Project report DevOps and Cloud-Based Software

Group 2:
Daan Meijers 10727167
Rosco Kalis 10771603
Antonino Sauleo 12394866
Tjarco Kerssens 11153407

March 2019

1 Introduction

This report describes the work done for the group project for the course DevOps and Cloud-based software. For this, a definition of the project is provided in the Project outline section. Since we work scrum on this project, we have also provided a description of the progress in the Scrum documentation section. Furthermore, a literature study and reflection on the teamwork is provided.

2 Project Outline

2.1 Project Description

For this project, created a website where it is possible for users to create and share recipes. These recipes can be viewed by anyone, but in order to use the site users have to be logged in. When logged in, users can create recipes with a title, ingredients, instructions and a picture. Editing and deletion is only possible on owned recipes.

2.2 Infrastructure

In this section, we give an overview of the processes and infrastructure which are used in the project to have a good setup with continuous deployment and integration.

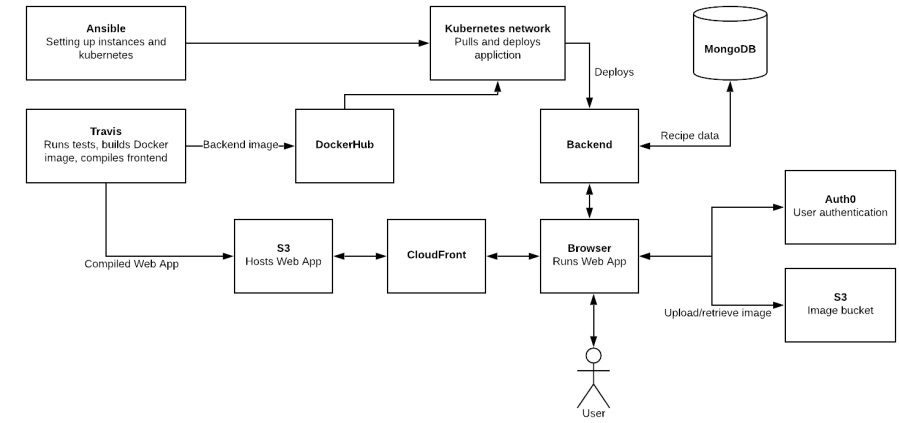

Figure 1: An overview of the infrastructure

For the infrastructure of the application, we took a lot of inspiration from the labs and lectures. Furthermore, we did some comparative literature research in differences between Chef, Ansible and Saltstack. We choose to go with Ansible, since we already had experience with it for the lab and the research also concluded that it would be the most intuitive and simplest to set up. Also, Ansible performed second when looked at performance, but required the least server resources. Since we use the cheapest possible Amazon instances, this is an important factor in our choice for Ansible.

We wrote two Ansible playbooks, one for the set-up and one for deployment. This will take place on Amazon EC2 instances, where there is one master and multiple workers. The setup playbook will install all dependencies and setup the Kubernetes network. The Deployment playbook will configure the server to use the deployment file which instructs how many replicas it requires and where to pull the container image from, including which version. This image is hosted on DockerHub.

For our continuous integration we chose to use Travis, as our literature review showed that this was best suited for open source projects hosted on GitHub, it had very extensive documentation, and not too much of a learning curve. We use Travis to run all the tests, build a Docker image and compile the front-end when a commit is pushed to the master branch. The Docker image is pushed to DockerHub and the compiled web application is put on S3. A CloudFront service provides an endpoint to the hosted Web Application for the user.

For the back-end, MongoDB was installed in an EC2 instance in order to store the recipes due to its great flexibility for storing object models as documents. Another reason we decided to use this database, is the fact that schema-less

databases provide a much faster rate of data processing and efficiently manage large volumes of data.

Furthermore, the front-end makes use of the Auth0 services to manage a user store. This service is responsible for the storage, registration and authentication of users. The Auth0 also provides a pre-packed login/register page called Lock, which is used to validate input and securely communicate with their user store endpoint. Finally, a second S3 bucket is used to store the images that are uploaded from the front-end for the recipes.

2.3 Architecture

In this section, a description is provided for the architecture of the service. All of the components are hosted at Amazon Web Services.

