Chapter 8
The CloudScale Method for Managers

Steffen Becker, Gunnar Brataas, Mariano Cecowski, Darko Huljenić, Sebastian Lehrig, and Ivana Stupar

Abstract Having described the CloudScale method for engineering scalable cloud computing applications in the previous chapters, we explicitly address managers of software development processes in this chapter. It answers questions managers have in mind when considering the CloudScale method: Is it worth implementing the CloudScale method in my organization? What does it take? What are the benefits? What will be the costs? How should I get started? This chapter addresses all these questions and provides answers based on our own experience that we gained when introducing and applying the CloudScale method in practice. In the course of the chapter, we distinguish two types of managers: project managers, who are concerned with managing project teams that implement the business requirements, and technical managers, who manage the actual development efforts and take technical decisions.

The chapter is structured as follows. After a brief introduction (cf. Sect. 8.1), it first addresses project managers. We illustrate key considerations that project managers should be making when applying the CloudScale method (cf. Sect. 8.2). Afterward, we sketch how the CloudScale method interacts with other development processes (cf. Sect. 8.3) and what its pros and cons are (cf. Sect. 8.4). The remainder of the chapter addresses technical managers. First, it sketches a pilot project

S. Becker (✉)
University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
e-mail: steffen.becker@informatik.uni-stuttgart.de

G. Brataas
SINTEF Digital, Strindvegen 4, 7034 Trondheim, Norway
e-mail: gunnar.brataas@sintef.no

M. Cecowski
XLAB d.o.o., Pot za Brdom 100, 1000 Ljubljana, Slovenia
e-mail: mariano.cecowski@xlab.si

D. Huljenić • I. Stupar
Ericsson Nikola Tesla, Krapinska 45, 10000 Zagreb, Croatia
e-mail: darko.huljenic@ericsson.com; ivana.stupar@ericsson.com

S. Lehrig
IBM Research, Technology Campus, Damastown Industrial Estate, Dublin 15, Ireland
e-mail: sebastian.lehrig@ibm.com

S. Becker et al. (eds.), *Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications*, DOI 10.1007/978-3-319-54286-7_8

in Sect. 8.5 to guide the discussion. Using this pilot, in Sect. 8.6, we briefly outline how to set up CloudScale's IDE, which can be complemented by third-party tools introduced in Sect. 8.7. We apply the CloudScale method on the pilot in Sect. 8.8.

8.1 Introduction

After introducing the CloudScale method and its tools for software architects and developers, you as project and technical managers might still wonder how the entire process of following the CloudScale method looks like in practice. Despite that the CloudScale method steps were already described in more detail in previous chapters, in particular in Chaps. 5–7, this chapter gives an overview of the entire CloudScale method targeting management aspects. First, aspects related to project management are discussed, followed by technical management aspects. For the latter, we introduce in Sect. 8.5 a simple application used as a pilot.

The goal of this chapter is to guide interested project managers with respect to the cost and effort required to perform the steps of the method. You will be informed about some of the key considerations in the CloudScale method in order to gain most benefit out of it. We will discuss the pros and cons of the CloudScale method based on our own experience. The relation of the CloudScale method to other engineering practices is discussed and possible integrations are introduced. Afterward, we guide technical managers through the process of using the CloudScale method for an analysis and optimization of a simple pilot project. This includes the setup of the CloudScale Environment, as well as the choice of the CloudScale tools and complementing tools needed to complete the CloudScale method iteration. At the end of the chapter, we briefly outline future prospects of the CloudScale method usage.

8.2 Key Considerations

When project managers consider using the CloudScale method, they should answer a set of questions for their specific organization and project context (which might be elaborated together with the technical manager). We call these questions the *key considerations* to make before implementing the CloudScale method.

Is it worth it at all? The first question that project managers should address is whether they want to invest the money and effort required in implementing the CloudScale method at all. The answer to that question depends, to a large extent, on the risk of failing to meet scalability, elasticity, or cost-efficiency requirements. The more business or mission critical these requirements are, the greater is the benefit you get out of the CloudScale method. For example, when you implement an online flower shop, it is important to scale to the customer load faced on Mother's Day as well as using only few resources during the remainder of the year.

