university of groningen

faculty of science and engineering

Cost awareness in pull requests of open source repositories

Abel van der Til

university of groningen

faculty of science and engineering

University of Groningen

Cost awareness in pull requests of open source repositories

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computing Science
at the University of Groningen under the supervision of
Prof. V. (Vasilios) Andrikopoulos (Computing Science, University of Groningen)
and
dr. D. (Daniel) Feitosa (Computing Science, University of Groningen)

Abel van der Til (S4091469)

June 10, 2024

Abstract

When managed properly, hosting applications in the cloud can be affordable. However, cloud hosting can quickly become very expensive, for example when developers configure their cloud environment to require more than they need. This configuration can be done with the use of Terraform files, this is a specific infrastructure configuration file that is stored as code.
In a previous research by Feitosa et al. [5], they looked for cost awareness in commits that modified these Terraform files. We will expand on this by researching pull requests that modify Terraform files. Through the use of Mining Software Repositories (MSR) we collect pull requests that make use of cost related keywords in the pull request or their containing commits. These pull requests will then be labeled based on the human written text.
From this we find out that pull requests do not efficiently demonstrate cost awareness for most repositories, but do show more insights on cost awareness compared to commits. Furthermore, reviews on pull requests lead to cost awareness in at least 23% of the cost related pull requests. Finally, pull requests can be used as an intermediate step to find otherwise difficult to find cost related commits.

Contents

1 Introduction **4**
- 1.1 Infrastucture as Code . . . 4
- 1.2 Prior work . . . 5
- 1.3 Related work . . . 5
- 1.4 Scope . . . 6

2 Study Design **7**
- 2.1 Methodology . . . 7
- 2.2 Labels . . . 7
- 2.3 Dataset update . . . 8
- 2.4 Pull requests . . . 9

3 Results **12**
- 3.1 Commits update . . . 12
- 3.2 Descriptive statistics . . . 12
- 3.3 Keyword locations . . . 13
- 3.4 Labels . . . 15

4 Analysis **19**
- 4.1 Descriptive statistics . . . 19
- 4.2 Keyword locations . . . 20
- 4.3 Commit and pull request linking . . . 21

5 Discussion **22**
- 5.1 Cost awareness in pull requests . . . 22
- 5.2 Cost awareness in code review . . . 22
- 5.3 Finding new commits . . . 23
- 5.4 Linked dataset . . . 23
- 5.5 Limitations & Future research . . . 23

6 Conclusion **25**

A Additional Data **27**

List of Figures

2.1 Cost awareness labels 8
2.2 Steps required to update commit dataset 9
2.3 Steps required to label pull requests 9

3.1 Label frequency of commits 13
3.2 Pull request label frequency 16
3.3 Co-occurrences of labels 16
3.4 Frequency of sets of labels 17
3.5 Label frequency of pull requests linked with commits 18
3.6 Label frequency of commits in unrelated pull requests 18

4.1 Infracost example 20

List of Tables

3.1 Keyword frequency per location 14
3.2 Frequency of empty locations 14
3.3 Frequency of awareness per location. 14

A.1 Label definitions 27

1 | Introduction

When hosting an application, costs are an inevitable consequence. In the past, one had to purchase and manage a dedicated server, which was both costly and complex [3]. In recent years Cloud computing has become more popular. This allows for many websites to be hosted on a network of servers. This way it is not required for a application owner to pay for the whole server and does not necessarily need the technical expertise that one needs to host a dedicated server [3]. Cloud providers often make use of the pay-as-you-go model [8], this means that one only pays for what one consumes. Think of what type of processor, how much system memory etc. This can save a lot of money, compared to dedicated hosting, as it is easier to quickly scale up and down. However, it is also possible for cloud computing to become expensive. For example whenever the cloud instance is ill configured. When one asks for more than the application needs, one needs to pay for it too. So for example when one's instance is configured for 128GB of RAM, while only ever needing 32GB, you still end up paying for the full 128GB.

This begs the question whether developers are aware of the costs of hosting on the cloud and whether they act to reduce costs. By finding out whether developers are aware and what changes are commonly made to achieve this, we could help other developers to reduce their costs.

1.1 Infrastucture as Code

Infrastructure as Code (IaC) uses a high-level descriptive coding language to automate the provisioning of IT infrastructure [6].

This means that developers can create a machine readable file, like YAML, that describes how an application is deployed. This deployment is automatic, and does not need any user interaction [2]. For example, this file contains data on what kind of virtual machine the application will run on [2], which determines the CPU and amount of system memory. Changing this file has great implications of the costs of the application. These artifacts are stored in the source code of the application and thus are part of the version control. This means they can only be modified through the use of commits. IaC is the name of the process, which is then implemented by a tool. There are many different of such tools, but common examples are Ansible and Terraform [6].

1.2 Prior work

Previously, a single study by Feitosa et al. [5] has looked into cost awareness on the cloud. This study is based on GitHub issues and commits messages where Terraform files have been modified. They only focussed on Terraform as it was widely adopted and focusing on multiple tools would be too complex. They looked for any commit that changed a Terraform file and extracted the commit message. This way, many commits were found that talk about changing these files because of cost related reasons. Issues did not seem to contribute as much, but did allow them to look at the decision making process. Combining the changes and the decision making process is not trivial, as commits are rarely linked to issues. The study proposes to look at pull requests as an attempt to get code changes and the decision making process together.

1.3 Related work

There are no papers that specifically look into cost awareness in pull requests. There are however many studies that use MSR to retrieve data from pull requests. One of these studies was performed by Kononenko et al. [7]. In this paper they looked at code reviews in pull requests in the repository of Shopify and its forks. In these pull requests they looked at the way these pull requests were merged with the main code base and the time it took to merge. They collected meta data about each pull request, as well as the description and comments. These pull requests have been labeled on the different merge types. The labeling was independently done by two authors and afterwards compared. The remainder of the study is not relevant for this paper.

In another study by Bertoncello et al. [1], they looked at the relative contributions to open source software by so-called casual contributors. These are developers who often contribute a single time to the project. This study is based on another similar research where this was first researched by mining commits. Similarly to our paper, the study improves on this by using the same methodology but also applied to pull requests. They specifically compared the commit and pull request approach for collecting data. For this they used the same set of repositories as before. With this, they managed to find more casual contributors with pull requests than with commits. Furthermore, they found less false negatives with the use of pull requests. Even though the type of data retrieved from this study (contributors) is different than the data required for this paper (natural language), it shows that by collecting data from pull requests instead of commits more relevant data can be collected and that the data can be more accurate.

