Ethics in DevOps, The Attitude of Programmers Towards it

Muhamed Skenderi
University of Tetova
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Shkurte Luma-Osmani
University of Tetova
20 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Journal of Natural Sciences and Mathematics of UT, Vol. 5, No. 9-10, 2020

UDC: 004.4-05:172
Review article

ETHICS IN DevOps, THE ATTITUDE OF PROGRAMMERS TOWARDS IT

Muhamed Skenderi[1], Shkurte Luma-Osmani[1*], Florinda Imeri[1]

[1]Faculty of Natural Sciences and Mathematics, University of Tetova, Republic of North Macedonia*

[*]Corresponding author-mail: shkurte.luma@unite.edu.mk

Abstract

DevOps represents combination of the engineer's team and the operations team, working together in an automated environment and also in a repeatable way. Working this way will help in getting the things done faster. Development and operations teams are entirely two different teams and it has been always impossible for them to work together. DevOps is not based on stringent methodologies and processes; it is based on professional principles that help business units collaborate inside the enterprise and break down the traditional silos. This paper contains general information about DevOps technology in the Software Engineering. The first part contains an introduction in DevOps, a short history of it, the concept of making the development team and operations team work together and the role of the DevOps Engineers. Later, we will talk more deeply for DevOps and its own tools, thus continuing with the ethical issues in DevOps, where we talk about the principles in coding and their importance in developing. Continuing this way with the results from a survey where each question is demonstrated with a figure and explanation in which all participants are Developers. The main purpose of the survey is to get the opinion of Programmers about ethical issues in Developing, how much they are informed about principle in ethical coding, do they practice them and have they ever written some unethical code. In the end as a conclusion we have mentioned some reasons for code that accomplishes something unethical during the process of Development in Software Engineering.

Keywords: ethics in coding, development, operating, software engineering, testing

1. Introduction

If we had asked about the origin of DevOps, the answer would be Patrick Debois. He was an IT Consultant from Belgium, who in 2007was part of a project from the Belgium government on large data center migration, where he was on charge of testing. During the work on this project, he recognized that a lot of time and effort was wasted navigating the project between the world of Developers and the Operations Team. This was the exactly moment when things took direction.

In 2008, during an agile conference in Toronto, Andrew Shafer tried to put a meetup session called "Agile Infrastructure". Patrick was the only one who took part on this meetup. They formed a discussion group for other people to post their ideas for how to solve this divide between development and operations later that year. Things became more popular when Patrick put out a call on Twitter for a gathering to bring developers and system administrators together in Ghent on October 2009. However, he knew that he needed a name for this event and he called it DevOpsDays. A lot of ideas were exchanged that day on the meeting, although after the meeting the debate continued virtually in Twitter, by using DevOps hashtag.

So why is so important this history? Well, if you don't know where you are from, you really can't know where you are going.Transforming your organization into a DevOps culture is not as simple as buying new software system. It is important to know that DevOps is not a single product as it evolved from the need for adaption and continuous improvement.

The guiding principles of DevOps include culture, measurement, automation and sharing.DevOps is considered to be a new approach to the more traditional application lifecycle management (ALM) process. In the enterprise there is a need to break down silos, where business units operate as individual entities within the enterprise where management, processes and information are guarded. Anyway, when problems arose, hardware engineers would come in and declare, “It’s not the machine; it’s the operators” (Davis & Daniels, 2016).

2. DevOps Cycle

DevOps is basically a combination of two words, "development" and "operations." Usually, DevOps is a culture that follows the set of practices to combine the Development and IT Operation teams. Its major goal is to shorten the system delivery life cycles. There is a total of seven phases in the DevOps lifecycle – Continuous development, continuous integration, continuous testing, continuous delivery, continuous feedback, continuous deployment, and continuous operations.

There is a lot of detail in the following overview image of the Continuous Delivery pipeline, and you most likely won't be able to read all the text. While the basic outline of this image holds true surprisingly often, regardless of the organization. There are, of course, differences, depending on the size of the organization and the complexity of the products that are being developed. The early parts of the chain, that is, the developer environments and the Continuous Integration environment, are normally very similar. The number and types of testing environments vary greatly. The production environments also vary greatly (Verona, Duffy, & Swartout, Learning DevOps: Continuously Deliver Better Software, 2016).

2.1 Continuous Development

In the first phase of DevOps lifecycle, you should plan your application objectives that must be delivered to the customer. Once you are sure of application objectives, start with the project development. It includes activities like code generation and putting the same to the next phase.