What is my use case? Once you have decided to use the CloudScale method, the next thing to consider is the use case in which you want to use the CloudScale method. In case you have an existing development ongoing, you most likely want to verify it to identify potential issues. This often happens in the context of migrating an existing system to the cloud. In this case, you have the source code of this system. The next thing to consider is the language and maturity of the code: Is it written in Java? Does it follow a clean modularization in terms of classes or components or is the code a "big ball of mud"—grown without governance over time? In the former case, you get more support by the existing CloudScale tools; in the latter case, you should consider asking the technical manager to refactor the source code in general before addressing its scalability or before migrating it to new platforms.

How complex is my system? What analysis granularity do I need? When tool usage on the source code is possible, the next thing to consider is the system's complexity. In case it is a huge system, this has two consequences. First, the system's implementation might be too large to be consumed by the CloudScale tools. In this case, you should plan for efforts involved in adjusting the CloudScale tools to your system's complexity. Second, you should carefully consider all configuration options which have an impact on the granularity of CloudScale's analyses (cf. Sect. 5.2). In the case of huge systems, you should definitely start with a coarse-grained analysis and refine it later on.

Which analysis questions do I need to have answered? Which requirements do I have? Next, you should also consider your particular problem or question. First, you should know the quality attributes which are crucial for your mission's success. This also means, if you have not gathered scalability, elasticity, or cost-efficiency requirements and quantified them, now is a good time to do so. This will help you in your future development even when not using any of CloudScale's methods or tools. When collecting such requirements, you should also additionally collect the critical use cases and key scenarios. From our experience, problems with scalability of systems are often rooted in ill-specified or non-existing scalability requirements!

Do I already know about scalability issues? When the requirements are clarified, you should consider where in your system you may have weak spots which might prevent requirements' fulfillment. Typically, the development team of a certain software system has a good idea about the quality properties of the system. Go ahead and interview them to learn about the most critical components in your system. If you are lucky, you also learn about concrete problems with your system as it is at that time. You should use this information to focus on particular parts of your software and analyze them right in the beginning.

Which metric do I want to have analyzed? Finally, you should be clear about the type of information you would like to gather during the analysis of your system; i.e., you should have concrete questions and metrics you want to know about the system. Having a proper understanding of this helps again to focus the analyses and also ensures that the answers you get are the answers you wanted.

How do I map my system and context to the CloudScale method? Before you start analyzing your system, you should revisit the CloudScale method and check whether you fully understood all method steps needed. In particular, you have to be

able to map your system and context to the CloudScale method steps. All required inputs (documentation, code, etc.) should be available for all steps. If not, you have to gather them first.

When modeling, do I have enough skilled people? In case you want to implement a new function or large parts of your system from scratch, using a model to plan and verify your design is more useful. You should consider the maturity of your developers and of the development process before doing so. In particular, you need to ensure that you have the needed skills available in your development team. We recommend also to start modeling on a small project or subsystem before moving to large systems.

After running the method, how do I reflect on its success? All the above key considerations are pure theory as long as you have not tried to execute the CloudScale method at least once. When you have decided to do so, you should also define an evaluation plan to judge the method application in the end: Did it provide the benefits you expected from it? Were all the tools scalable for your use case? Did the tools provide enough usability? Did you provide good-enough third-party tools? What could you improve on the next iteration of the method? Did you properly identify the context of the method application or do you need to readjust this for the next iteration? In addition, you should track the time needed for each of the method steps and compare it to your expectations. In case of misalignments, identify the causes and try to improve it.

To summarize, there are several questions to consider and answer by taking your context into account. It helps to have an idea about the CloudScale method (as provided by the pilot project; cf. Sect. 8.5). It also helps to interview your technical manager or developers about the current status of the system and the most important known issues. Also, you should have clear plans where you want to go with the system, i.e., when migrating it from legacy to cloud platforms. In particular, you should know the resulting requirements based on the new or extended business models behind the newly developed system features or quality properties.

8.3 Relation to Other Engineering Methods

The CloudScale method will almost, in any case, not be the sole engineering method needed for the development of your cloud application. The CloudScale method will rather be complementing your normal development processes and methods. In that way, the CloudScale method complements classical development processes like SCRUM or the rational unified process (RUP). When looking at such processes, the CloudScale method complements several traditional steps with enhanced or additional activities and tools.

When looking at the commonalities of all engineering processes, we can identify the following *core ingredients* to engineer cloud systems:

Foundations This covers the fundamental basics, underlying concepts, guiding principles, and taxonomies of the used cloud concepts, terms, and technologies.

Implementation This covers the building blocks and practices used to realize cloud applications.

