Capability Management in the Cloud: Model and Environment

Jānis Grabis(✉) and Jānis Kampars

Institute of Information Technology,
Faculty of Computer Science and Information Technology,
Riga Technical University, Kalku street 1, Riga 1658, Latvia
{grabis, janis.kampars}@rtu.lv

Abstract. Capabilities represent key abilities of an enterprise and they encompass knowledge and resources needed to realize these abilities. They are developed and delivered in various modes including in-house and as a service delivery. The as a service delivery mode is provided in the cloud environment. The cloud-based approach allows offering capabilities possessed by the service provider to a large number of potential consumers, supports quick deployment of the capability delivery environment and enables information sharing among the users. The paper describes a cloud-based capability management model, which support multi-tenant and private modes. The architecture and technology of the cloud-based capability development and delivery environment is elaborated. The pattern repository shared among capability users is a key component enabling information sharing. Additionally, this paper also shows usage of the cloud-based capability and delivery environment to build cloud native capability delivery applications.

Keywords: Capability management · Capability as a service · PaaS
Scalability

1 Introduction

Capabilities represent key abilities of an enterprise and they encompass knowledge and resources needed to realize these abilities. These capabilities can be used internally or provided to external companies as a service. Capability as a service implies that the capability bearer delivers abilities and resources to other companies on a contractual basis. For instance, a consulting company has IT management capabilities and these capabilities are delivered to its contractors as well as internally in the company itself.

Providing capability as a service stipulates specific requirements towards capability development and delivery:

- Rapid deployment to onboard new consumers quickly without forcing them to alter existing IT landscape;
- Scalability to support many consumers and to deal with computationally demanding context processing and adaption needs;

R. Matulevičius and R. Dijkman (Eds.): CAiSE 2018 Workshops, LNBIP 316, pp. 40–50, 2018.
https://doi.org/10.1007/978-3-319-92898-2_4

- Collaboration to enable participative capability design and evaluation of capability delivery results;
- Knowledge sharing to benefit from exchange of capability usage experiences by different consumers.

These requirements can be met by using cloud technologies. Capability development and delivery over the cloud combines features of Platform as a Service (PaaS), Software as a Service (SaaS) and Business Process as a Service (BPaaS). PaaS enables quicker and better software development and deployment by using on-demand development and execution tools catering to specific needs [1, 2]. SaaS provides on-demand access to various software packages and reduces efforts associated with software maintenance. In order to improve business and IT alignment in the cloud environment, a next level of abstraction is introduced – BPaaS [3, 4] Domaschka et al. [5] define that the key part of BPaaS is an ability to specify and executed distributed multi-tenant workflows and BPaaS should be naturally integrated with other layers of cloud computing. Customization is an important concern of BPaaS. Taher et al. [6] show that business processes can be customized on the basis of the meta-solution. A multi-layered approach to customization where different customization aspects are separated in dedicated layers contributes to tailoring business services to individual consumers [7]. Capability management in the cloud can be perceived as yet a higher level abstraction relative to BPaaS focusing on development of enterprise core competencies as a service offering.

The Capability Driven Development (CDD) methodology [8] can be used for capability development and it is supported by an Eclipse based capability design tool. This paper describes conversion of this tool for the cloud environment. However, the cloud-based CDD environment is not only a technological change, it also enables capability delivery as a service. A company possessing specific knowledge and resources of providing services in varying circumstances is able to specify those abilities in terms of the capability model and to provide the cloud-based CDD environment to offer them to potential consumers. Additionally, the cloud-based capability management both enables and benefits from capability delivery information sharing. The pattern repository [9] is the key component for information and knowledge sharing.

The paper describes the Capability as a Service (CaaS) capability management model and cloud-based CDD environment as a key enabler of this model. Additionally, this paper also shows usage of the CDD methodology to build cloud native capability delivery applications combining the cloud-based CDD environment and cloud ready CDA. These applications concern development and delivery of the scalability capability. Scalability, which is one of the requirements for cloud-based capability development and delivery, itself is context dependent [10], and the CDD approach can be used to develop the scalability capability. The scalability capability ensures that computational resources used by CDA are adjusted in response to the context situation.

