Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

DevOps Dashboard with Heatmap*

Márk Török[a], **Norbert Pataki**[b]

[a]Department of Programming Languages and Compilers,
Eötvös Loránd University
tmark@caesar.elte.hu

[b]ELTE Eötvös Loránd University, Budapest, Hungary
Faculty of Informatics, 3in Research Group, Martonvásár, Hungary
patakino@elte.hu

Abstract

DevOps is an emerging approach that aims at the symbiosis of development, quality assurance and operations. Developers need feedback from the test executions that Continuous Integration servers support. On the other hand, developers need feedback from deployed application that is in production.

We have created a DevOps dashboard tool that visualizes how the deployed applications behave in production. In this paper, we present an extension of our dashboard tool. This extension is a heatmap that presents the features' usage. DevOps tool provides result from the end-users, so it can be seen if a new feature is unused or an old one needs more capacity because too many users take advantage of it.

Keywords: DevOps, Dashboard, Feedback, Heatmap

MSC: 68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)

1. Introduction

Nowadays software development is rather a workflow than a process. It has plenty of phases and usually many teams work on a software separately but at the same time. While software engineers design the architecture, and do the programming,

*The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

the team of test engineers prepares the test cases for the application and business analysts extend the requirements of the further features. The team is responsible for the Continuous Integration (CI) pipelines as well as setting up the environments for the different phases of the workflow, like integration environments for the test team, or for the developers in which resources for the team like fixtured databases, REST or storage services are provided.

All members of the team have to keep their eyes on those environments and pipelines that belong to their roles. They monitor the changes and, through their tools, provide feedback of the state of the workflow at any time. Instant feedback responses are essential from the perspective of time, money and, many times, motivation. The faster the teams get a picture about the particular part of the workflow, the faster they can react.

As we see, monitoring is one of the main and most important responsibilities of the whole engineering team. An environment could be full of measurable segments, from the different memory usages to the usability of the CPU, to the application-specific information, to the healthcheck or status of services. Moreover, the environment can contain databases, application or web servers, storage services, mail servers, or can be the host of multiple, parallelly running jobs or processes. All of these consume memory and CPUs or GPUs, and most of them can provide custom information about their memory consumption, number of working threads running inside the application, or healthchecks.

Since most of these environments belong to the test or developer teams, the necessary enviroment changes by commits or merge requests. Every commit could bring changes on the code base as well as on the tests. These changes can result in different memory, CPU, GPU and service usages. Comparing the results of the different build pipelines, teams can easily get a picture about their current work and the effects that those changes could bring.

Giving feedbacks about the observed system can provide information about how the memory consumption is optimized or how well the processes are scheduled. Introduction of new features can suprise the engineering team if they use more resource than it was expected. As a new dimension on the top of resource monitoring is the tracking of user-actions and exception handling of features, and how a newly introduced feature change the usage of resources.

In this paper, we present our tool that can provide additional dimensions above the usual monitoring as well as provides instant feedback about the different environments and status information about the services running inside those environments in the mirror of the changes. We present the implementation details of the heatmap functionality. This tool is able to work together with DevOps-related infrastructures, such as Docker [2].

The rest of this paper is organized as follows. The usual DevOps toolset is presented in Section 2. After, our dashboard tool is presented in Section 3. We detail our heatmap functionality in Section 4. Finally, this paper is concluded in Section 5.

2. DevOps Toolset

In this section, we show the useful tools in the DevOps culture. We use these tools as a basis of our solution; the proposed approach is an actual extension of this toolset.

Continuous Integration (CI) is a widely-used practice to discover compilation and functional defects rapidly during development. The major aim of CI is to avoid integration hell [9]. This is a state in which developers spend more time preparing the code to be checked into the repository than writing the code itself. Extreme programming (XP) emphasizes the importance of continuous integration. A build or CI server (e.g. Jenkins) checks the version control repository periodically and launches the compilation and testing process if any change is discovered [10]. If there is some kind of failure, it can be seen who have committed changes, what are the commit messages and which files are affected in previous changes in the repository. The status of the CI pipeline is usually displayed on big monitors so software engineers can observe it over time. Major benefits of the CI approach are the fast feedback, visibility and traceability.

Continuous Delivery (CD) is a software development discipline. This discipline aims at building software in such a way that the software can be released to production at any time [4]. It is a series of processes that aims at the safe and rapid deployment to the production. Every change is being delivered to a production-like environment called a staging environment. Rigorous automated testing ensures that the business applications and service work as expected. Since every change is delivered to a staging environment using complete automation, one can have confidence the application can be deployed to production easily when the business logic is ready [6]. Therefore, this approach is widely-used in modern software engineering [3].