Lifecycle This aspect covers the lifecycle model of the cloud application. It typically describes end-to-end iterations of a particular cloud development and operation.

Management This aspect addresses the management questions arising when realizing a cloud application. It should cover both design time and runtime cloud management. Each management aspect should be tackled from multiple perspectives.

Our CloudScale method and its tools partly cover all these cloud engineering elements. This method contributes to the foundations of cloud concepts, terms, and technologies with scalability guidelines and a library of appropriate Architectural Templates (ATs). It furthermore provides definitions of important concepts like scalability, elasticity, and cost-efficiency. In the implementation phase, the CloudScale method provides a plethora of analyses and guiding principles that software architects can apply to systematically address scalability, elasticity, or cost-efficiency issues. CloudScale's support is more limited toward the lifecycle ingredient. However, it can cover elements related to design via models, or to quality assurance and evolution support via its Spotters. For the management ingredient in particular, we have included lots of experiences, key considerations, a pilot project, etc. in this chapter.

However, other methods and tools also cover aspects related to cloud systems' realization. For many domains, there are standard supporting engineering principles. In the cloud environment, is important to emphasize that used methods and tools have to enable the process of designing the system in a way that it leverages the power and economics of cloud resources to solve a (scalable) business problem. As cloud applications is still a growing market, it is taking a serious commercial track and there are a lot of supporting tools and ready-to-use documented HowTos, for example, specialized cloud patterns. The main focus of current offerings on the market is tools for price/cost calculation of the required IaaS resources in a particular cloud environment like Amazon EC2. Often, these offerings are much simpler than the CloudScale method, but also much more inaccurate. In addition, they are often in-transparent to the customer; i.e., the customer cannot verify whether the approach computes correct costs and whether it suggests the best IaaS provider (in contrast to suggesting IaaS providers based on hidden contractual relationships).

To summarize, there are two different relations of the CloudScale method to other engineering methods. First, it has to be integrated and mixed with classical software engineering methods like SCRUM or RUP. This integration will be organization and project specific. The CloudScale method provides method steps and tools which have been tailored to be reusable building blocks, which can be put on top of other engineering methods. Second, there are alternative or competing cloud engineering methods and tools. In particular, there are several commercial offers today on the

market. For these, you have to judge the extent to which they can complement the CloudScale method or they are in conflict with the CloudScale method. You also need to identify their real contributions in one of the above listed core ingredients. In addition, you need to judge whether they provide open and objective or biased information.

8.4 Pros and Cons of the CloudScale Method

Introducing any new step in the organizational system development process always poses a question of the benefit that the organization can gain from the newly added step (novel part of the process). There is no method that can provide gains exclusively, without any expense, because there is always effort and cost that must be invested into achieving advantages and potential profit. This is the main reason why in this chapter, we analyze the parameters that are important for making the decision on how useful the employment of the CloudScale method and its tools is. These analyses are the foundation for identifying organizational pros and cons of using CloudScale method and tools. The organization that wants to implement the CloudScale method should be prepared to actually calculate the benefits of the needed investment for the additional quality provided, in contrast to potential losses due to potential system problems and costs to maintain or reengineer bad system components in operation, or, even worse, with system failure or delays during operation.

8.4.1 Critical Success Factors for Method Adoption and Use

Successful adoption and use of the CloudScale method and tools relies on several conditions in terms of requirements.

- Having clearly instantiated requirements in order to identify particular system scalability, elasticity, and cost-efficiency needs, as well as having established system behavior scalability conditions.
- Having use cases related to system usage, which enables extracting usage patterns and correlating them with scalability requirements (primarily system work and load expected through the system lifecycle, with a focus on the behavior in the limits).
- Freedom to change currently implemented architectural patterns. Some organizations use general architectural patterns, or patterns adapted in a custom way, which in most cases need serious reengineering in order to fulfill the new system requirements. This usually requires thoughtful architectural discussions and decisions.
- An organizational development process: Many organizations use a plethora of available processes and methods, and as such, they are usually less limited and

controlled in adopting new ones. In contrast, there are big development organizations that follow standardized general-purpose or adapted/tailored development processes in a much stricter way. In such cases, it is usually required to elaborate benefits for introducing some new process or changing the steps in their own established development process environment.