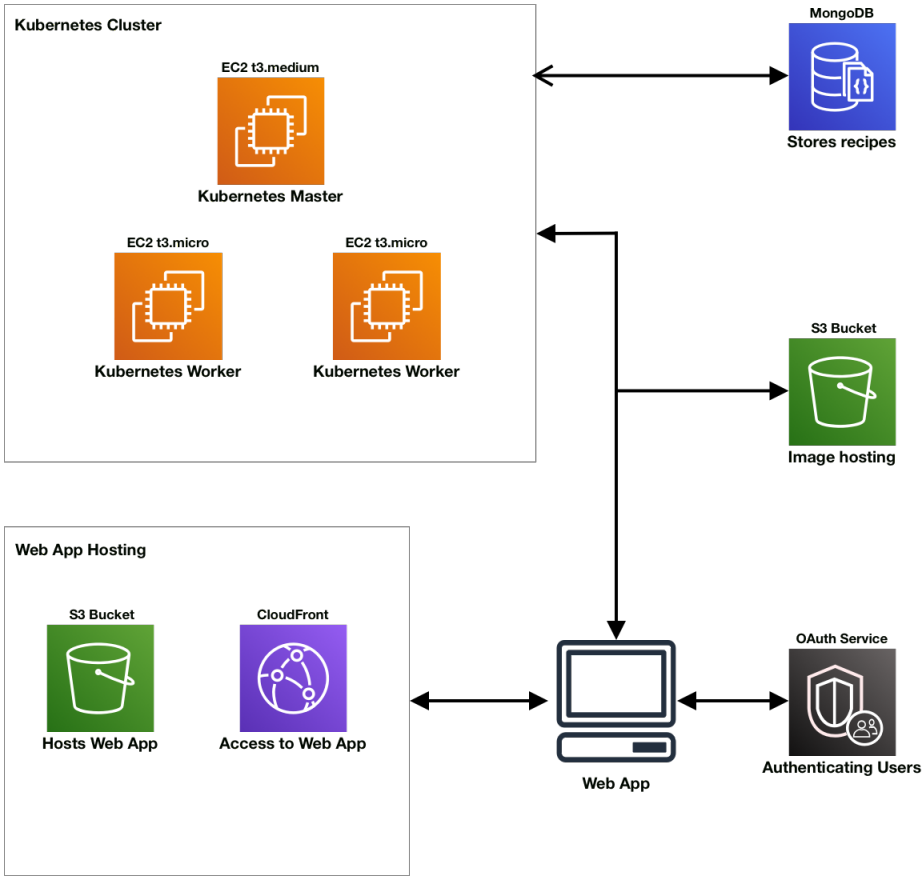

Figure 2: An overview of the architecture

As described in the infrastructure section, the back-end is deployed on a Kubernetes cluster. This cluster is hosted on EC2 instances, where the master is hosted on a t3.medium instance and we have two t3.micro workers. All the instances are running on the Ubuntu operating system.

Furthermore, we use S3 Buckets for storage of the website and the images and a MongoDB database for the storage of recipes.

The OAuth service is not part of the Amazon Web Service stack, but a separate service that provides integration opportunities with AWS.

3 Scrum Documentation

For this project, we are working as a scrum team and are applying the concepts we have learned from experience and the guest lecture by Jelena Gordijenko. Below, we have outlined how we have applied the different aspects of a scrum workflow to our project.

3.1 Roles

First, we have defined roles for the project, with some minor adaptations.

Nino: Scrum Master
As scrum master, Nino will make sure that we as a team will follow the scrum process for the project activities. Nino has experience with working scrum and can apply this in this project. We have made an adaption to this role where Nino will also be participating in the team work, rather than only governing the process as is custom with a scrum master.

Daan: Product owner
Daan had the idea for the product and is therefore the product owner. He prioritizes the work in the backlog and makes sure that the user stories are relevant to the product he has in mind. Daan also still works on the project as team member.

Rosco: Team Member
Rosco works as a team member. During sprint meetings, decisions are made as a team, where all project members have an equal say.