1.4 Scope

As stated before, the previous study by Feitosa et al. [5] focused on cost awareness in commit messages. This paper expands on that same idea in the form of cost awareness in pull requests. Pull requests have certain advantages over commits, namely that they can contain multiple (related) commits. This is useful, as before commits are easily missed if they do not contain a keyword. If a pull request is relevant, all commits have an increased chance of being relevant as well. Next off, when creating a pull request, the author has the option of writing a more elaborate description than a commit message of why those commits should be added to the main code base. Furthermore, it also allows other developers to voice their opinion and start a discussion. This makes it ideal for finding cost related changes together with the reasoning why these modifications are made. This allows us to see whether developers create new pull requests in order to reduce costs or if it is of secondary importance. This paper will only focus on Terraform files specifically, as it is a continuation of the previous paper.

In order to find out whether pull requests can be used as a tool to find developers' awareness of costs, the following 4 research questions have been formulated:

RQ1: Do pull requests show cost awareness, and to what extent?
RQ2: Do pull request review comments demonstrate cost awareness, and to what extent?
RQ3: Do pull requests lead to commits that would not have been caught using the methods in the previous study, and how often?
RQ4: Does the linked data alter or strengthen the conclusion of cost awareness, compared to the previous study?

In Chapter 2 the design of this study is laid out. After which, the results and analysis are discussed in Chapter 3 and Chapter 4, respectively. At the end there is Chapter 5 for the discussion, which answers the research questions formulated above in detail and finally the conclusion in Chapter 6, which summarizes most important takeaways of the paper.

2 | Study Design

In this chapter, we go over how the research is performed. This includes methods, but also explanation of the different tasks, including their steps, that need to happen in order to get to the results.

2.1 Methodology

In order to find meaningful data for cost awareness in pull requests we make use of MSR (Mining Software Repositories) on the same repositories that were used in the previous study [5], which can be found in their repository [4]. As the source of the labels is the same in both cases, it allows us to analyse the combination of the datasets from the two studies. The initial study [5] has already composed a list of any repositories that are used for their dataset, this list is also used for the current research. Furthermore, the methods to retrieve relevant pull requests are heavily based on the methods to retrieve the relevant commits [5]. We look for human written text on pull requests that modify a TF file. The keywords that are used to search for pull requests are used exactly in the same way as for commits. They are written in the shortest form, such that any variations of the words are also captured. These keywords are: 'bill', 'cheap', 'cost', 'efficient', 'expens' and 'pay'.

2.2 Labels

In order to gather data, we label each pull request. The labels are mostly the same labels that were also used for labeling commits [5]. Some labels to further define another label are added. These are 'nat' and 'vpn' for networking, 'cpu' and 'ram' for instance. Furthermore, 'test' and 'change' have been added to distinguish the action of the change.

The labels are separated in 3 different levels, as can be seen in Figure 2.1.

First we have the effect of a pull request. As the name suggests, it means what the effect of this pull request is on the costs of the application. Awareness in itself does not have a direct effect on costs, but is still added to show that the developer is cost aware.

Next off, we have the action. This shows whether the change is a temporary

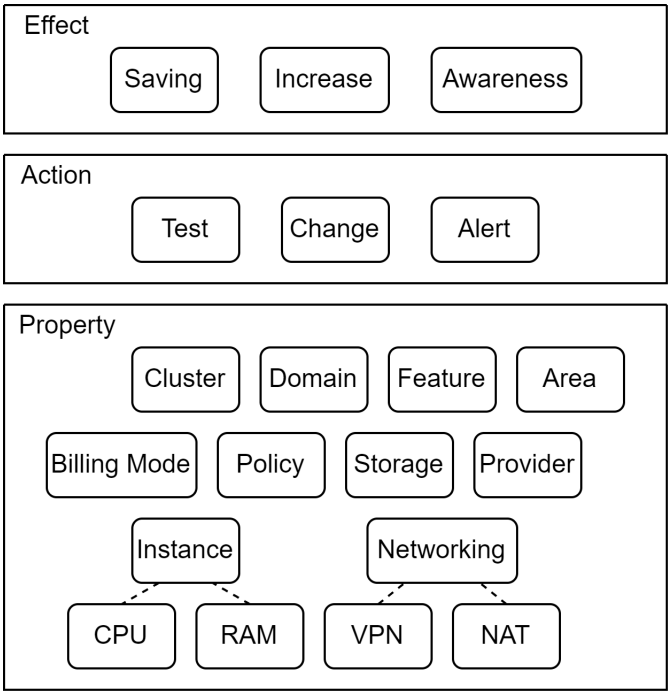

Figure 2.1: Cost awareness labels in different categories. Labels that further describe another label are linked with a dotted line.

test or whether it is a (semi) permanent change. Alert is used when costs are not changed, but cost monitoring is changed.

Finally, we have the property labels. These labels are to define what the specific cost related changes are, according to the description. Because of this, it is the largest set of labels. An example: the label is 'instance' is awarded in case a cheaper instance type is used. Some of these labels are used to further define what or why it is changed, for example the instance had too little memory. Then it would receive both 'instance' and 'ram'.

All the definitions of each label are described in the appendix in Table A.1.

Note that pull requests also receive an additional label to demonstrate where awareness is first mentioned. This can be either 'body', in case awareness is found in either the title or description. In case awareness is found in a review comment it receives the label 'comment'. As these labels are more meta data, they are not treated as other labels. This means they are not included in any tables or graphs unless otherwise specified.

2.3 Dataset update

As the pull requests are retrieved after the latest included commit of the previous dataset, it is important that the commit dataset is also updated. This because it is possible that commits that are relevant, but not reviewed are part of a pull request. In order to update the dataset, all commits are retrieved from the known set of TF repositories used in the previous study [5].