- Team experience in dealing with system performance issues: This is usually also related to the selected development method and the possibility to solve overall system behavior issues or to focus on a particular system problem. Another question that needs to be considered is how much freedom does the development team have to experiment and prototype the system before they start to produce a solution.
- If the solution for the new system, or a part of it, is produced by reusing an existing code base, a good understanding of the source code that has been reused in the new system is of great importance. Similarly, a solid knowledge about the system is required when a solution for the problems with a system in operation needs to be found.

  In case of applying the CloudScale method and its tools on a system that reuses an existing code base, it is very important to have the reused source code available for analysis. If the system is built from scratch, it is crucial that your organization is willing to follow the concept of model-based development. Also, an appropriate person that can create a model of the analyzed system has to be assigned to achieve an adequate accuracy in the prediction of the potential system behavior.
- Documentation of the newly introduced method and tools: Depending on the extent of the existing documentation, users should be prepared to invest certain effort into researching and investigating new tools and their features, since they are likely not documented to the same level as mature tools.
- The tool's ability to deal with a large existing code base.
- Potential of the tools to work with so-called *grey-box* components and *grey-box* models, meaning that the capability of creating models and conducting analysis does not depend on having the entire code available; instead, it should be possible to just describe the component behavior. This can be very important for systems which access external services without available source code.
- Usage of existing frameworks as a base for developing or encapsulating new services: In this case, an architect is not interested in the analysis of existing frameworks, but only in analyzing the interaction with a custom-developed component.

If the provided method and its tools can fulfill the requirements for their implementation in the development process of an organization, it is possible to predict successful usage and user's satisfaction. In the case of the CloudScale method and its tools, fulfillment of stated requirements is the foundation for their application on the system under analysis.

8.4.2 Organizational Issues

The development of new systems and services is a team effort in any organization. The development process starts with the collection of requirements and defining the expected development scope. The product manager, together with the customer (e.g., in cloud-based system, it is most likely a service provider), formulates both functional and extra-functional requirements and determine the main focus of the cloud system's scalability requirements. Based on the collected requirements and the expected system behavior, the system architect makes early decisions on the overall system architecture. This is the foundation for the organizational decision which parts of the expected final architecture can be reused from previously developed systems, and what new services need to be developed. This is where the first challenges may arise because the combination of certain existing components/services can look promising, and the decision needs to be made quickly about the way the system will be realized. In such situations, it is crucial to determine whether the proposed realization of the system is feasible, or at all possible, and how it will impact the expected system behavior. Also, it is important to consider what can happen to the scalability requirements when composing services according to the selected architecture. If an organization skips this step, which aims at identifying the appropriate solution and making the right decisions about the system early on, and immediately jumps to service implementation, it can lead to costly issues found during the testing phase, or even worse, during the system's operational phase. Managing scalability requirements from the beginning of the system development requires full understanding of the scalability concepts and usage of all needed tools and methods that can support architects and developers to make fast selections of good solutions. In terms of organizational preconditions, achieving such a way of working can sometimes be challenging due to the complexity of the decision-making process in large companies. This is due to heterogeneous teams participating in the development process using the method and tools in different system lifecycle phases.

8.4.3 Costs

Using almost any kind of method, and especially introducing a new one in the development process, results in certain costs due to the time and effort invested in performing specific method steps. The costs involved when using the CloudScale method can be structured as follows:

New process steps The CloudScale method introduces process steps that will be new compared to the current methods employed in the organization's development process. Performing each of those steps has costs in terms of human effort.

Employee training Training of the involved personnel requires teaching them about the conceptual aspects of the CloudScale method (e.g., cloud computing

paradigm, SLOs, scalability concepts, etc.), as well as training them in the use of the CloudScale method and its tools. From our experience, the entrance barrier is somewhat lower for CloudScale's migration and evolution support, as it is more closely oriented toward analyzing the system's implementation. Training personnel in model-based analysis typically requires certain time, since most developers and software architects working in practice are unfamiliar with such approaches. Hence, they need to learn and understand modeling techniques and languages, as well as identifying the right model abstractions. In addition, CloudScale's tools require some familiarity with stochastic theory, which is a knowledge that in practice often needs to be refreshed, especially in an industrial environment. We estimate that training these skills on a basic level requires a 2- or 3-day workshop.

New organizational processes The organization itself must also be adapted to the CloudScale method. New ways of collaboration may be required due to the different roles involved in conducting the CloudScale method.