Tjarco: Team Member
Same as with Rosco, Tjarco works as team member without an explicit extra role in the scrum process.

3.2 Process

We do one week sprints, from Wednesday to Tuesday. Therefore, we have a sprint meeting Wednesday after the lecture. Later in this document, we provide a detailed description of each sprint. In each sprint, we make sure that there is enough work for every team member. We do this by writing stories in the backlog and assigning points to them. We then divide the stories over all project members, ensuring everyone has about equal work.

We start each sprint with a retrospective, where we talk about what we did last week, what went well and what could have been better. We try to take this possible improvements into account for the next sprint.

3.3 Storyboard

We have created a storyboard on Trello, where we keep track of the work. See figure 3 for a screenshot of our board.

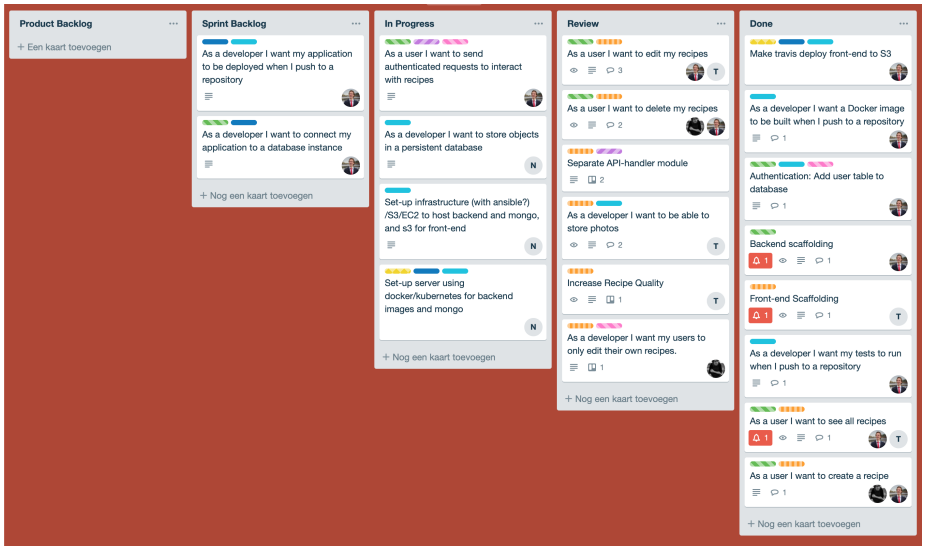

Figure 3: The Trello board with user stories

We write each story from an actor perspective, to create a better idea of the actual feature. In figure 4 you will find an example of a user story.

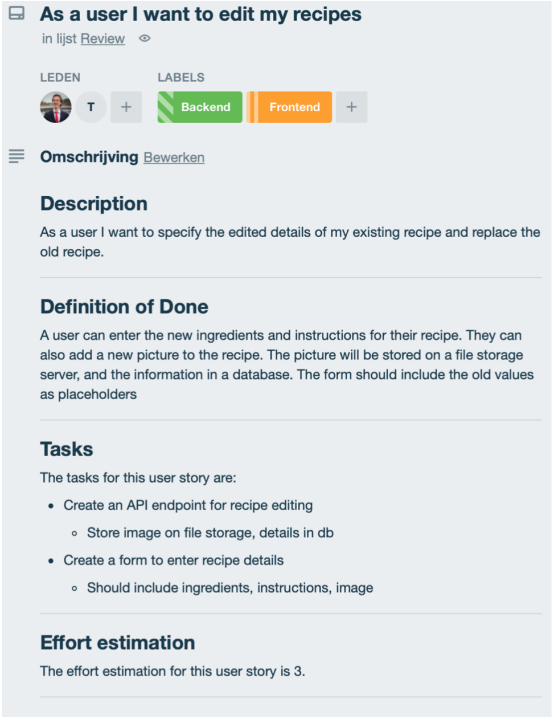

Figure 4: An example of a user story

3.4 Sprint 1

We started the first sprint in week 3, where we had a meeting with all project members and we decided on the process. During this meeting, we have filled the backlog and assigned stories to all project members.