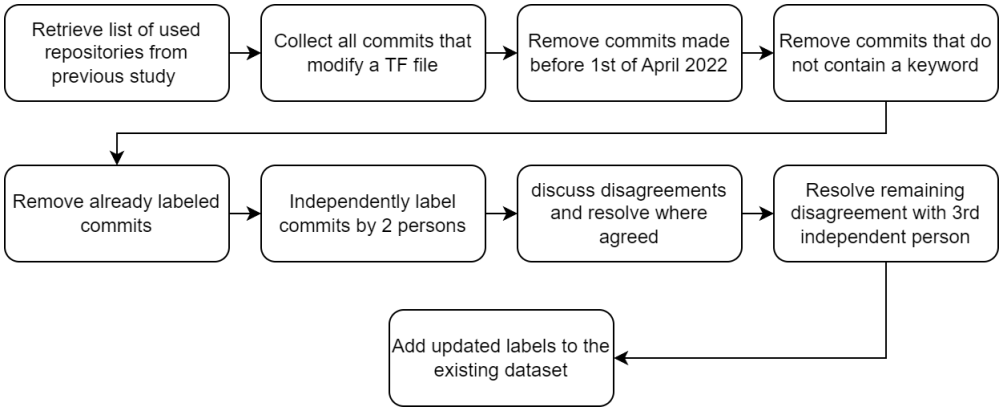

Figure 2.2: Steps required to update commit dataset

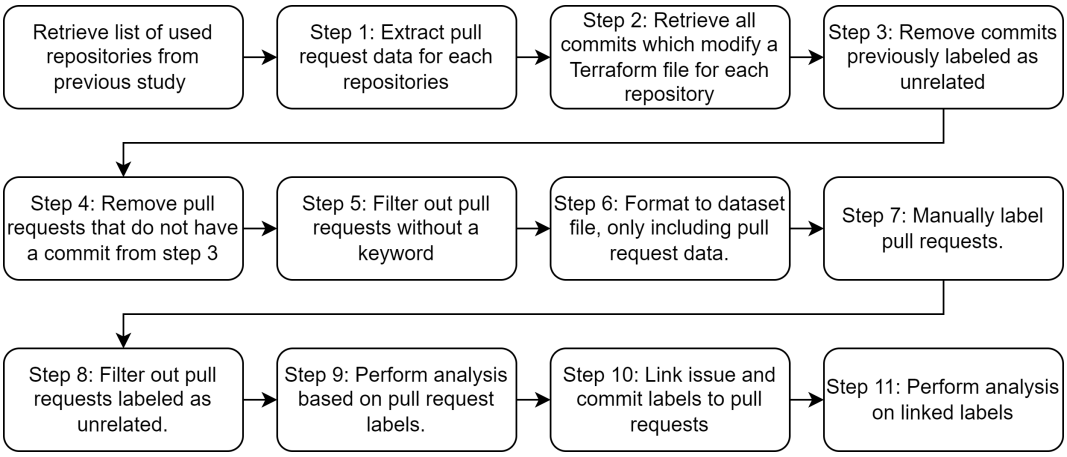

Figure 2.3: Steps required to label pull requests

All commits that are created after the 1st of April 2022 are filtered on whether they include a cost keyword as defined in Section 2.1 and modify a file ending with '.tf' or '.tf.json'. From the resulting list of commits, any commit that has already been labeled is excluded, the remaining are manually labeled. The labeling process is done separately by 2 reviewers, after which any disagreements are discussed and resolved when both reviewers agree. Any remaining disagreements are resolved by a 3rd person, i.e. one of the supervisors of this work. See Figure 2.2 for a graphical overview of the process.

2.4 Pull requests

Since the data is collected from GitHub it is likely that not all repositories still exist. Any repository that no longer exist, is therefore not be inspected for any pull requests. Furthermore, not all repositories make use of pull requests, therefore these are also not included.

As mentioned previously, the same list of repositories as for the commits are used to find pull requests, which are stored in the repository of the previous study [4]. The steps to find relevant pull requests are described below. See

Figure 2.3 for a graphical summary.

- Step 1: Scrape pull request information for each repository in this list from the GitHub API, this is then stored in a JSON file. For each pull request, the url, title, description, review comments and the hash for each commit is retrieved. It is important to note that only a maximum of 250 commits per pull request can be retrieved, due to limitations of the GitHub API. Any missing commits are therefore ignored in the next steps.

- Step 2: Retrieve the commits of the repositories that still exist (as of May 2024) and have at least 1 pull request. Store all commits that modify a TF file for later use.

- Step 3: Remove reviewed, but unrelated commits from the previous study [5] from the list of step 2. Commits which have been reviewed but did not receive a cost related label are said to be unrelated, either because it was a false positive or because it does not demonstrate cost awareness. Any commit that has been reviewed and is found to be relevant remains in the list, just like any commit that has not been reviewed. This step is not strictly required but removes unnecessary commits, which reduces work later on.

- step 4: Step 1 and 3 are combined; Ignore all commits for each pull request that does not appear on the list from step 3. This means that only commits remain that modify a TF file and are not found to be unrelated. If there are no more linked commits, the pull request is excluded. This because it is known that all the commits of the pull request are not relevant, and therefore assumed that the pull request is also not relevant.

- Step 5: Filter out any pull request that does not have a keyword in either the title, description, any review comment or any commit message. The place of where the keyword is found is stored.

- Step 6: Format the remaining pull requests to a dataset format comparable to that of the previous study [5]. For the content, only the pull request title, content and (review) comments are included. The commit messages are not included, as they should not influence the labeling of the pull request.

- Step 7: Label each pull request by giving it a label from the predetermined list in Table 2.1. The labeling is done by looking at text that is written by humans in the pull request title, description and review comments. This means that any text that is clearly automatically generated is not used to award labels to said pull request. As mentioned before, the commit message is not used to assign a label. Another label is added that demonstrates where cost awareness is demonstrated for the first time. This can be either in the body / title of the pull request, or in the review comments.

Due to the project being done alone, a single person does the labeling.

- Step 8: Remove any pull request that was deemed to be unrelated.
- Step 9: Perform statistical analysis. Results show how often pull requests are found to be relevant, as well as how often code reviewers initiate cost awareness. These results are required to answer RQ1 and RQ2.
- Step 10: Link pull requests labels to the labels of their already labeled commits.
- Step 11: Perform analysis on the linked datasets. This shows how many commits are contained in the relevant pull requests, which in turn answers RQ3. Furthermore, with the connected data we can answer RQ4.

3 | Results

This chapter will display the results gathered from the study in separate sections. Sections are separated by what kind of data is collected. This chapter will only show results, the analysis on this data will be performed in the next chapter.