8.4.4 Covering the Cost of the CloudScale Method Adoption

As mentioned in the previous section, the CloudScale method will inevitably produce a certain cost, and this expense should be offset by savings in the overall system lifecycle costs. In that way, adoption of the CloudScale method will become profitable, and is able to benefit the organization on the long run. We identify four ways of reducing the overall lifecycle costs when using the CloudScale method:

Better quality When a service suffers from insufficient scalability, their users become dissatisfied and leave. Ultimately, in addition to losing customers (and in severe cases the organization's reputation), additional money may be lost by investing in the system's maintenance and paying for SLO violation penalties. A controlled system design from the beginning of the product lifecycle will result in better system quality, thus reducing the potential cost of complex and expensive problem-solving activities in the operational system phase.

Less redesign It takes an additional engineering effort to fix a service with poor scalability. Other than just introducing the cost of providing the problem solution, your engineering team will not be able to build new functionality due to their involvement in fixing the problem. Additionally, redesigning a service takes time, during which the original services will suffer from an inadequate scalability.

Less gold plating A development team which is eager to avoid scalability problems may use too much engineering effort when designing parts of a service. As a result, development costs will be higher, but since a scalable service may sometimes also be more complex, maintenance costs may also increase. A better overall view of the system's scalability requirements may reduce gold plating, so engineering efforts are used only where required.

Lower operating costs Services with poor efficiency will require a considerable amount of costly cloud computing resources. A system with poor elasticity will also be challenging to operate and will therefore require more manual tuning, thus requiring more engineering man-hours.

Having both costs and benefits in mind, development organizations should perform serious analysis before they decide to start using the CloudScale method and its tools so that they can understand the potential offset of their investment by savings in the overall system's lifecycle expenses.

8.4.5 Risks

We have identified the following risks related to using the CloudScale method:

Too high effort The effort involved in using the CloudScale method may simply be higher than what is possible to argue in an organization. This risk can be reduced if the CloudScale method is used on smaller projects first so that the organization gets a better grip of the actual costs involved in using the CloudScale method. As the organization gets more familiar with the CloudScale method and its tools, the size of the projects may gradually be increased.

Lack of accuracy It is very important to have an overall idea of the required level of accuracy when using the CloudScale method. To a large extent, the accuracy depends on the quality of the parameters used.

Based on our experience in applying the CloudScale method and its tools on differently sized systems and in different organizational environments (small, medium, and large companies), with careful selection of the project and a well-balanced granularity of the system, the mentioned risks can be mitigated and properly addressed.

8.4.6 Critical Factors for Successful Projects

Whenever an organization tries to apply a new or improved method in its development process, it is important to understand the potential implementation environment. The three essential aspects of the targeted environment are: the development team that will produce the first pilot, the system selected for the experimentation/piloting, and the time available for learning new concepts and making experimental developments. Based on these presumptions, we can define a list of critical success factors for the project that you should consider when applying the CloudScale method:

- Modeling experience—assign a development team which has experience with abstract concepts and system performance modeling.

- Time for learning the methods and tools—the team must have enough time to learn the concepts implemented by the tools and to be able to map the problem to the adequate tool.
- Compatibility with the current development environment—many development organizations already use standard (or customized) development frameworks that encapsulate tools and methods, and it is very important to have sufficient time to adapt to the new environment and have enough freedom to correlate and include the outputs of the new method in their existing environment.
- Availability of the source code for the system analysis—especially in the case of system composition, it is important to have the system's source code as well as any documented problems with the currently used frameworks.
- Organization of the existing code—currently used frameworks and applied development conventions should reflect the current state of the art.

8.5 A Pilot Project

After highlighting the considerations to be made by the project manager, in this section, we will introduce a very simple example system which can be used by technical managers or senior developers to get started with the CloudScale method. Its purpose is to help them to get an impression of what the CloudScale method is, what it provides, and how complex it is to get started, and to assess whether it is worth to consider using it in their own development projects. This *pilot* project is also part of the CloudScale integrated development environment (IDE), as an example ready to be used out of the box so that it can be understood and inspected in all aspects.