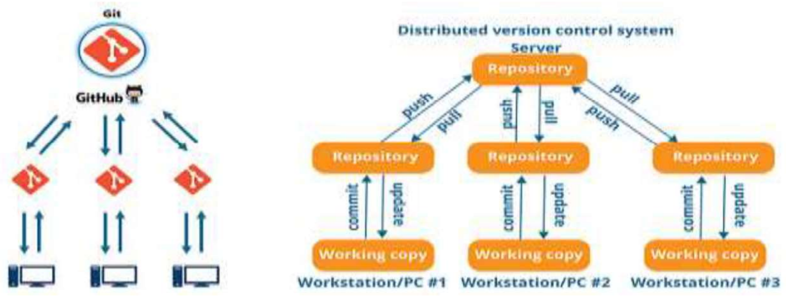

Figure 1. Continuous Development with Git technology
Source: https://www.edureka.co/blog/devops-tools

2.2 Continuous Integration- The continuous integration process automatically starts after development. It includes several steps like the planning of tests that will be carried out in the next phase, understanding the code to produce the desired outcome as needed in the initial project documentation. Continuous integration is the seamless process in DevOps that leads to the next phase in an efficient manner.

2.3 Continuous Testing- Testing process checks the actual use of an application in the DevOps. Beta testers produce results while still ensure that application can have its intended use in a live environment. The testing process gives more information about different aspects of an application that in turn is sent to the development process to improve the application.

2.4 Continuous Monitoring- The monitoring phase is the operational phase in DevOps where key information about application usage is recorded and carefully processed to find out trends and identify the problem areas. It enhances the operational efficiencies of a software product that may occur in the form of documents or produce massive data about application parameters when the application is in a continuous use position.

2.5 Continuous Feedback- The application performance is improved consistently by analyzing the final outcome of the product. The continuous feedback is an important phase of the software application where customer feedback is a big asset to improve the working of the current software product and release new versions quickly based on the response. With continuous feedback you can improvise the current product and release new versions quickly.

2.6 Continuous Deployment- Ensures product is deployed with maximum accuracy. The deployment process is performed in such a way that any changes made in the code should not affect the functioning of high traffic website. It is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

2.7 Continuous Operations - Also known as automate release process with shorter development cycles. All DevOps operations are based on continuity with complete automation of the release process and allow organizations to accelerate the overall time to market on an ongoing basis.

It is clear from the discussion that continuity is the critical factor in DevOps removing the abundant steps that often distract the development, take it longer to detect issues, and producing a better version of the product after several months. With DevOps, you can make any software product more efficient and increase the overall count of interested customers in your product. Let us see how businesses are benefitted through DevOps deployment. DevOps is a practical and valuable asset for organizations because of its numerous measurable benefits through continuous integration and continuous delivery without compromising on the quality of software products or services. DevOps adoption enables businesses to revive the pace(Jan Bask Training, 2019).

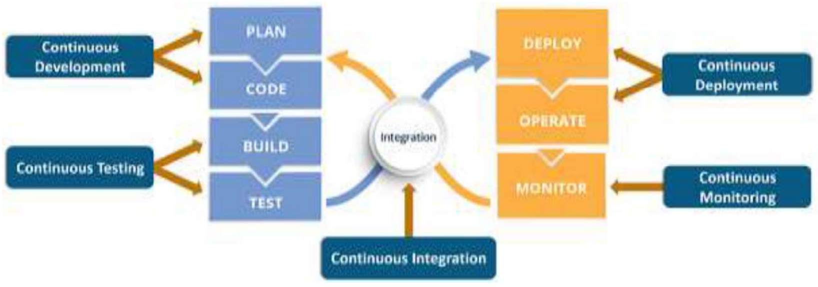

Figure 2. DevOps Cycle
Source: https://www.edureka.co/blog/devops-tools

3. DevOps Tools

There are a lot of tools and technology of DevOps that enable companies of all sizes and resource levels to leverage the technical practices of DevOps, regardless of where they are along the spectrum of organizational culture change. Because DevOps strives to turn software development into a strategic asset, it's critical to select the right tools for the job. DevOps promotes speed, efficiency, and reliability, and the right tools and technology make these benefits a reality.

3.1 Log monitoring tools

There are number of DevOps log monitoring tools available in market. Here in this section some of the DevOps log tools are explained.

1. *Splunk*- an America Multinational Corporation situated at San Francisco, California. It is a log monitoring tool which produces software via web interface for searching, monitoring and also for analyzing huge amount of data generated by machine. Also, it generates indexes and relates this with real time data which is in a searchable repository to generate graphs, reports, alerts and visualization. Splunk is the world's first Data-to-Everything Platform(Splunk, 2020).
2. *Logstash*- an open source tool for collecting, parsing, and storing logs for future use. Kibana 3 is a web interface that can be used to search and view the logs that Logstash has indexed. Both of these tools are based on Elasticsearch. Elasticsearch, Logstash, and Kibana, when used together is known as an ELK stack.(Anicas, 2014).
3. *Kibana*- an open-source data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support. Kibana is a free, open-source visualization tool (Anicas, 2014),(Amazon Web Services, 2020).
4. *Paper Trail*- a log management DevOps tool that is available for general availability. Nowadays companies have learned that it can get business IT insights from log data. It is open source software which do not limit for storing the data daily; it has the retention time of 28 days. It alerts the user or the admin whenever the log data arrivers or whenever the retention time is reached through email; it can also be used to manage the cloud-based organization; it supports on demand data analysis; in this there is a guarantee of the query completion.