3.1 Commits update

The initial dataset of commits contains 538 labeled commits. Between April 2022 and May 2024, a total of 282 new commits from the same repositories have been found. Out of which 68 relevant commits have been added to the dataset. For the new commits, some new labels have been included, as described in Section 2.2. The initial dataset does not contain these new labels. Therefore, in the following, these labels have not been included in figures where commits are used. In Figure 3.1 one can see the frequency of labels in the updated commit dataset.

3.2 Descriptive statistics

Out of the 1278 initial repositories, only 610 (48%) still exist and make use of pull requests. From these, 469 (77%) have pull request(s) with a TF changing commit. Out of these, 203 (43%) repositories have a pull request that has a keyword in either the pull request itself or one of its commits. That is 33% of the existing repositories with pull requests.

Within these 203 repositories, a total of 33797 pull requests which modify a TF file exist, of which 899 pull requests contain a cost-relevant keyword (2.7%). Out of these pull requests, 249 (27.7%) have received a label, and thus are found to be relevant to this study. This is 0.0074% of the total amount of TF pull requests from these repositories.

All 249 pull requests that received a label are from the same 100 repositories, which means that 103 of the 203 repositories do not show cost awareness in their TF pull requests that contain keyword(s).

All the pull requests that modify a TF file have 130720 commits combined. Of these, 3816 (3%) modify a TF file. 754 (20%) of these commits are part of a

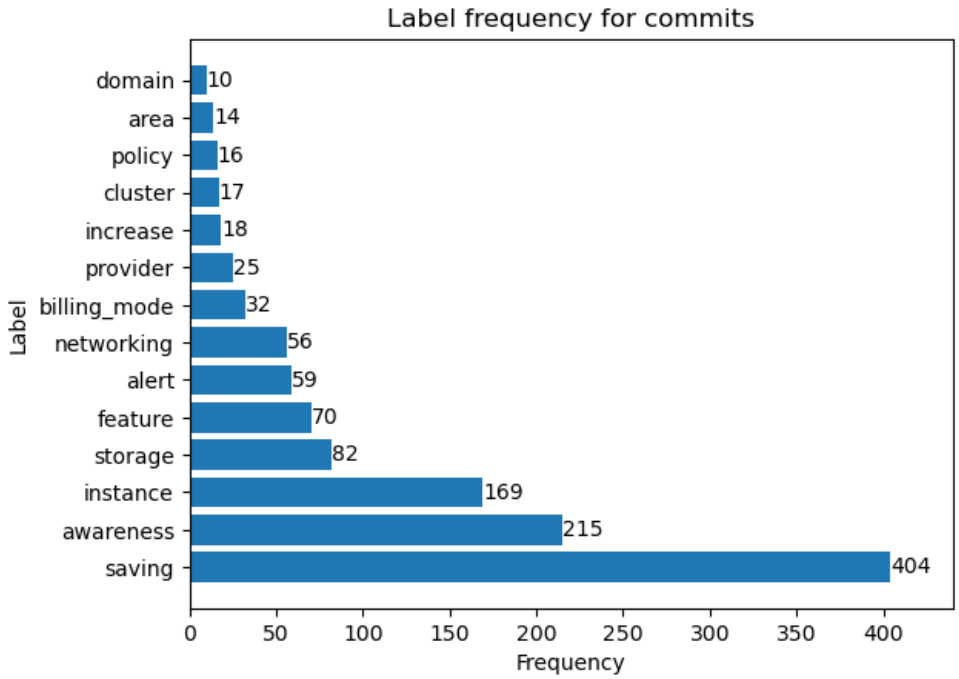

Figure 3.1: Label frequency of 606 commits that modify a Terraform file and show cost awareness. These commits are combined from the dataset of the previous study and the commits that have been added since the commits update.

labeled pull request of which 203 (27%) contain a keyword.

3.3 Keyword locations

As mentioned before, there are 4 locations in which keywords can be found. These are the pull request title, pull request description, pull request (review) comments and commit messages. Table 3.1 shows the frequency of keywords in each location. Pull requests can have keywords in multiple locations, all of these are counted. Not all parts of a pull request are always used. Table 3.2 shows how often each location is left empty. Commit messages were not investigated in this case, as this text was not examined while labeling. It does not mean that there is always awareness in a given location, even if that pull request is deemed to be relevant. In Table 3.3 one can see where awareness was first demonstrated. The title and description were combined, as the description did not always exist and also because the title and description are often similar and they are both written by the author.

Location	Any location (relevant)	Single location (relevant)
Title	111 (47)	28 (4)
Description	363 (161)	211 (59)
Comment	450 (70)	407 (54)
Commit message*	194 (122)	62 (7)

Table 3.1: Keyword frequency per location and frequency of relevance out of 899 pull requests. Keywords can appear in multiple locations. Whenever one or multiple keywords are only found in a single location of a pull request, its location is counted in the 'single location' column. The location is always counted in the 'any location' column, even if keywords also appear in other locations. For each case, the frequency of pull requests that have been found relevant is stated between brackets.
* Note: unlabeled but reviewed commits have already been filtered out.

Location	frequency
Title	0
Description	31
Comment	93

Table 3.2: Amount of pull requests that have no text in a location. All of these pull requests modify a Terraform file and have a keyword.

Location	frequency
Description / Title	192
Comment	57

Table 3.3: Frequency of awareness per location. Denotes the location of a pull request where cost awareness was demonstrated first.

3.4 Labels

A total of 783 labels are assigned between 249 pull requests, with a maximum of 5 labels for a single pull request and an average of 3.14 labels per pull request. Note, this does not include the label for location of awareness. Figure 3.2 shows the frequency of each label. Figure 3.3 shows the frequency of co-occurrences of different labels within pull requests. Figure 3.4 shows all combinations of labels in a pull request, combined with the frequency.

There are 166 pull requests that have labeled commits. 109 of these pull requests are labeled, the remaining 57 pull requests are reviewed but not labeled. When comparing the labels of pull request with the set of the labels of their commits, we have to separate the labeled and unlabeled pull requests. If we would directly compare the labels of all pull requests with their commits, the commits would be over represented. This because unlabeled pull requests do not, as the name suggests, have any labels. In Figure 3.5 one can see the intersection of the label frequency of related pull requests and their commits. In Figure 3.6 one can see the frequency of labels in commits of unlabeled pull requests. For these two figures, only labels that were also used in the initial commit dataset are shown.