The rest of the paper is structured as follows. Section 2 describes a cloud-based capability management model, which is supported by the cloud-based CDD environment presented in Sect. 3. Application of the cloud-based CDD environment is demonstrated in Sect. 4. Section 5 concludes.

2 Management Model

The CDD methodology supports capability delivery in various modes including internal capability development as well as provisioning capability as a service to external consumer. The cloud-based delivery is essential in the latter case. Two main delivery modes (Fig. 1) can be distinguished in the case of capability delivery as a service (CaaS). The service provider owns some of the knowledge and resources needed to deliver the capability and the service consumer uses this knowledge and resources to serve its customers or support her internal processes. The service consumer also contributes some of the knowledge and resources to capability delivery, chiefly in the form of knowledge and resources committed to running information systems involved in capability delivery referred as to Capability Delivery Applications (CDA). The first CaaS mode implies usage of the shared multi-tenant CDD environment. The second CaaS mode is deployment of the private CDD environment for every consumer (be it private or public cloud and operated by capability provider, service consumer or third party).

In the case of the shared multi-tenant mode, the capability owner has developed the capability design, which describes capability delivery goals, context, processes and context-dependent adaptions. The capability is deployed in a shared CDD environment. Multiple instances of the capability can be setup within this deployment and configured according to the needs of individual capability consumers. However, this setup is limited to providing individualized data binding for context data and consumer specific treatment of context and performance indicators.

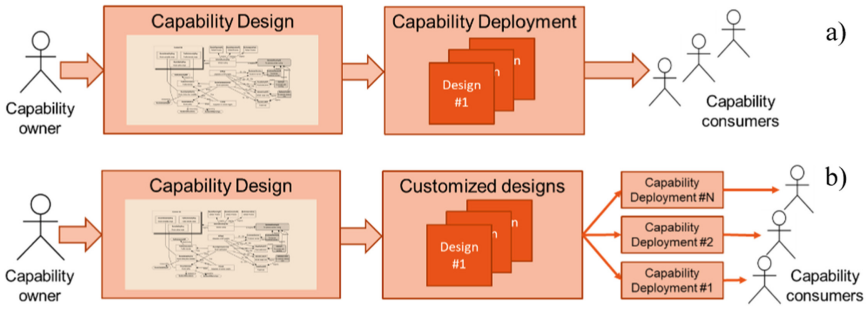

Fig. 1. CaaS design and delivery modes: (a) shared multi-tenant mode; and (b) private mode.

In the case of the private mode, the capability design is used as a reference model for creating customized designs for individual consumers. These customized designs are used to configure private capability deployment for each capability consumers. This way every consumer gets an individualized capability design, which supports unique requirements while also requires separate maintenance. From the cloud base capability management standpoint, it is important to emphasize that the customized designs still retain clearly identifiable elements from the reference design (Fig. 2) to enable information sharing among the capability consumers. The customized design consists of

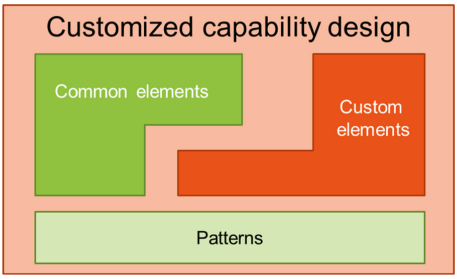

Fig. 2. Composition of the customized capability design

common elements inherited from the reference design, custom elements added to a design tailored for specific consumers and capability delivery patterns. The patterns are reusable chunks of capability design what are used to design and customize capabilities [9].