5. *Loggly*- a cloud-based enterprise log management and analytics tool. A log is the automatically produced and time-stamped documentation of events relevant to a particular system. Virtually all software applications and systems produce log files. Log data has traditionally been of interest to technical departments, like IT operations and DevOps environments. Loggly makes log data accessible and useful to more groups within an enterprise. By facilitating the retrieval of specific data and presenting it in a user-friendly interface, the Software as a Service (SaaS) product makes log data more valuable to customer service, application development groups, management, marketing and more.(SearchITOperations, 2020).

3.2 System Monitoring Tools

1. *Graphite*- an open source tool written in Python, which is used to track the values of any metric that changes dynamically. It stores the data, renders the graphs and it will monitor the performance of a computer system. The user has to use some existing tools (like collectD, statsD, Gmond and so on) or write some applications for collecting the data. It handles numeric time-series data. This scalable graphing tool can also be used with the cloud environment. Plugins are used to collect the data, checks are used for monitoring the data. Mainly it has three components (Davis C. , 2014):
   - Carbon-this is the processing backend which listens to the data points and it can handle a huge number of clients.
   - Whisper-it is similar to RRD that offers fast and reliable storage of the received data points over time.
   - Graphite Webapp-Django webapp will render the graphs on demand, when there is a request for graph it retrieves the data from the disk and if it is not yet been written on to the disk then it will take the data points directly from the carbon in order to produce real-time graphs.
2. *Ganglia* - a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world. It has been used to link clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes (Ganglia Monitoring System, 2018).
   Ganglia monitoring daemon-which will be installed on all the nodes that has to be monitored and it will interact with the operating system of the host to acquire the system-specific metrics. Ganglia Meta daemon- it will collect the information from many gmond or gmetad sources and stores it in the RRD (Round-Robin database). Ganglia PHP Web front-end- it will present the collected data. The graphs of any metric from two or more hosts can be aggregated so that it can be viewed simultaneously.
3. *Sensu*- an open source tool written in ruby; it is used in the cloud environment. Sensu is designed to monitor everything from the server closet to the cloud. Install the Sensu agent on the hosts you want to monitor, integrate with the Sensu API, or take advantage of proxy entities to monitor anything on your network. Sensu agents automatically register and de-register themselves with the Sensu backend, so you can monitor ephemeral infrastructure

without getting overloaded with alerts (Sensu Go, 2020). This monitoring router runs checks on the systems that need to be monitored which will return one of the following common exit status codes:

- 0 – OK
- 1 – WARNING
- 2 – CRITICAL
- 3 – UNKNOWN

3.3 Network Monitoring Tools

1. *Zabbix*– created by Alexei Vladishev, and currently is actively developed and supported by Zabbix SIA. Zabbix is an enterprise-class open source distributed monitoring solution. It is software that monitors numerous parameters of a network and the health and integrity of servers. Zabbix uses a flexible notification mechanism that allows users to configure e-mail-based alerts for virtually any event. This allows a fast reaction to server problems. Zabbix offers excellent reporting and data visualization features based on the stored data. This makes Zabbix ideal for capacity planning. Also, it supports both polling and trapping. All Zabbix reports and statistics, as well as configuration parameters, are accessed through a web-based frontend. A web-based frontend ensures that the status of your network and the health of your servers can be assessed from any location. (Zabbix, 2020).

2. *Nagios* - an open source monitoring system for computer systems. It was designed to run on the Linux operating system and can monitor devices running Linux, Windows and Unix operating systems (OSes). Nagios software runs periodic checks on critical parameters of application, network and server resources. For example, Nagios can monitor memory usage, disk usage, microprocessor load, the number of currently running processes and log files. Nagios also can monitor services, such as Simple Mail Transfer Protocol (SMTP), Post Office Protocol 3 (POP3), Hypertext Transfer Protocol (HTTP) and other common network protocols. Active checks are initiated by Nagios, while passive checks come from external applications connected to the monitoring tool(SearchITOperations, Nagios, 2020).

3.4 Build and Test

1. *Jenkins*- an open-source implementation of a Continuous Integration server written in Java. It works with multiple programming languages and can run on various platforms (Windows, Linux, and macOS). It is widely used as a CI (Continuous Integration) & CD (Continuous Delivery) tool. It has vast community support, and there are many plugins available for integrating it with other tools like Slack, GitHub, and Docker. Also, anyone can develop a Jenkins plugin and contribute to it. By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate a build and run tests to ensure the functionality is working fine. Jenkins supports the entire software development life cycle that includes building, testing, documenting the software and deploying (Singh & Krishnakumar, 2019).
2. *Gradle*- a project automation tool that builds upon the concepts of Apache Ant and Apache Maven and is licensed under the ASL. Introduces a Groovy-based domain-specific language (DSL) instead of the more traditional XML form of declaring the project configuration. Unlike Apache Maven, which defines lifecycles, and Apache Ant, where targets are invoked based upon a depends-on partial ordering; it uses a directed acyclic graph to resolve the order in which tasks can be run (Muschko, 2014).Gradle build scripts are written in Groovy, not

XML. But different other approach this is not for simply exposing the raw scripting power of a dynamic language. Its Architecture contains Deep API, Plugins, Build tools and Libraries. It is the first build integration tool, it supports Easy of migration, Groovy, wrapper, scales and Multiproject builds. Also including CD optimization, Performance tuning, Standardization, Plug-in development.