In the first sprint, we have delivered a MVP of our final product. In this MVP, it is possible to view, add and edit recipes. We also have setup the workflow with Travis. For the next sprint, we can iterate on this process by improving recipe quality, creating the infrastructure and expanding the database with user accounts.

Since everyone has already seen the product, the demo was not that surprising. We did not find out any new things during the demo.

During the retrospective, we have concluded that the user stories are too dependent on each other, making it hard to work really agile, since we often had to wait for other stories to be finished first. We have tried to reduce this dependency for the next sprint.

3.5 Sprint 2

During our second sprint the team realized that it would be more effective to adopt pair programming as a methodology since some of our members already finished with their tasks and could help other group members with the most relevant user stories that would help us deliver the final product with higher quality code on time.

Furthermore, after knowledge acquisition from our previous tasks, we decided to re-define the infrastructure of our system. This is due to the fact we realized some functionalities did not bring any added value to the final product, and would have delayed other much more relevant user stories. This was very helpful since it allowed the team to focus on finishing with a working prototype on time and taught us the usefulness of applying agile methodologies.

4 Reflection

During this project, we have learned a lot about working together in a scrum team to deliver an actual product, using DevOps principles of continuous integration and deployment.

At the start, we still had to get used to working in this Agile way. The sprint meetings were longer, since we needed to figure out how we were going to shape the process. Luckily, we got a guest lecture by Jelena Gordijenko in order to learn more about the ways to work scrum.

This experience in working scrum is probably very important for our later careers, since more and more companies seem to be adapting a scrum work process.

Furthermore, gaining first hand experience with common tools used for continuous integration and deployment, will probably also give an advantage in the work field.

We also gained some experience with pair programming, which seems a promising method for improving the quality of the code.

5 Literature Study

5.1 Comparison Of Resource Utilization And Deployment Time For Open-source Software Deployment Tools [6]

The purpose of this study is to compare the software deployment tools Ansible, Chef and SaltStack regarding deployment time and their respective resource utilization. The study is aimed to assist system administrators to make informed decisions when deciding which application to use to manage their computers and infrastructures.

This study is interesting for our project, since we will be using a deployment tool for the application. During the lectures, Yury has talked about these tools used for the deployment of applications, and we thought it would be a good idea to find research into the differences.

Ansible is an open-source tool that uses SSH to handle automated deployment. It does not use agents and no background daemons are necessary. No network traffic is needed when the Ansible server is idle. Ansible is also regarded as very simple, compared to Chef and SaltStack.

Chef is also open-source, but uses remote agents on each client. Cookbooks and recipes are used to automate deployment and are managed by a Chef server. Chef cookbooks have version control features, that enables collaboration by multiple administrators on the same cookbooks.

SaltStack has an open-source and an enterprise version. It can be use either with or without agents. Agentless, it uses ssh, just like Ansible and is quite similar.

The paper researches the following questions:

- RQ1: Do the software deployment tools Ansible, Chef, and SaltStack still differ in deployment time in the same way compared to Benson et al. (2016) when deploying services and/or applications?
- RQ2: Do the software deployment tools Ansible, Chef, and SaltStack vary in resource utilization when deploying services and/or applications?

The research concludes that Chef has the fastest deployment time, followed by Ansible and Saltstack is the slowest. However, Ansible uses the least server resources (CPU, memory), which can be important for our project since we deploy free tier (or at least as cost effective als possible) servers. Eventually we chose for Ansible over Chef and Saltstack. While Chef is the fastest, it also uses

more resources and was more complex to implement. Ansible seems to be the best choice.

5.2 Making Software: Pair Programming [8]

Pair programming is a style of programming in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test. It has been identified that pair programming has positive effects on the quality of the code, while only increasing the time programmers need to spend marginally (15%).

If only one person understands an area of code, the team can suffer if this person leaves the team or is unavailable due to sickness or vacation. When more than one person is at least casually familiar with each area of the code, risk is reduced for the team.

Pair programming is an agile software development technique that we decided to try during the project, hence this resource was used to identify the benefits and techniques needed to have a good setup for pair programming.