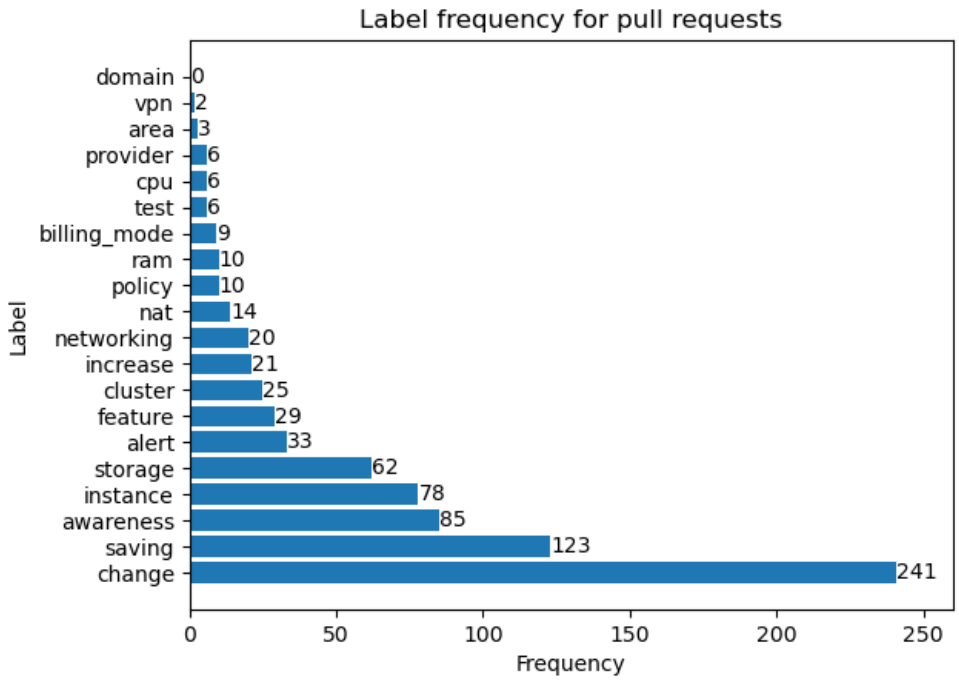

Figure 3.2: Label frequency of 249 pull requests that modify a Terraform file and show cost awareness.

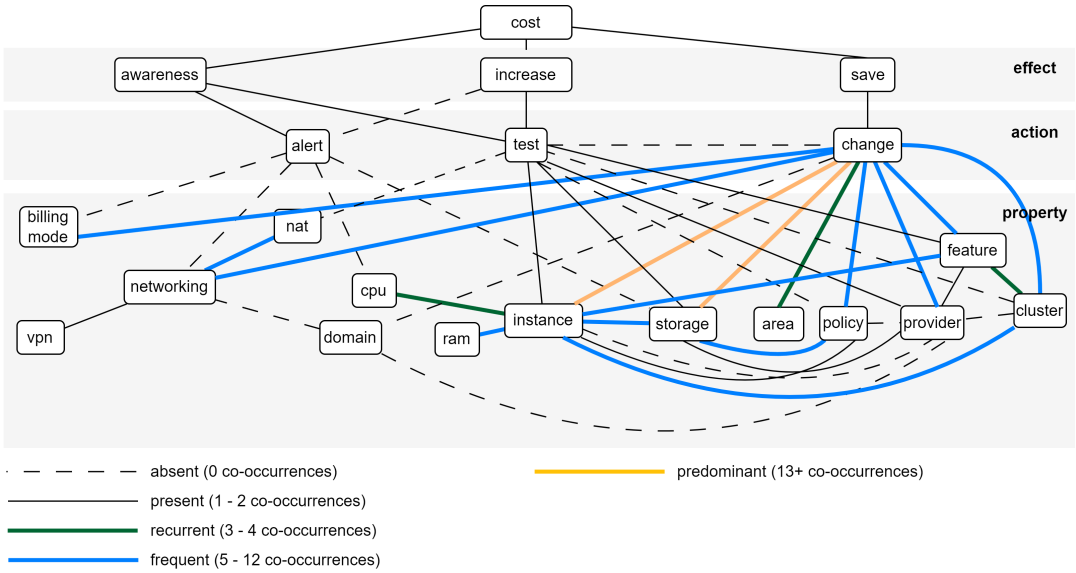

Figure 3.3: Knowledge graph describing the frequency of co-occurrences of labels within a pull request that modifies a Terraform file and shows cost awareness.