3. *CruiseControl* - this modularity allows users to install CruiseControl where it will best fit their needs and environment. Using remoting technologies (HTTP, RMI), it is possible to control and monitor the CruiseControl build loop. Those are turned off by default for obvious security reasons(CruiseControl, 2020).This tool is composed of 3 main modules:
    a. The build loop: core of the system, it triggers build cycles then notifies various listeners (users) using various publishing techniques. The trigger can be internal (scheduled or upon changes in a SCM) or external. It is configured in a xml file which maps the build cycles to certain tasks, thanks to a system of plugins. Depending on configuration, it may produce build artifacts.
    b. The jsp reporting application allows the users to browse the results of the builds and access the artifacts
    c. The dashboard provides a visual representation of all project build statuses.

3.5 Deployment and Configuration Tools

1. *Puppet* - along with Chef, are the most popular DevOps programs, according to RightScale's 2016 survey of DevOps trends. Puppet is used by 42 percent of businesses that use DevOps methodologies, followed closely by Chef with 37 %. Puppet is an open source software configuration management and deployment tool. It's most commonly used on Linux and Windows to pull the strings on multiple application servers at once. But you can also use Puppet on several platforms, including IBM mainframes, Cisco switches, and Mac OS servers. Like other DevOps programs, Puppet does more than automate system administration. It changes the human workflow, and enables developers and system administrators to work together. Programmers can write, test, and launch applications without waiting on Ops staff to deliver the resources needed. For example, Microsoft and Puppet recently partnered with the RISCO Group, an Israeli security project company, to create a Puppet and Azure Resource Manager-powered self-service web portal. The result, says Ido Vapner, RISCO Group's head of DevSecOps and technology, is a development workflow that enables the company to "provision an entire environment with a single click" instead of it taking "a week to build a new environment" (Vaughan-Nichols, 2017).
2. *Chef*- a configuration management and automation platform from Opscode. Chef helps to describe the infrastructure with code. Since infrastructure is managed with code, it can be automated, tested and reproduced with ease. Chef is a powerful automation platform that transforms complex infrastructure into code, bringing servers and services to life. Whether the user operating in the cloud, on premises, or a hybrid, Chef automates how applications are configured, deployed, and managed across your network, no matter its size. Chef is a thin DSL (domain-specific language) built on pinnacle of Ruby. This approach allows Chef to provide just enough abstraction to make reasoning about your infrastructure easy. Chef includes a built-in taxonomy of all the basic resources one might configure on a system, plus a defined mechanism to extend that taxonomy using the full power of the Ruby language. Ruby was chosen because it provides the flexibility to use both the simple built-in classification, as well being able to handle any customization path that organization requires (Chef, 2020).

3. RANCID - monitors a router's (or more generally a device's) configuration, including software and hardware (cards, serial numbers, etc.) and uses CVS (Concurrent Version System), Subversion or Git to maintain history of changes. RANCID does this by the very simple process summarized as:
    - login to each device in the router table (router.db), run various commands to get the information that will be saved, cook the output; re-format, remove oscillating or incrementing data, email any differences (sample) from the previous collection to a mail list, and finally commit those changes to the revision control system (RANCID, 2016).
4. *CFEngine* - an IT infrastructure automation and Continuous Operations framework that helps engineers, system administrators and other stakeholders in an IT organization manage IT infrastructure while ensuring service levels and compliance. It runs on the smallest embedded devices, on servers, in the cloud, and on mainframes, easily handling tens of thousands of hosts. It is available as both open source and commercial software. CFEngine combines modeling and monitoring to bring actual state into compliance with Desired State. You describe Desired State using a declarative, model-based or knowledge-oriented approach:

    a. Define the Desired State of your infrastructure with a special knowledge-oriented language; or, use tools that help you design a Desired State from off-the-shelf resources.
    b. Simulate configuration changes before committing to them, then study the impact of changes using the knowledge-based inference engine.
    c. Confirm the decided Desired State for an automatic self-healing implementation. CFEngine then continuously corrects configuration drift, keeping systems in compliance with their Desired State.
    d. Collect reports on the differences between Actual and Desired States and changes or failures encountered while implementing the Desired State using a highly compressed data stream(CFEngine, 2020).
5. *Ansible* - the Ansible architecture is agentless; it doesn't need a running daemon on the client side like Puppet does. Instead, the Ansible server logs in to the Ansible node and issues commands over SSH in order to install the required configuration (Verona, 2016). Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates. Ansible's goals are foremost those of simplicity and maximum ease of use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with an accelerated socket mode and pull modes as alternatives), and a language that is designed around auditability by humans – even those not well known with the program. This tool is appropriate for managing small setups with a handful of instances as well as enterprise environments with many thousands. Ansible manages machines in an agentless manner. Ansible is decentralized – it relies on your existing OS credentials to control access to remote machines; if needed it can easily connect with Kerberos, LDAP, and other centralized authentication management systems(Ansible Documentation, 2019), (Verona, Practical DevOps, 2016).