With pair programming, one of the pair, called the driver, types at the computer or writes down a design. The other partner, called the navigator, has many jobs. One of these is to observe the work of the driver—looking for tactical and strategic defects in the driver's work. Some tactical defects might be syntax errors, typos, and calling the wrong method.

Furthermore, we used the article written by Knyga et al. to identify what practical scenarios and techniques could be used in a pair programming setup [2]. They identified a problem where the driver would make the most decisions and the navigator would wander off in his mind over time. One proposed solution is 'strong style pair programming', where the driver can only write what the navigator directs. If the driver has an idea, he should give the keyboard to the navigator [2].

We did not actually use this idea of strong style pair programming, but we did switch often between the driver and navigator. We found that we could debug errors faster, but that some aspects may have been completed faster if we would have divided the work op traditionally. We also did not do the whole project pair programming style, but only when we were working together at the UvA.

5.3 Auth0's Approach to Information Security[3]

Auth0 provides Identity-as-a-Service, which means that they host, manage and secure user stores. The organisation has published multiple whitepapers regarding their work, of which we have reviewed one that describes their security policies[3]. These policies are found in the quality of their development team, the development process, infrastructure, disaster recovery and other topics.

Everyone in the workforce of Auth0 has to comply with certain security policies to ensure compliance with law and best security practices. These policies includes thorough screening from a third-party company, which is repeated every five years. At the head of the workforce is a security director with nearly two decades of experience in security work. Furthermore the organisation developed an access-request tool that handles requests from developers. This system is automated and generates a reliable audit trail.

Auth0 has designed a development process that is constantly monitored, is able to rapidly respond to issues and encourages efficient software testing. In order to do this the company uses a DevOps software process for which they use tools such as Puppet and Jenkins. The organisation made effort to comply with multiple security specification, which include; OAuth 2.0, OpenID Connect, SAML, WS-Federation and LDAP. To ensure code quality and security all developed components are subject to peer-review. The organisation also makes significant use of thoroughly reviewed open-source code which they use for their systems, to which to often contribute patches. Furthermore they have developed tools that checks for common developer mistakes, such as hard-coded credentials or vulnerable or outdated third party packages. Auth0 also has a Responsible Disclosure Program that encourages researchers to investigate the companies services and search for vulnerabilities. Finally they follow OWASP (Open Web Application Security Project) recommendations that feature topics such as input validation, SSL/TLS protection and more.

Auth0 uses the public cloud to deploy their services, and therefore the security of the management of their clouds services is one of their top priorities. In order to do this they constantly monitor the state of their services, enforce multifactor authentication (MFA) and scan all accounts for configuration errors. Managing the cloud infrastructure is done with programmable configurations using Puppet.

Finally, since the Auth0 is in charge of multiple user stores redundancy plays a big role. Therefore the Auth0 service makes use of long-established hosting providers with sophisticated redundancy protocols. In order to ensure continuation of support in times of crisis a Business Continuity Plan (BCP) has been developed. This BCP provides guidelines that ensure availability of appropriate personnel for all services and infrastructure at all times. Finally backups are frequently made, monitored and verified as part of the BCP.

5.4 Introduction to DevOps on AWS[4]

As became apparent during this course, DevOps methods enable efficient and effective code development, deployment and monitoring. The core principles of DevOps have been embedded in the Amazon Web Services platform. This whitepaper discusses a multitude of services offered by AWS that facilitates the use of DevOps methods[4]. Due to the fact that we have been discovering the use of DevOps for the larger part of this course we did not get to work with all of these services. However, we have become familiar with a few and have acquired useful knowledge for future enterprises. The following part will discuss a few of the services mentioned in the whitepaper.

The structure of the report follows the DevOps cycle, starting with the infrastructure. A core principle of DevOps is treating the infrastructure as code. This means that the infrastructure configurations should be defined in a declarative way and stored in a version management system. Also, using "infrastructure code" the infrastructure should be managed and provisioned. This paper lists two options of using infrastructure as code in the AWS environment, using CloudFormation and AMIs. CloudFormation has been before used during the weekly assignment, so we are somewhat familiar with this service. The service allows users to deploy infrastructure by defining them in templates. These templates can be custom made, but there are also sample templates available on AWS. In order to deploy EC2 instances the user can make use of AMIs (Amazon Machine Images). These images are some sort of digital templates, and can contain various software configurations for webservers, application servers and databases. As described in the architecture section, we have used Ubuntu AMIs ourselves to create three EC2 instances for our Kubernetes cluster.

