedback.
Communication	We control our communication using a voice and chat service called discord. We set up a channel and added all the group members to it at the beginning of the semester. We used it for our weekly scrum meetings because it was very convenient to do it there. We also used it to send in comments, files, updates, questions and feedback during the week while everyone was working on their task.

8. Communication

8.1. Client Communication

We met with the client every Friday at around lunchtime. This time was the most convenient for all of us as our timetables didn't clash as much on a Friday. We used zoom for the meetings because we were all familiar with it and it was easy to use. We sometimes gathered in person in the glass rooms and used a single device, and other times some of us were online and joined the meeting from home. During these meetings, we discussed our progress and asked for guidance. We performed a few demonstrations of the work we have done so far to make sure it was what the client wanted.

8.2. Project Team Meetings

We met with the team every week using discord just after the client meeting using zoom. It was the most convenient to do the two meetings back to back because we could discuss the client meeting afterwards. Friday was also a suitable date because it was the end of the week, and we could reflect on the work we did in the previous week, along with the plans we had for next week. During this meeting, we gave feedback, suggestions, analysed our progress and decided what we need to get done for next week, and set soft guidelines for the deadlines.

8.3. Demonstrator and Team Meetings

We met with our demonstrator every Monday at 6:50 pm. Here we discussed our progress and the demonstrator gave us some feedback. We also asked questions about things we were unsure about, and the demonstrator gave us some suggestions to help out. Our demonstrator was very helpful because she always reviewed any work we submitted to blackboard and talked about it in the meetings.