The DevOps approach extends the CD discipline and focuses on comprehensive CD pipelines: starting with building, followed by different kinds of testing. Unit testing, component testing, integration tests, end-to-end tests, performance testing should be performed on the software. Code coverage is measured as well. In the meantime, static analyzer tools try to find bugs, code smells and memory leaks in the source code [1]. 3rd-party compliance shall be checked in the build pipeline. Automated vulnerability scanning of the software is mandatory to discover security gaps. The documentation of software (e.g. user guide, API description, etc.) can be generated during the execution of pipeline. The visibility of the whole process is guaranteed.

After this phase, the automatic deployment of application starts. Application Release Automation (ARA) tools are available that can communicate with the CI server and the deployment steps can be designed on the graphical user interface of these tools. The DevOps culture argues for the deployment automation at the level of the application. The automatic upgrade and roll-back processes involve many difficult challenges. Database schemas, the path and filename attributes of configuration files, names and number of configuration parameters, APIs, and 3rd-

party components (e.g. message queues) may be changed when a new software version is released. The deployment process has to cover these changes as well. This approach requires automation and visibility.

DevOps considers the monitoring and logging of the deployed application in the production environment [7]. The development team is eager for feedback from the application which is in the production environment. The feedback may include many aspects of the software: for instance, unused features in the software, memory or other resource leak detection or performance bottlenecks. Developers have to get as much information as possible to be able to take care of a trouble. Problems may cause automatic roll-back of the application to the previous stable version.

ELK stack is a mainstream end-to-end approach for collecting, filtering, analyzing and visualizing log [8]. ELK stack consists of three separate applications that work together [5]. Elasticsearch, Logstash and Kibana are included in the ELK stack.

The Logstash component is responsible for collecting logs. It supports wide variety of sources where the logs come from and it ensures the centralized logging infrastructure. Logstash provides custom logic for parsing logs come from any kind of application.

Elasticsearch stores the collected logs. It is a search server which provides a distributed, multitenant-capable full-text search engine. Elasticsearch has a RESTful API using JSON for querying and filtering the underlying Logstash that contains the collected logs. Client libraries are available for many programming languages.

Kibana is the front-end of the ELK stack. It is a log comprehension tool which is able to visualize the logs as a web application. It provides sophisticated analytics capabilities, seamless integration with Elasticsearch, and wide range of charts and other graphical opportunities.

Logs of complex applications can be incomprehensible for human processing. Log analysis may involve big data analysis. Tools are available that extend the ELK stack with big data analysis features (e.g. Elasticsearch for Apache Hadoop (ES-Hadoop)).

The analyses of logs and monitoring data are application-specific and their evaluation may be difficult. Using big data analysis, machine learning can be involved. In this paper, we do not focus how the logs should be processed in a general way. We assume that logs of a specific application can be processed and usage information can be extracted automatically. However, in this paper, we take into consideration if the feedback reports are received by architect or design team who defines the work for the development team.

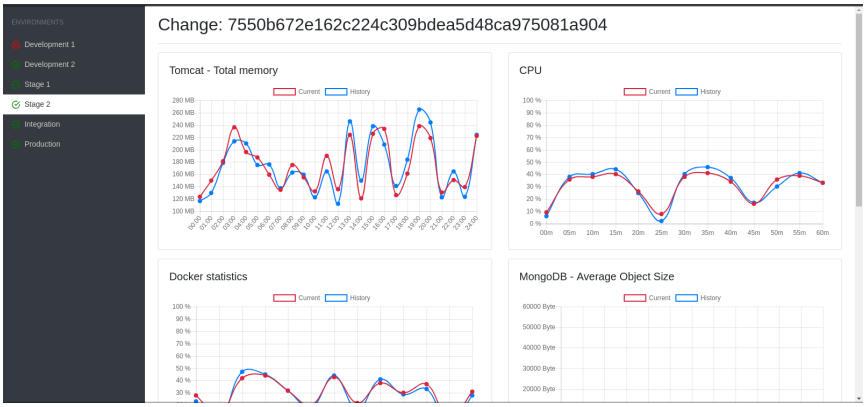
3. Our Dashboard

Dashboards have been part of the DevOps toolset since the beginning. There can be different kind of dashboards on the market with different purposes and goals from the beginning to the end of the CI/CD pipeline.

A classic dashboard for monitoring tool shows the metrics which are collected from the different environment (eg. test, stage, production). Our dashboard extends the classic approach with an additional dimension. We show the differencies based on merge requests, and can compare the current metrics to the historical ones.

Monitoring these changes can provide a closer picture about how the application works in the different environments and provide feedback about the changes that have been applied. This approach reflects whether the new features, bug fixes, optimizations can bring better performance on the specified resources. Monitoring an environment or a service inside an environment requires such an interface to gain information about them. In our approach, we took advantage of agents to observe the changes. These agents are located on machines that play the role of the hosts of the environments. Every single agent is reliable for watching only one service per environment and to hand over the logged data to the Dashboard application.