The pilot system is a simple client/server application implemented in Java. The server generates HTTP responses under pre-defined URLs (in a REST-like style) and the client issues corresponding HTTP requests. The client can be scaled up so that it generates a significant amount of load on the server. The two main operations provided by the server are (a) a computation of a dynamic number of Fibonacci numbers, and (b) a simple query operation, which, however, contains a synchronized block, i.e., can be executed only mutually exclusive. Figure 8.1 shows an excerpt of the code for the query operation where the `call` method of the `OLB` singleton contains the synchronized block.

```
1  @GET
2  @Path("testOLB")
3  @Produces(MediaType.APPLICATION_JSON)
4  public String testOLB() {
5          OLB.getInstance().call();
6          return "Hello␣from␣OLB␣Test␣Method!";
7  }
```

Fig. 8.1 Source code of the server-side example query method in the pilot project

```
1  public final class OLB {
2  [..]
3      /**
4       * Method leading to a One Lane Bridge.
5       */
6      public synchronized void call() {
7          try {
8              Thread.sleep(TIME_TO_SLEEP);
9          } catch (final InterruptedException e) {}
10     }
11 }
```

Fig. 8.2 Source code of the call() method in the OLB singleton

At runtime, the client can execute this method by issuing an HTTP request to the server for the URL `http://<servername>/testOLB`. The idea of the two methods of the pilot project is that one method is rather resource intensive (fibonacci), while the other represents an One-Lane Bridge (OLB) scalability HowNotTo (cf. Sect. 2.10; code shown in Fig. 8.2). Both methods are easy to understand in a few seconds and illustrate both common problems of web services and the capabilities of the CloudScale method and its tool support.

The pilot project can be used to showcase the CloudScale method. It covers both use cases of the CloudScale method. Software developers can easily create a model of the system. This can be done manually using the editors, as the system's complexity is really low, or by using the Extractor tool, as the system is provided as object-oriented Java code (i.e., it fulfills the Extractor's prerequisites). Using the model, software developers can analyze questions like:

- What is the response time of the system under varying load?
- What is the system's capacity?
- How does the system scale up or down during elastic adaptations?
- etc.

On the other hand, the pilot project can also be used in the second CloudScale method use case, the detection of HowNotTos. This can be done statically (again, the necessary prerequisite is that a model can be extracted by the Extractor) or dynamically by executing the pilot project. The project's client and server come with the needed stand-alone web server and client libraries so that they should run as soon as a Java virtual machine (JVM) can be found in the target environment. When executed, the CloudScale's Dynamic Spotter can detect the HowNotTos which have been injected into the example (i.e., the OLB or the excessive CPU consumption HowNotTos). To get things up and running here also with little effort, the Dynamic Spotter provides a custom-tailored load generator for the pilot project which can be used with minimum configuration effort.

Overall, this pilot should give project and technical managers a good idea of the benefits and considerations when wondering whether to try the CloudScale method.

We will also come back to the pilot project when we give a more detailed high-level walk-through through the CloudScale method based on the pilot in order to highlight the management decisions needed when executing the CloudScale method in Sect. 8.8 (in contrast to the technical aspects of the method discussed in the previous chapters).

8.6 Setting Up the CloudScale Environment

When the decision has been made to use or try the CloudScale method, the first thing you want to do is to learn the CloudScale method and to install its tool support in form of the CloudScale Environment. CloudScale's tools are altogether integrated in the so-called **CloudScale Environment**. This integration makes it easy to use all tools from a coherent user interface providing a unified user experience. In addition, the CloudScale Environment supports you when following the CloudScale method: It provides a build-in CloudScale method view, which you can use as a workflow engine to track your progress in the CloudScale method. The CloudScale Environment can freely be downloaded at CloudScale's web page [1].

It is available for all major platforms. It requires a JVM on the developer machine, as it is based on the well-known Eclipse IDE, which is implemented in Java. There is no need to install it; downloading and extracting the provided files should be enough; i.e., it is easy to try it out without polluting your development environment too much.

After starting the CloudScale Environment for the first time, we suggest to get used to it using the provided pilot project. It can be found under `Examples` as the `Minimum Example`. Using the pilot, you can go through all CloudScale tools. They can be accessed using the most common parameters only from the CloudScale perspective. However, in case you need to fine-tune a single tool or access advanced features, the CloudScale Environment also provides dedicated perspectives which provide access to the advanced features and settings of each tool.

Overall, the CloudScale Environment provides the following tools and guides software architects in the order in which they have to be optimally applied:

ScaleDL editors The CloudScale Environment provides several editors to create and modify Scalability Description Language (ScaleDL) models. CloudScale's catalog of ATs is integrated in these editors, allowing software architects to efficiently create even complex models.