4. Ethical issues in DevOps

Nowadays, DevOps Engineering as a job is on rise continuously. In the United States the average salary of the DevOps Engineers is $93,605. This fact examines the importance of this technology in the software development industries. Coding is recognized as one of the core health information management (HIM) functions within healthcare. Due to the complex regulatory requirements affecting the health information coding process, coding professionals are frequently faced with ethical coding and coding-related challenges.

The Standards of Ethical Coding are important established guidelines for any coding professional and are based on the American Health Information Management Association's (AHIMA's) Code of Ethics. Both reflect expectations of professional conduct for coding professionals involved in diagnostic and/or procedural coding, data abstraction and related coding and/or data activities.

A Code of Ethics sets forth professional values and ethical principles. In addition, a code of ethics offers ethical guidelines to which professionals aspire and by which their actions can be expected and be judged. HIM and coding professionals are expected to demonstrate professional values by their actions to patients, employers, and members of the healthcare team, the public, and the many stakeholders they serve. A Code of Ethics is important in helping guide the decision-making process and can be referenced by individuals, agencies, organizations, and bodies (such as licensing and regulatory boards, insurance providers, courts of law, government agencies, and other professional groups).

The Code of Ethics is relevant to all AHIMA members, students, and CCHIIM credentialed HIM and coding professionals, regardless of their professional functions, the settings in which they work, or the populations they serve. All core health information coding activities are performed in compliance with state and federal regulations, and employer policies and procedures (AHIMA, 2016).There are 8 principles predefined in coding that each developer, especially in DevOps technology has to follow them. We will talk about each principle in details.

4.1 Principle I – Public

Software engineers shall act consistently with the public interest. Meaning that they should accept full responsibility for their own work and moderate the interests of the software engineer, the employer, the client and the users with the public good. Programmers should reveal every potential danger to the user, the public or the environment, which they believe it is related to the relevant software.
By following this principle, programmers should approve software only if they are sure that the software is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment.
It is recommended to cooperate in different matters of grave public concern caused by software, such as: its installation, support, documentation and maintenance. Be fair and avoid deception in all statements, particularly public ones and be encouraged to volunteer professional skills to good causes.

4.2 Principle II – Client and Employer

Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. Following this principle, the developer should use the property of a client or employer only in ways properly authorized, and with the clients or employer's knowledge and consent. Here you should not use software that is obtained or retained either illegally or unethical, with or without consciousness.

An important dimension here is the privacy. Developers should keep the privacy of any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

It is recommended for developers to identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client. In case of ethical concern is being compromised, inform the employer or another appropriate authority of the ethical concern.

4.3 Principle III - Product

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. There are some important elements regarding this principle such as: high quality product, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

Only this way you can ensure proper and achievable goals and objectives for any project on which you are working. The project managers must be ensured that developers are qualified for any project on which they work or proposes to work by an appropriate combination of education, training and experience. As a part of the project, you should follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.

Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects. This way, you should write documentation about all your work and strive to fully understand the specifications for software in progress. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized, maintain the integrity of data, being sensitive to outdated or flawed occurrences and most important treat all forms of software maintenance with the same professionalism as new development.

4.4 Principle IV – Judgment

Software engineers shall maintain integrity and independence in their professional judgment. This principle addresses various issues and situations where you as a DevOps Engineer or Developer can be judged for different unethical or illegal issues.

In this context, do not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices. Refuse to participate, as members or advisors, in a private, governmental or professional body concerned with software related issues, in which they, their employers or their clients have undisclosed potential conflicts of interest. Also, it is recommended to maintain professional objectivity with respect to any software or related documents which you are asked to evaluate.

4.5 Principle V - Management

Software engineering managers and leaders shall promote an ethical approach to the management of software development and maintenance. Meaning that they have to ensure good management skills for any project on which they are working. At this point, regarding security, the developers or software engineers should be informed about the employer's policies and procedures for protecting passwords, files and information that is confidential to the client or confidential to others. There should be fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed. It is important to note that nobody cannot be punished for expressing ethical concerns about a project.

4.6 Principle VI – Profession

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. This way they can help develop an organizational environment favorable to acting ethically and promote public knowledge of software engineering. Developers should be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful. In this context they should take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.

4.7 Principle VII – Colleagues

Software engineers shall be fair to and supportive of their colleagues. This principle defines definitions about colleagues, respectively the interaction between co-workers. Colleagues should be encouraged to adhere this principle, they should assist each other in professional development, credit fully the work of others and refrain from taking unethical credit.