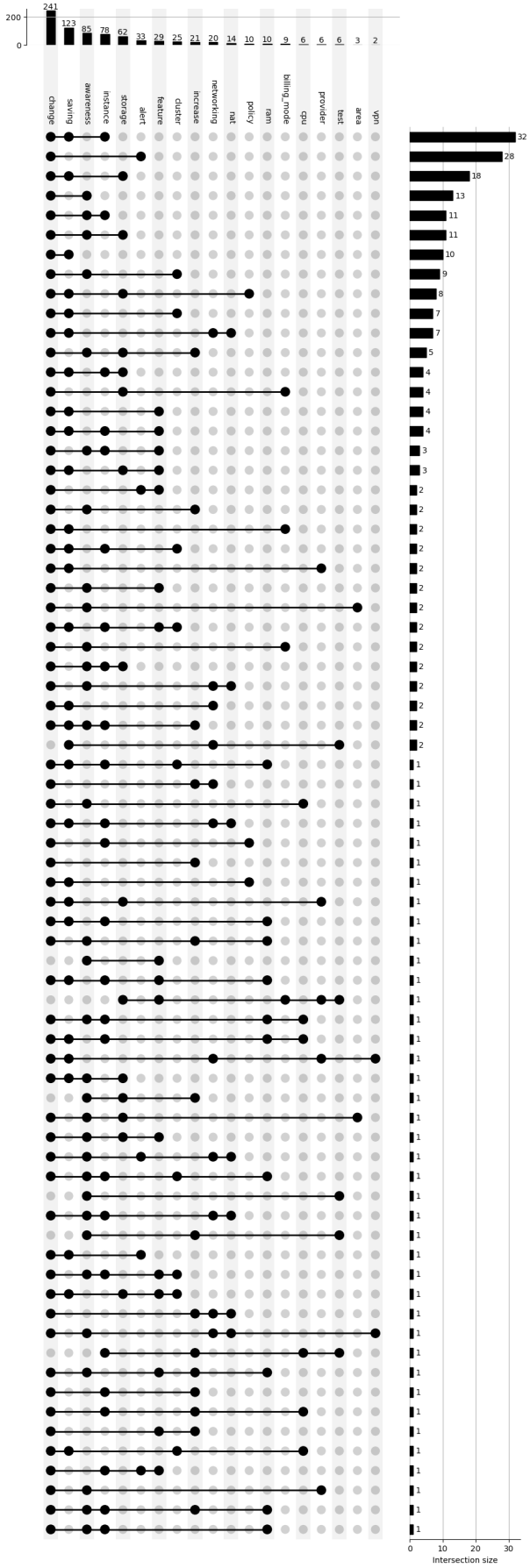

Figure 3.4: Upset plot showing the frequency of sets of labels in the dataset of relevant pull requests.

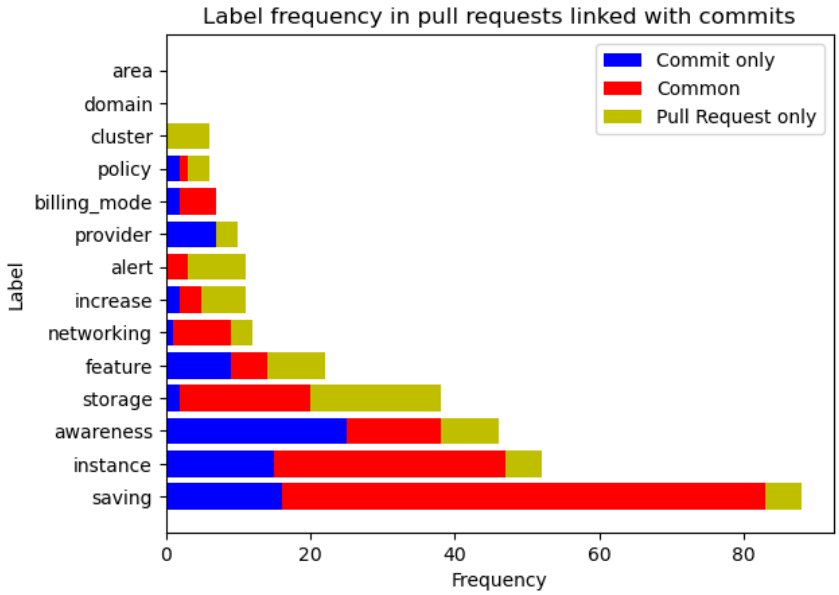

Figure 3.5: Label frequency of 109 labeled pull requests linked with their commits. The set of labels of all commits of a pull request are compared to the set of labels of the pull request itself. If label is both in set of pull request and commit labels, it is a common label. Otherwise the label is either commit only or pull request only.

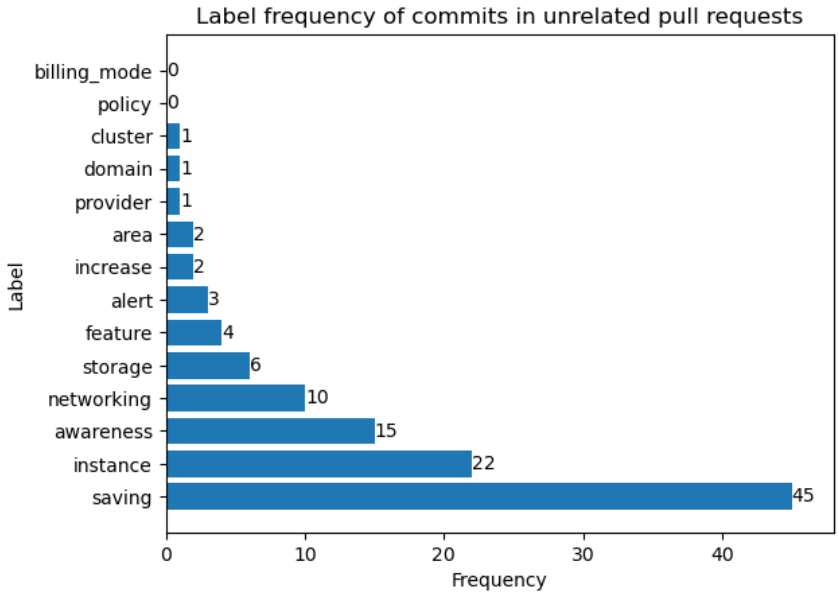

Figure 3.6: Label frequency of commits in 57 pull requests that were reviewed, but did not demonstrate cost awareness in itself.

4 | Analysis

In this chapter we will analyse the results from the previous chapter. For each section from Chapter 3 we will interpret the data and explain it in its own section in this chapter.

4.1 Descriptive statistics

As can be seen in Section 3.2 the majority (77%) of the repositories that make use of pull requests, have modified their TF file using a pull request. However, less than half of those also contain a cost-related keyword. Possible reasons for this can be:

- developers do not explain why changes are made.
- developers do not explicitly mention cost related keywords while explaining, but use more general words.
- developers document their reasoning on an external websites like Trello.
- developers do not make modifications for cost saving reasons.

In the case of external documentation, we did find pull requests that contained links to external websites, however none of them were publicly visible and therefore could not be further investigated. In order to see for what reason these TF file were changed, one needs to look at the source code of the changes.

Section 3.2 also states that only 27% of the TF modifying commits contain a keyword, therefore almost 73% of TF changing commits in labeled pull requests have not yet been investigated. Since these commits are part of a labeled pull request, the chance of them being relevant is greater than a random commit that modifies a TF file. Therefore, investigating these commits can lead to more insights of how relevant commits are described, when they do not contain a keyword.