Extractor The Extractor is a reverse engineering tool for automatic model extraction. It parses source code and generates partial ScaleDL models that can be further used by the Analyzer and the Spotter.

Analyzer The Analyzer allows to analyze ScaleDL models regarding scalability, elasticity, and cost-efficiency of cloud computing applications at design time. For these capabilities, CloudScale integrated novel metrics for such properties into the Analyzer. Analyses are based on analytical solvers and/or simulations. The

Analyzer particularly supports to analyze self-adaptive systems, e.g., systems that can dynamically scale out and in.

Spotter The Spotter allows to statically and dynamically detect scalability issues. For a static detection, the Spotter automatically detects search patterns on ScaleDL instances created by the Extractor. Found patterns are interpreted as potential scalability anti-patterns. All scalability anti-patterns are defined in a pre-defined but extensible pattern catalog.
For a dynamic detection, the Spotter provides a framework for measurement-based, automatic detection of software performance problems in Java-based enterprise software systems. The framework combines the concepts of software performance anti-patterns with systematic experimentation.

Distributed JMeter [2] Distributed JMeter is a workload generator application. When estimating resource demands, e.g., to parametrize ScaleDL models, such a workload generation is needed. Distributed JMeter can be deployed on Amazon web services (AWS) or OpenStack.

To get started, go through each tool and use it on the provided pilot project!

8.7 Complementing Tools

The engineering of cloud computing applications cannot be completely automated; however, appropriate tools can help software engineers in becoming more efficient. For the CloudScale method, CloudScale's Environment is a good choice because it is tailored for an efficient use within this method.

The CloudScale method is not tied to the official CloudScale tools: other tools can substitute or complement the official ones. This independence has the benefit that there is no vendor lock-in, and that tools that are already in use can be retained.

The following list of tools exemplifies their integration into the CloudScale method. The list is not exhaustive and should only serve for exemplification and initial pointers.

Palladio [3] Palladio is the open-source software architecture simulator that underlies CloudScale's Analyzer. Palladio can be used in stand-alone mode, without depending on the CloudScale Environment.

Apache JMeter [4] Apache JMeter is an open-source application to generate workload and measure performance metrics. It was the basis for CloudScale's Distributed JMeter, but can also be used in stand-alone mode.

Kieker [5] Kieker is an open-source performance monitoring framework. Kieker allows to instrument source code with probes which gather response time measurements at runtime. Response time measurements are especially important for estimating resource demands, e.g., to parametrize ScaleDL models. CloudScale does not provide a dedicated tool for this purpose; CloudScale's evaluations also used Kieker.

Dynatrace Application Monitoring [6] Dynatrace Application Monitoring is a commercial alternative to Kieker with advanced features that ease performance

problem detections. The tool therefore may replace dynamic detection parts of CloudScale's Spotter.

JProfiler [7] JProfiler is a commercial application to profile Java applications. Profiling helps in understanding performance characteristics of applications, which finally allows to identify critical use cases.

Java VisualVM [8] Java VisualVM is a freely available profiling tool for Java applications. The tool comes together with Oracle's JVM and therefore provides a good alternative to JProfiler.

R [9] R is a freely available programming language and environment for statistical assessment of data. With R, the data from performance monitoring tools can be appropriately prepared to be used as resource demand estimates.

8.8 Following the CloudScale Method for the Pilot Project

This section will provide a walk-through through the CloudScale method based on the provided pilot project. As such, it brings together the lessons learnt from the previous sections on the pilot and the tools, and shows key considerations and management decisions which are useful for the small example system.

When starting with the CloudScale method, the first step is to elicit SLOs and derive critical use cases and key scenarios from them (cf. Sect. 5.5). For the pilot project, we assume in the following that it has a response time SLO which says that both provided operations should react in acceptable times for Internet users. This can be considered to be 2 s. Therefore, we define SLO violations to happen if the mean response times over 10 s intervals exceeds those of 2 s. As critical use case, we consider a mix of equally calling both the OLB and the Fibonacci service. A key scenario is one where 100 concurrent users access the system hosted on a single machine for this. The latter can also serve as technical requirements for scalability in this simplified example.

In the CloudScale method, we have to consider then whether we want to model the system or whether we want to base analyses on code. As we have no real idea of the properties of the pilot project, we consider using the code as the main basis for the first method iteration. An additional aspect which suggests using code-based analysis is that the code is provided in Java and is well-structured.