By doing so, you assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general, in this way the interaction between co-workers will be more productive and the working process with be more efficient.

4.8 Principle VIII – Self

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. Developers always should try to improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time. Technology is moving and evolving fast, developers should be able to adapt with the newest tools in development environment, by trying to improve their ability to produce accurate, informative, and well-written documentation.

In fact, there are so many fields that developers in this context should improve themselves such as: Improve their understanding of the software and related documents on which they work and of the environment in which they will be used, improve their knowledge of relevant standards and the law governing the software and related documents on which they work.

Also, they must be careful not to give unfair treatment to anyone because of any irrelevant prejudices and not influence others to undertake any action that involves a breach of this principle (IEEE Computer Society, 1999).

5. The Attitude of Programmers about Ethical issues in DevOps

Based on the fact that the humans are the ultimate consumers of the software, definitely software developers should take ethics into consideration during software development lifecycle. Actually, any unethical software has consequences, depending on the audience of the software usage. The real question here is about the number of consequences, which varies from the number of users that consumes services of the software. There are a lot of ethical considerations a developer might face during the process of coding. Sometimes trying to help the company earn more money, developers forget about ethical considerations.

To understand the right statement of programmers about ethical issues, we have asked programmers to answer some questions in the survey posted online https://s.surveyplanet.com/O1XzJ4_j.

In the survey, developers are asked to answer some questions about Ethics in Development and Operating (DevOps). The participation in this survey is completely voluntary and there are no foreseeable risks associated with this project. All survey reposes are strictly confidential and all data in this survey are reported only in aggregate.

Survey participant's information

The total number of participants is 40, where 77.5% of the Developers that completed the survey are male and 22.5% of them are female.

Table 1. Participants Gender

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Male	31	77.5	77.5	77.5
	Female	9	22.5	22.5	100.0
	Total	40	100.0	100.0	

It is important to know the living city of the developers, because we know that the living style and culture differ from country to country. As related to the place the participants come from, as illustrated in figure 2, the living city of the developers who participated in the survey, most of them are from Tetovo (26), Skopje (6), Gostivar (3), Ankara (3) and Istanbul (2).

Table 2. Participants City

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Tetovo	26	65.0	65.0	65.0
	Skopje	6	15.0	15.0	80.0

	Gostivar	3	7.5	7.5	87.5
	Istanbul	2	5.0	5.0	92.5
	Ankara	3	7.5	7.5	100.0
	Total	40	100.0	100.0	

Regarding to the level of education, 75% of the Developers are with Bachelor degree, 22.5% High School and 2.5% Master degree (figure 3). Whereas 45% of them are Front-end Developers, 33% of them Back-end and 22.5% (figure 4).