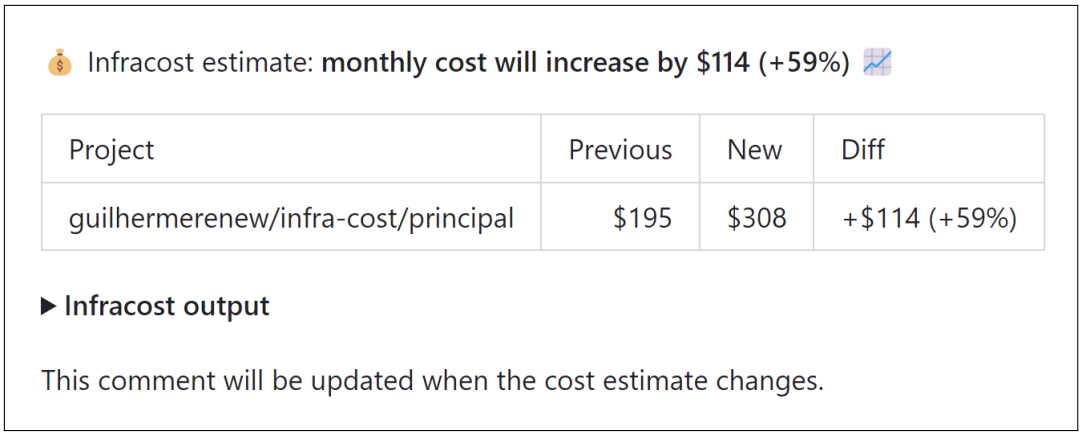

Figure 4.1: Example of an Infracost message in a pull request.

4.2 Keyword locations

Next off, Table 3.1 shows that 50% of the pull requests with keywords have a keyword in the comments. When we look at pull requests with keywords in a single location they are also most commonly found in the comments. However, only 13% of these are found to be relevant. This is most likely because of bots that automatically respond to pull requests. An example of such a bot is 'Infracost', which appears in 29 (3.2%) of the TF pull requests with a keyword. This bot gives an estimate of monthly hosting costs before and after the change, and calculates the difference between them. An example of which can be seen in Figure 4.1. These bots logically contain cost related keywords and the pull requests are therefore flagged. However, as mentioned before, we are only interested in human written text. The addition of such a bot shows awareness. However, these messages by themselves do not, as we are looking for developer awareness.

The least common location for keywords is the pull request title. These titles are quite short and frequently do not mention why a change is made. Often the title and the description are very comparable, which explains why so little keywords are exclusively found in the title. As can be seen in Figure 3.2, the description is left empty in 31 cases, in which case the title often functions as a short summary of the changes. This is comparable to that of a commit message. Whenever the description is the only location that contains a keyword, almost 28% of the pull requests are relevant. This is the highest percentage out of all the exclusive locations. The lowest percentage of relevant pull request out of the exclusive locations is for commit messages. This can be explained by the fact that these messages are not included while labeling the data. Interestingly, 7 pull requests are still given a label, even though they did not contain any keywords. This shows that pull requests can demonstrate awareness, even if they do not contain a keyword.

When we look at Table 3.3, we see that most often a pull request is created with cost awareness in mind. In approximately 23% of the pull requests, a

reviewer mentions cost awareness in a pull request that was not intended for this purpose. This clearly does show that reviewing helps to reduce costs. Furthermore, as can be seen in Figure 3.2, many pull requests do not have any review comments. This number is even higher if we filter out any bot comments from pull requests. This means that the percentage could be higher than 37%, depending on the amount of pull requests with only bot comments.

4.3 Commit and pull request linking

In Figure 3.5 we can see the frequency of labels that are common between labeled pull requests and their labeled commits. For most labels, a clear overlap is visible, for example for the 'saving' label and 'instance' label. For labels like 'cluster' and 'provider' no overlap is recorded. The labels 'provider', 'feature' and 'awareness' are often only found in commits. The case for 'awareness' can be explained by the fact that pull requests can only get either 'awareness' or 'saving', while the commits in the pull request can have both. When the pull request only has 'saving', it means that 'awareness' is commit only. However, this rule does not apply for the 'saving' label to the same extent, as it is mostly a common label. This could be because these pull requests are more elaborate and therefore are more easily identified as 'saving', instead of the more general label 'awareness'. Another thing that can be seen in this figure, is the fact that the more specific labels are more often pull request only. A good example are the labels 'storage', 'policy' and 'cluster'. This could be explained by the fact that pull requests are documented in more detail compared to commits. Therefore these labels appear more often.

5 | Discussion

In this chapter we will go over all of the research questions and give answers with an explanation. Afterwards, we will go over how future work could look like and why it is worth doing beyond addressing the limitations of this work.

5.1 Cost awareness in pull requests

RQ1: Do pull requests show cost awareness, and to what extent?

From Figure 3.2 we can see that there are 249 pull requests created that in some way show awareness. All of these show clear awareness. Most of them in the form of a cost reduction. Therefore, we can conclude that pull requests can show cost awareness.

Not all repositories make use of pull requests. From the 610 repositories that make use of pull requests, only 100 repositories have at least 1 pull request that shows cost awareness. From our findings, it seems that for most repositories, pull requests are not an efficient method to identify cost awareness.

5.2 Cost awareness in code review

RQ2: Do pull request review comments discuss costs, and to what extent?

In Table 3.3 we can see that most often the awareness originates with the author. In about 23% of the time, it originates in the comments because of a reviewer. The percentage could be higher when only factoring in pull requests that have been reviewed. Furthermore, many pull requests exist without any human written comments. Thus, review comments in pull requests are responsible for cost awareness in at least 23% of the reviewed pull requests.

5.3 Finding new commits

RQ3: Do pull requests lead to commits that would not have been caught using the methods in the previous study, and how often?

Section 3.2 shows that only 27% of the TF commits that are a part of a labeled pull request contain a keyword. That means that 73% of these commits do not have a keyword in their message, but have a higher likelihood of showing cost awareness. Using pull requests as a mean to efficiently find previously hidden commits shows to be quite effective in order to find more commits that can show cost awareness.

5.4 Linked dataset

RQ4: Does the linked data alter or strengthen the conclusion of cost awareness, compared to the previous study?

Figure 3.5 shows that commits and pull requests do not receive the same labels in the same way. Having labels for both the commits and pull requests therefore increases the amount of labels for a certain change and thus strengthens the evidence of cost awareness.

5.5 Limitations & Future research

Firstly, an assumption that was made in step 4 of Section 2.4 might not hold in practice. In this step a pull request is rejected if all TF modifying commits are reviewed but not found to be relevant. In theory it is possible that the author did not put enough detail in the commit message, knowing that the pull request would contain all the required details. This means that it is possible for a relevant pull request to be rejected in this step. It would be interesting to see whether developers omit such details from commit messages, but not from pull request descriptions.