After making this decision, we have to run the Spotter on the pilot project. In a first step, the Static Spotter extracts a model from the code using the Extractor and, when configured correctly (i.e., using the right granularity, which is a low one for this simple project), spots the OLB method already and reports it as a potential scalability HowNotTo. In the second step, we run the pilot project using the provided built-in server and use the Dynamic Spotter on it, together with the provided load generator. We set the number of concurrent users to 100 and the SLO violation threshold to 2 s. After running the Dynamic Spotter, we analyze the results and find that the OLB method—due to its limited concurrency—violates our technical requirements.

Hence, we can rework the pilot project. For example, we could assume that the synchronized keyword is a mistake made when implementing the method. As a simple fix, we can remove it and then rerun the analysis. Now, everything should be fine, as one server is usually strong enough to deal with 100 users.

This concludes the CloudScale method for the pilot project. However, we could for example go ahead and add another service to the pilot and use a model to analyze this evolved system. We would extend the critical use case and the key scenario to cover also the new method. Using the Extractor and manual model annotations would yield a model to be analyzed. Using this model, we could then check for example what the system's capacity is for the 2 s response time SLO.

8.9 Conclusion

In this chapter we discussed the CloudScale method from the perspective of managers considering to introduce the CloudScale method into their development processes. Introducing a new method in the current development process established within an organization will inevitably introduce expenses as well. For this reason, the decision on implementing a new method has to be based on a carefully performed analysis of potential benefits and novel costs implied by the integration of the novel process steps, which require effort and time. This chapter presents a practical point of view regarding the adoption and usage of the CloudScale method and its tools. We identify potential issues, costs, and risks of the CloudScale method adoption, as well as how those risks can be mitigated, and what are the factors determining the success of both the introduction of the new method and the project it will be applied on. One of the main challenges related to the CloudScale method is the time it takes to get familiar with the concepts it brings to its potential users (i.e., system performance modeling, etc.) and the readiness of the organization to invest in the proper integration of such methods in their currently established processes and environments. The whole process of following the CloudScale method is explained on a simple pilot so that the reader can promptly get a grip on the method steps. We consider and present different aspects of the CloudScale method adoption from the perspective of development teams in organizations that are already using various development processes and frameworks.

Despite all considerations, the CloudScale method provides major benefits to the development of scalable, elastic, and cost-efficient cloud computing applications. It offers benefits to software architects to make the right engineering decisions, but also to managers as to how to avoid risks of implementing systems which will fail in the end to meet their scalability, elasticity, or cost-efficiency requirements.

The overall objective for further work on the CloudScale method is to introduce improvements, such as simplifying the use of the tools (i.e., users could be presented with fewer choices, and the manual modeling effort could be reduced), and potentially further reducing the need for new engineering knowledge. This also requires more studies of how scalability, elasticity, and cost-efficiency are actually

handled in organizations today and how it could be handled more effectively in the future. With such additional studies, we can also provide more guidance with respect to choosing the right granularity level or planning for the amount of manual effort needed. Additional improvements such as more ATs or scalability anti-patterns can drastically reduce the modeling effort and increase tools' usability and, therefore, increase the return on investment. In this way, the CloudScale method may become a part of standard practice in contemporary software development, especially for projects where scalability, elasticity, and cost-efficiency requirements are major threats to a project's success.

References

1. CloudScale: The CloudScale Environment (2016). http://www.cloudscale-project.eu/results/tools/ [Visited on 12/19/2016]
2. CloudScale: Distributed JMeter (2016). http://github.com/CloudScale-Project/Distributed-Jmeter [Visited on 06/20/2016]
3. Palladio: The software architecture simulator (2016). http://www.palladio-simulator.com [Visited on 06/20/2016]
4. Apache JMeter: (2016). http://jmeter.apache.org [Visited on 06/20/2016]
5. Kieker: (2016) http://kieker-monitoring.net [Visited on 06/20/2016]
6. Dynatrace Application Monitoring: (2016). http://www.dynatrace.com/en/application-monitoring/ [Visited on 06/20/2016]
7. JProfiler: (2016). https://www.ej-technologies.com/products/jprofiler/overview.html [Visited on 06/20/2016]
8. Java VisualVM: Java Virtual Machine Monitoring, Troubleshooting, and Profiling Tool (2016). https://visualvm.github.io/ [Visited on 06/20/2016]
9. R: The R Project for Statistical Computing (2016). https://www.r-project.org [Visited on 06/20/2016]