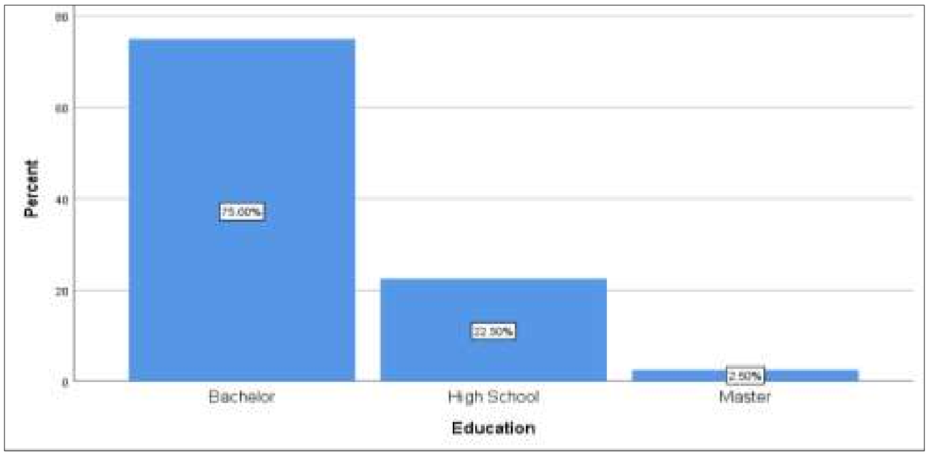

Figure 3. Participants Education

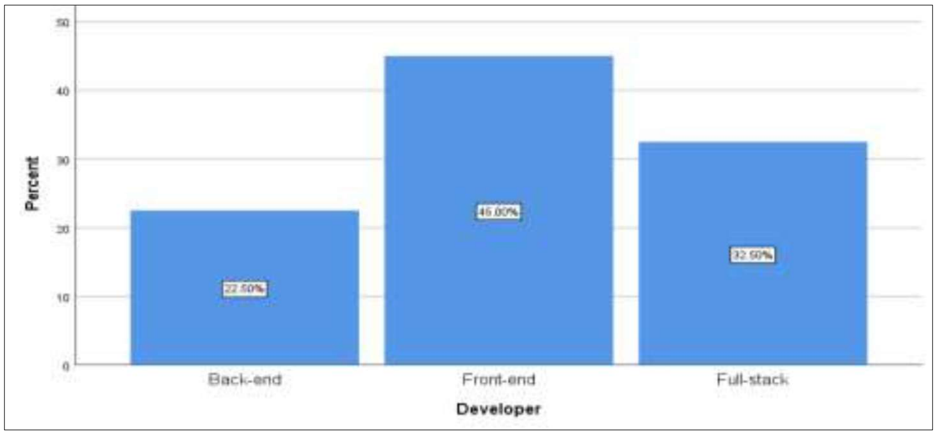

Figure 4. Participants Developer Type

It is important to know some basic information about developers like: What do they use coding for? Do they test the code before deploying? In fact, most of them used coding both for hobby and work (57.5%). Some of them use coding just for work (32.5%) and the minority use coding just for hobby (10%).

Table 3. Code Usage Reason

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Hobby	4	10.0	10.0	10.0
	Hobby & Work	23	57.5	57.5	67.5
	Work	13	32.5	32.5	100.0
	Total	40	100.0	100.0	

Hypothesis

In order to achieve a better insight regarding the attitude of the developers in the direction of the ethical issues, two hypotheses were raised as follows.

Hypothesis 1: IT Developers with higher level of education have more positive ethical attitudes towards those with a lower level of education.

Hypothesis 2: Back-end developers have more positive ethical attitudes than those who develop Front-end and Full-stack.

Table 4 presents the descriptive data of the main variable named *ethical attitudes* of IT programmers. It shows that the obtained average is M = 11.80, the minimum achieved value is 8, the maximum value is 14 and the standard deviation is 1.36 To summarize, this demonstrates that IT programmers in general have positive ethical attitudes.

Table 4. Main Variable Assessments

	N	Minimum	Maximum	Mean	Std. Deviation
Ethic	40	8,00	14,00	11,8000	1,36250
Valid N (listwise)	40				

Table 5 presents the differences in the ethical attitudes of IT programmers according to the level of education. Results show that programmers with higher level of education (Bachelor in this case) KW = 22.10 have more positive ethical attitudes than those who have lower level of education (High School) KW = 13. These differences were statistically significant with sig: 0.031 according to Kruskal Wallis Test (table 6). Therefore, the first hypothesis: Those who have a higher level of education have more positive ethical attitudes as opposed to those who have a lower level of education, are verified.

Table 5. Differences in the ethical attitudes of IT programmers according to the level of education

Education		N	Mean Rank
Ethic	Bachelor	30	22,10
	High School	9	13,00
	Total	39	

Table 6. Test Statistics[a,b]

	Ethics
Kruskal-Wallis H	4,660
df	1
Asymp. Sig.	,031
Exact Sig.	,031
Point Probability	,004

a. Kruskal Wallis Test

b. Grouping Variable: Education

Table 7 shows the differences in the ethical attitudes of IT programmers based on the type of developer, whether it is Back-end, Front-end or Full-stack. The outcomes according to Kruskal Wallis Test illustrate that there are no statistically significant differences with sig = 0.739 (table 8), hence the hypothesis second that: Back-end IT developers have more positive ethical attitudes than those that develop Front-End and Full-stack, the hypothesis is not verified and the same is rejected.

Table 7. Differences in the ethical attitudes of IT programmers according to the type of developer (Back-end, Front-end, Full-stack)

Developer		N	Mean Rank
Ethic	Back-end	9	17,94
	Front-end	18	21,03
	Full-stack	13	21,54
	Total	40	

Table 8. Test Statistics[a,b]

	Ethics
Kruskal-Wallis H	,604
df	2
Asymp. Sig.	,739

a. Kruskal Wallis Test

b. Grouping Variable: Developer

5. Conclusion

DevOps is a set of practices that combines software development (Dev) and information-technology operations (Ops) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. In fact, there are a lot of reason that causes unethical code:

- **For fun -** As we mentioned in section above, there is a number of reasons that can accomplish unethical code. For example, writing a code that can infect the user's PC just for fun. In this case the user can be our friend, which trusts in us, and we do this action just for joking him/her, but we can get so many sensitive information about them with or without intention.
- **Business benefits -**Nowadays, many companies engage Developers to write code that can increase the benefits of organization, but without worrying about user's privacy. There are

a number of organizations that collect a lot of data from you. They do it in ways you are very likely unaware of. And they collect data that you are not aware they collect. They use that data for purposes you are not fully aware of and may or may not do it in your interest. They may or may not share that data with other entities, including governments and governmental agencies.

- **Malicious intentions -** usually this kind of actions are performed by hackers when they attack you and your privacy. Whether you're attacked today or tomorrow, it's important to understand the motivation and objective of your intruders — doing so can help you devise an appropriate defense. Malicious hackers can, in fact, be broken out under some broad classifications like: Cyber criminals, Spammers and adware spreaders, corporate spies, advanced persistent threat (APT) agents etc.