As described before, automated bot messages are a frequent occurrence in pull requests. The same is true for (example) code that has been included in a message. All of this text is a hindrance in the research and should be relatively straight forward to filter out. Removing this will result in less false positives, which would greatly speed up the manual labeling process. It would also be interesting to see how the removal of bot review comments has an effect on the percentage of cost awareness in review comments. Especially when comparing to other pull requests that have other developers reviewing the pull request.

Moreover, as described in Section 4.2, pull requests can show cost awareness even if they do not contain a keyword. This shows that the current list of

keywords is not necessary complete. However, adding to this list might also increase the frequency of false positives. A further study about deciding appropriate keywords could give insights how different keywords lead to more pull requests, comparing the amount of resulting relevant pull requests to the amount of false positives.

Furthermore, the current set of labels and their definitions should be improved upon. Many definitions leave room for interpretation and not all labels are specific enough. It should be intuitive and clear for a new researcher to label a dataset using the set of labels. Rules should be put in place on how to label and when to award each label. Currently the policy and feature label especially have a vague definition that can be interpreted in multiple ways. A policy can be to not allow developers to debug on expensive cloud instances, but this is not explicitly an implemented rule. Should it then receive the label? Feature is defined by 3 examples of features, but in practice means anything that is not a specific label, which is not evident from the definition. Furthermore, the alert label should not be awarded when a user receives access to certain cost reports. But one might assume this by looking at the definition. An example of a rule that can be put into place is that there should always be exactly one effect and action type label. As multiple researchers will be labeling different datasets, it is important that there should be the least possible room for a researchers own interpretation. This is to improve the accuracy of the labels awarded.

Next off, how does the selection of repositories have an effect on the outcome. For this research only older repositories are used that were created in or before 2022. Many of these might be abandoned repositories that were merely created for a study project. When including new repositories and only select repositories that are actively being worked by multiple developers and are actually being hosted in the cloud, it would most likely give better insights into many of the research questions.

Last off, it would be interesting to see what the unlabeled commits show in relevant pull requests. Finding out the reason why these commits were not found before might give us more insight in cost awareness for developers. Perhaps the messages were generic and do not show cost awareness or maybe cost awareness is described with words other than the used keywords. Furthermore, it might also be worth to study the code changes themselves. Perhaps, changes are made that show cost awareness, but are not described as such.

6 | Conclusion

Using MSR to retrieve pull requests which modify Terraform files and make use of cost related keywords, a total of 899 pull requests in 610 unique open source repositories were found. All of these are manually reviewed and received labels where appropriate. This results in a total of 249 pull requests in 100 unique repositories that show cost awareness. From this we conclude that pull requests can show cost awareness, but pull requests by themselves are not effective at showing cost awareness for most repositories. When pull requests are linked to their commits, they can strengthen the reason of cost awareness, compared to commits. This because pull requests are often more detailed and therefore more precisely explain why changes are being made.

In 23% of these pull requests, the pull request shows cost awareness because of a reviewer. This shows that code reviewing can be effective at reducing costs. Linking commits to relevant pull requests, shows that 73% of these commits do not contain a keyword in the commit message, even though these commits have an increased likelihood to be created out of cost awareness. This shows that many commits that could show cost awareness do not necessarily make use of the keywords used. Pull requests can function as an efficient intermediate step to retrieve these types of commits.

References

[1] Marcus Vinicius Bertoncello et al. "Pull Requests or Commits? Which Method Should We Use to Study Contributors' Behavior?" In: *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2020, pp. 592–601. DOI: 10.1109/SANER48275.2020.9054855.

[2] Leonardo Rebouças de Carvalho and Aleteia Patricia Favacho de Araujo. "Performance Comparison of Terraform and Cloudify as Multicloud Orchestrators". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. 2020, pp. 380–389. DOI: 10.1109/CCGrid49817.2020.00-55.

[3] Dirox. *The Evolution of Web Hosting*. 2023. URL: https://dirox.com/post/the-evolution-of-web-hosting.

[4] D. Feitosa et al. Dataset and population information regarding hosting costs. 2023. URL: https://github.com/feitosa-daniel/cloud-cost-awareness.

[5] Daniel Feitosa et al. "Mining for cost awareness in the infrastructure as code artifacts of cloud-based applications: An exploratory study". In: *Journal of Systems and Software* 215 (Sept. 2024), p. 112112. ISSN: 0164-1212. DOI: 10.1016/j.jss.2024.112112. URL: http://dx.doi.org/10.1016/j.jss.2024.112112.

[6] IBM. *What is IaC?* Accessed in 2024. URL: https://www.ibm.com/topics/infrastructure-as-code.

[7] Oleksii Kononenko et al. "Studying Pull Request Merges: A Case Study of Shopify's Active Merchant". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. 2018, pp. 124–133.

[8] Sujatha R. *What is pay-as-you-go Cloud Computing (PAYG)?* 2024. URL: https://www.digitalocean.com/resources/article/pay-as-you-go-cloud-computing.

A | Additional Data

Label	Definition
alert	text expressing concerns related to billing alarms enforcing an upper threshold on costs
area	text expressing concerns related to server or instance geographical location.
awareness	text simply mentioning taking cost into account. (without necessarily implying action).
billing_mode	text expressing concerns related to the type of billing plan being used (e.g., on-demand for development or normal plan for production).
change	text expressing a (permanent) change in production code
cluster	text expressing concerns related to cluster configuration.
cpu	text expressing concerns related to cpu configuration.
domain	text expressing concerns related to domain name system and IP addresses
feature	text expressing concerns related to various features such as logging, load balancers or usage of third party libaries.
increase	text expressing concerns related to increase in cost due to a change.
instance	text expressing concerns related to computational instances (e.g., Amazon AWS t2.micro) used in the deployment.
nat	text expressing concerns related to nat configuration
networking	text expressing concerns related to networking configuration
policy	text expressing concerns related to the implementation of general rules to prevent charges.
provider	text expressing concerns related to choosing a service providers (e.g., Amazon, Azure, Google).
saving	denotes mentioned changes made to save costs.
storage	text expressing concerns related to storage solutions (e.g., Amazon gp2) used in the deployment.
ram	text expressing concerns related to ram configuration.
test	text expressing a temporary change in production code for testing purposes
vpn	text expressing concerns related to vpn configuration

Table A.1: Label definitions, taken in the context of cost awareness.