AWS provides multiple services for continuous deployment. This whitepaper describes a couple of them; CodeDeploy, CodePipeline, CodeCommit, Elastic Beanstalk and OpsWorks, and finally Blue-Green Deployment. While we did not use any of these services, Blue-Green Deployment did seem interesting. This service uses domain name services to make application deployments. It assumes all traffic is going to the application in a blue zone. When a new version is deployed it is set in a green zone. When all tests pass the traffic can be rerouted to the new application zone, either incrementally or instantly. When all traffic is rerouted the first zone can either be used as backup or be decommissioned.

Another core principle of DevOps is automation. An example of automation as part of continuous deployment is provided in the Elastic Beanstalk service. This service allows the user to define an environment or technology stack for applications to work in. After creating a proper configuration, deployment is as easy as uploading a .zip file to Elastic Beanstalk. The service will build the environment and deploy the application automatically.

When using a multitude of different interconnecting services, it becomes increas-

ingly important and complex to monitor these. In order to do this effectively AWS provides the CloudWatch service. This service has also been used during one of the practices, so we are quite familiar with it. CloudWatch allows users to monitor multiple components of their DevOps environment. It is able to produce alerts when certain events occur. In order to facilitate automation, these alarms can be used to trigger actions such as the upscaling or downscaling of the infrastructure.

Finally comes the aspect of security in the AWS DevOps environment. Easy creation of security profiles is done using the Identity and Access Management (IAM). These profiles can be used to define permissions and form a "role". These roles can be appointed to users or services granting them the defined permissions.

From this course and this paper it seems that AWS is a proper platform to use for work using DevOps methods. The paper also provides an example of a DevOps pipeline, using CodeDeploy, CodeCommit, CloudFormation, Elastic Beanstalk and more. While it can be extremely useful to combine as many AWS components as possible, this is not necessarily efficient for all types of applications. We have used a couple of these tools, but it seemed inefficient to spend time figuring out and using most components due to the size and timescale of our project.

5.5 Challenges When Adopting Continuous Integration: A Case Study [5]

The purpose of this study is to analyse the challenges that can arise when implementing Continuous Integration processes in an organisation. To do so, they conduct qualitative research in the for of a case study at a company in Sweden. The company is a world leading telecommunication equipment and services provider and has over 100.000 employees. Several teams within the company have adopted CI into their process.

The study uses semi-structured interviews to find out how the process of migrating to CI within the teams has been. From these semi-structured interviews, they analyse trends and came to the following challenges in implementing CI processes.

Something that many interviewees mentioned was the developer mindset. Existing developers tended to be sceptical of the improvements that CI offers. They also have difficulty with changing their old development habits.

Another major issue was the supporting tools and infrastructure, which lacked maturity. In the case of this company, all the required tooling was developed in-house, so it can be expected that the maturity levels might be lacking.

Another challenge was in automated testing, which is a prerequisite to CI according to interviewees. The software at the company had too many manual tests, and some unstable automated test cases, which made the continuous integration difficult. This can be a big problem for many bigger software projects that implement CI after the fact.

5.6 Top 8 Continuous Integration Tools [9]

This online article presents a comparison of popular CI tools, in this case it compares Jenkins, TeamCity, Travis, Go CD, Bamboo, GitLab CI, CircleCI, and Codeship. Most of these tools offer both paid and free versions of their software, but for the sake of our project, we mainly took the free versions into account.

Of the reviewed tools, Jenkins was the most comprehensive and had the most possible functionality, but it does come with somewhat of a learning curve.

TeamCity is another comprehensive tool made by JetBrains that integrates well with other JetBrains tools such as IntelliJ or WebStorm. It focuses more on enterprise clients, and the free functionality can be limited.