Based on the results of the survey, we can conclude that a huge part of the participants developers in the survey even thou have a positive attitude towards ethical issues, they do not have much information about it, which means that there is a huge potential risk on causing unethical actions in coding. This means that chances of spreading unethical code are high with whom will be affected the corporations, small businesses and the whole society.

The right solution for this issue would be working on raising awareness about unethical code, by informing people about them and their consequences. This process can be done by starting from the learning institutions like: schools, faculties, educational training centers, corporate training centers, where they should inform their students or employers all about this issue and how to get away from this phenomenon. Only this way we can improve the process of Development in Software Engineering, especially in DevOps, by taking care of the people's most sensitive thing which is the privacy.

References

[1]. AHIMA. (2016). Retrieved January 10, 2020, from American Health Information Management Association Standards of Ethical Coding: http://bok.ahima.org/CodingStandards#.Xg8R10dKiUk

[2]. *Amazon Web Services*. (2020). Retrieved January 10, 2020, from https://aws.amazon.com/elasticsearch-service/the-elk-stack/kibana/

[3]. Anicas, M. (2014, June 11). *DigitalOcean*. Retrieved January 11, 2020, from How To Use Logstash and Kibana To Centralize Logs On Ubuntu 14.04: https://www.digitalocean.com/community/tutorials/how-to-use-logstash-and-kibana-to-centralize-and-visualize-logs-on-ubuntu-14-04

[4]. *Ansible Documentation*. (2019). Retrieved January 10, 2020, from https://docs.ansible.com/ansible/latest/index.html

[5]. *CFEngine*. (2020). Retrieved January 10, 2020, from https://cfengine.com/product/what-is-cfengine/

[6]. *Chef*. (2020). Retrieved January 10, 2020, from An Overview of Chef Infra: https://docs.chef.io/chef_overview/

[7]. CruiseControl. (2020). Retrieved January 10, 2020, from http://cruisecontrol.sourceforge.net/overview.html

[8]. Davis, C. (2014). Graphite Documentation Release 0.10.0.

[9]. Davis, J., & Daniels, R. (2016). *Effective DevOps - Building a Culture of Collaboration, Affinity, and Tooling at Scale.* California, USA: O'Reilly Media.

[10]. *Ganglia Monitoring System*. (2018). Retrieved January 10, 2020, from http://ganglia.sourceforge.net/

[11]. *IEEE Computer Society*. (1999). Retrieved January 10, 2020, from IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices: https://www.computer.org/education/code-of-ethics#:~:targetText=PROFESSION%20
%20Software%20engineers%20shall%20advance,consistent%20with%20the%20public%20interest.&targetText=SELF%20-%20Software%20engineers%20shall%20participate,the%20practice%20of%20the

[12]. *Jan Bask Training*. (2019, March 15). Retrieved January 10, 2020, from What is DevOps Lifecycle? Devops Lifecycle Phases Plan & Measure: https://www.janbasktraining.com/blog/what-is-devops-lifecycle/

[13]. Muschko, B. (2014, July 8). *Why Build Your Java Projects with Gradle Rather than Ant or Maven?* Retrieved january 10, 2020, from Dr.Doobs: https://www.drdobbs.com/jvm/why-build-your-java-projects-withgradle/240168608

[14]. *RANCID*. (2016). Retrieved January 10, 2020, from Really Awesome New Cisco confIg Differ: https://shrubbery.net/rancid/

[15]. SearchITOperations. (2020). Retrieved January 10, 2020, from https://searchitoperations.techtarget.com/definition/Loggly

[16]. SearchITOperations. (2020). *Nagios*. Retrieved from https://searchitoperations.techtarget.com/definition/Nagios

[17]. *Sensu Go*. (2020). Retrieved January 10, 2020, from https://docs.sensu.io/sensu-go/latest/

[18]. Singh, L., & Krishnakumar, P. (2019, August 2). *Tutorial on Continuous Integration with Jenkins*. Retrieved January 10, 2020, from BrowserStack: https://www.browserstack.com/guide/continuous-integration-with-jenkins-tutorial

[19]. *Splunk*. (2020). Retrieved January 11, 2020, from https://www.splunk.com/en_us/about-splunk.html

[20]. Vaughan-Nichols, S. (2017, May 10). Retrieved January 10, 2020, from Helwett Packard Enterprise: https://www.hpe.com/us/en/insights/articles/what-is-puppet-and-why-should-you-consider-it-for-your-cloud-and-servers-1705.html

[21]. Verona, J. (2016). *Practical DevOps.* Packt Publishing.

[22]. Verona, J., Duffy, M., & Swartout, P. (2016). *Learning DevOps: Continuously Deliver Better Software.* Birmingham, UK: Packt Publishing.

[23]. Zabbix. (2020). *Documentation 2.0*. Retrieved January 10, 2020, from https://www.zabbix.com/documentation/2.0/manual/introduction/manual_structure