In the regular way, after CI executes phases (build, deploy, test), sets up the services and then deploys the application, it executes tests against it. In our solution, CI pipeline is responsible to deploy not just the application onto the environment but it sets up the whole environment as well with services, in place of the old, altered one. CI installs our agents onto the environment with itself. This way we do not have stuck processes, dangling objects in the cache, our abandoned files on the disk. The agent sends information to the Dashboard containing all observed data about the deployed application and services. The Dashboard presents the collected data in charts whether it is a worker count from Tomcat or Docker statistics.



4. Heatmap

As to log the different metrics and get a subtle picture about the system, about the application, about the environment, and how they interact to each other, we introduced agents that send the specific information about them to the dashboard application which accumulates them and shows them as metrics.

Agents can gather information from the appropriate environment like how much processor is used by the application, how much memory is consumed, what size of volumes are taken during the execution, it can provide a picture about the cache, the band-width or any other instances that run on the machine.

Knowing how changes make affect on the system is inevitable if the developers are intended to make decisions about which way we would like to go, what technologies we would like to use, what patterns we would like to apply. On the other hand, sometimes the result of changes can be hidden from agents that monitor the resources of the given infrastructure of the environment. Though, the memory consumption has not changed, the cache is fast as it was in the previous release, but the number of requests from client side started dropping. What if we introduced a new feature to the application and realised that from that point, the CPU usage increased dramatically. We know exactly which release brought the changes, though we would like to visualise the reaction from the client side.

We provide a solution for tracking the behaviour of the application from inside as well. As shown below, we implemented an extension that can be placed in the application and log the specified parameters when the appropriate feature is in-use.

The feature itself is identified by a unique identifier that is passed as a required argument of the agent. Additionally user can pass such an arguments like which user called the function. Like date and time information are logged automatically, thus the dashboard can provide information regarding the usage in a timeframe as well.

```
get '/new-feature' do
    FeatureAgent.log('feature-identifier',
```

```
        user: current_user,
        call_stack: caller_locations)

    # some business logic
    # implementation comes here

    render 'new-feature.page'
end
```

Heatmap is a graphical technique and approach of visualising data. Here, toning starts from green, which represents the less active points, to yellow, to orange, to red, which represents the area with most frequent actions on it.

For the heatmap, we retrieve all usage information based on the logged features' usage data. We can create runtime metrics and statistics, and define how "popular" a feature is. This popularity value is mapped to colors. Thus, we do not deal with HTML structure and any client-related information.

5. Conclusion

DevOps is rather new approach for the development, maintenance of modern applications. DevOps is based on symbiosis of development, quality assurance and operations. DevOps requires feedback from all phases of development, deployment and usage of the production. DevOps focuses on the end-users because they do not want to deal with subtle background of the entire programming processes.

We have created a tool that retrieve information from the application from the staging or production. Our solution is able to compare the different releases or commits. We take advantage of hardware utilization as well, but we focus on the usage information for realizing if features are unused and can be discontinued. We present our heatmap extension for the tool.

References

[1] Babati, B., Horváth, G., Májer, V., Pataki, N., Static Analysis Toolset with Clang, In Proc. of the 10th International Conference on Applied Informatics (ICAI 2017), pp. 23–29.

[2] Bernstein, D., Containers and cloud: From LXC to Docker to Kubernetes, *IEEE Cloud Computing*, Vol. 1(3) (2014), 81–84.

[3] Cukier, D., DevOps patterns to scale web applications using cloud services, in Proc. of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity, SPLASH '13, 143–152.

[4] Chen, L., Continuous delivery: Huge benefits, but challenges too, *IEEE Software*, Vol. 32(2) (2015), 50–54.

[5] Langi, P. P. I., Widyawan, Najib, W., Aji, T. B., An evaluation of Twitter river and Logstash performances as elasticsearch inputs for social media analysis of Twitter, In Proc. of the 2015 International Conference on Information Communication Technology and Systems (ICTS), 181–186.

[6] Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V. P., Itkonen, J., Mäntylä, M. V., Männistö, T., The highways and country roads to continuous deployment, *IEEE Software*, Vol. 32(2) (2015), 64–72.

[7] Lwakatare, L. E., Kuvaja, P., Oivo, M., Dimensions of DevOps, In Proc. of the 16th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2015), 212–217.

[8] Révész, Á, Pataki, N., Containerized A/B Testing, in Proc. of the Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA 2017), pp. 14(1)–14(8).

[9] Révész, á., Pataki, N., Integration Heaven of Nanoservices, in Proc. of the 21 th International Multi-Conference INFORMATION SOCIETY IS'2018, Volume G: Collaboration, Software and Services in Information Society, pp. 43–46.

[10] Stolberg, S., Enabling agile testing through continuous integration, In Proc. of the Agile Conference, 2009, (AGILE '09), 369–374.

[11] Török, M., Pataki, N., Service Monitoring Agents for DevOps Dashboard Tool, in Proc. of the 21 th International Multi-Conference INFORMATION SOCIETY IS'2018, Volume G: Collaboration, Software and Services in Information Society, pp. 47–50.