Travis is praised for its excellent documentation, and it is seen as a mature solution that is trusted by many clients. It offers its full functionality for free for Open Source projects, and integrates very well with GitHub repositories

Go CD is named as a great solution for complex deployment scenarios, which can be suitable for bigger and more complex projects.

Bamboo is Atlassian's CI tool that is hosted on-premise. They offer a cloud version as well, called Bitbucket Pipelines, but this only offers a paid plan.

GitLab CI is integrated with the GitLab version control platform and it integrates seamlessly with all the other functionality of GitLab.

CircleCI is mainly used for its extreme performance. With up to 16x parallelisation it is able to run very fast. However, the associated cost is also higher.

Finally, Codeship offers a good all-round service, although the free version has some limitations, most notably a lack of Docker support.

5.7 DB-SECaaS: a cloud-based protection system for document-oriented NoSQL databases [1]

This article discusses the advantages and reasons why most companies are using document based NoSQL databases such as MongoDB on the cloud, the vulnerabilities of such databases, and proposes a comprehensive database security-as-a-service (DBSECaaS) system, on top of such databases.

Some of the advantages this paper discusses of NoSQL databases in comparison to relational database management system (RDBMS) is that NoSQL databases provide a much faster rate of data processing, which can result in a relatively inexpensive way for enterprises to efficiently manage large volumes of data. Furthermore, by being an schema-free database it provides great flexibility for updating data, without the need for any significant restructuring, such as storing an object model as a document. Another advantage is the improved performance and decreased concurrency side effects due to the independence of the documents from one another.

Nevertheless, such databases provide role-based access controls, where users having the permission to read from or write to a database can apply the operation on the entire database. This model of access control is discussed to have many vulnerabilities opening doors for malicious insiders to access and/or attack such databases. As a result, it proposes a DBSECaaS system which provides strong authentication, fine-grained authorization, and data encryption to ensure maximum security for the document-oriented database layer lying underneath.

5.8 Scrum: Enhancing Student Team Organization and Collaboration [7]

This paper discusses the importance and advantages of using SCRUM, and provides an activity for small groups in order to introduce them with Agile methods. Some of the advantages of using SCRUM in a consistent way include feelings of achievement and satisfaction, increased team member communication and organization, reduced team conflict and increased student reflectivity about the collaboration process. Furthermore, this paper claims that Scrum teaches students how to plan short, medium, and long-term goals, offers students tools for breaking complex tasks into manageable concrete pieces, makes prioritizing tasks according to project needs, and sharing information in non-evaluative ways. Also, it is also discussed that Scrum does not prevent a lazy member, however, the teams are able to document specifically that member's lack of action when they write team evaluations.

References

[1] Ghazi Yumna et al. *DB-SECaaS: a cloud-based protection system for document-oriented NoSQL databases*. https://link.springer.com/article/10.1186/s13635-016-0040-5.

[2] O Knyga et al. *Pair Programming Guide*. https://medium.com/@weblab_tech/pair-programming-guide-a76ca43ff389. 2018.

[3] Auth0. *Auth0's Approach to Information Security*. https://cdn.auth0.com/website/security/auth0s-information-security-wp.pdf.

[4] David Chapman. *Introduction to DevOps on AWS*. https://d1.awsstatic.com/whitepapers/AWS_DevOps.pdf.

[5] Adam Debbiche, Mikael Dienér, and Richard Berntsson Svensson. "Challenges when adopting continuous integration: A case study". In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2014, pp. 17–32.

[6] Jonathan Johansson. *A comparison of resource utilization and deployment time for open-source software deployment tools*. 2017.

[7] Susan Opt and Christy-Dale. *Scrum: Enhancing Student Team Organization and Collaboration*. https://www-tandfonline-com.vu-nl.idm.oclc.org/doi/pdf/10.1080/17404622.2014.939675?needAccess=true.

[8] Andy Oram and Greg Wilson. "Making software: What really works, and why we believe it". In: " O'Reilly Media, Inc.", 2010. Chap. 17: Pair programming, pp. 311–322.

[9] Vladimir Pecanac. *Top 8 Continuous Integration Tools*. 2016. URL: https://code-maze.com/top-8-continuous-integration-toolxws/.