The CDD environment consists of the Capability Design Tool (CDT), the Capability Navigation Application (CAN), which is responsible for configuration of individual deployments, monitoring of capability delivery and context-dependent run-time adaption of capability delivery, the Capability Context Platform (CCP), which captures capability delivery context, and CDA (see [11] for more details). The CDD environment can be deployed on the cloud-based infrastructure for both CaaS delivery modes. Using the cloud-based infrastructure enables horizontal scalability of the CDD environment (see Sect. 3). Thus, the capability service provider is able to serve a large number of potential capability consumers.

3 Cloud-Based Deployment

All components of the CDD environment are deployed in the cloud environment (Fig. 3). The deployment backbone is the Infrastructure as a Service (IaaS) layer. In this case, open source Apache CloudStack[1] software is used to create and manage the IaaS layer. It allows managing large networks of virtual machines what is necessary for quick deployment of all components of the CDD environment. CDT and CCP form the Platform as a Service (PaaS) layer of the cloud-based CDD environment while CNA forms the Software as a Service (SaaS) layer. In the case of the private deployment mode, every capability consumer is provisioned with a set of virtual machines hosting CDT, CCP and CNA. Kernel-based Virtual Machine (KVM)[2], which is a full virtualization solution for Linux on x86 hardware, was chosen as a hypervisor for the cloud-based CDD due its open-source nature. While KVM was used to provision fully pledged virtual machines, Docker[3] allowed to host applications inside lightweight,

[1] https://cloudstack.apache.org/.
[2] https://www.linux-kvm.org/.
[3] http://docker.com/.

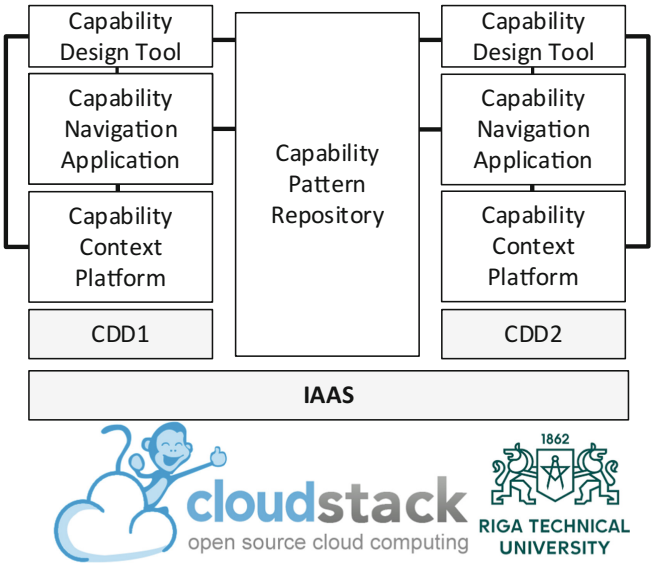

Fig. 3. Overview of cloud-based CDD environment

customized software containers. Experiments show that containerization results in equal or better performance than traditional virtual machines in almost all cases [12]. Docker was especially useful for CCP as it required Apache Camel, Apache ActiveMQ, PostgreSQL and Redhat Wildfly, which were deployed in a form of software containers on a single KVM virtual machine. This approach allows to run multiple isolated instances of CCP on a single virtual machine thus minimizing usage of cloud resources. CDA also could be deployed in the same cloud if requested by the consumer.

The capability pattern repository is managed by the capability service provider as a single instance. It is accessed by all capability service consumers and ensures information and knowledge sharing among all involved parties.

CDT is natively developed as an Eclipse based application. It is made available over the cloud using desktop virtualization technologies (Fig. 4). A single CDT virtual machine instance can be used by multiple users having either dedicated or shared workspaces. The cloud-based CDT supports all functionality of the desktop CDT, does not require installation of any specific software and is available on multiple devices and platforms.

The cloud-based CDD environment is vertical scalability. The components also can be made to support horizontal scalability. Both CNA and CCP of the single deployment can be replicated across multiple virtual machines though dynamic resource allocation is not supported out-of-the-box. A fully horizontally scalable context data integration, processing and adjustment solution is described in [13].

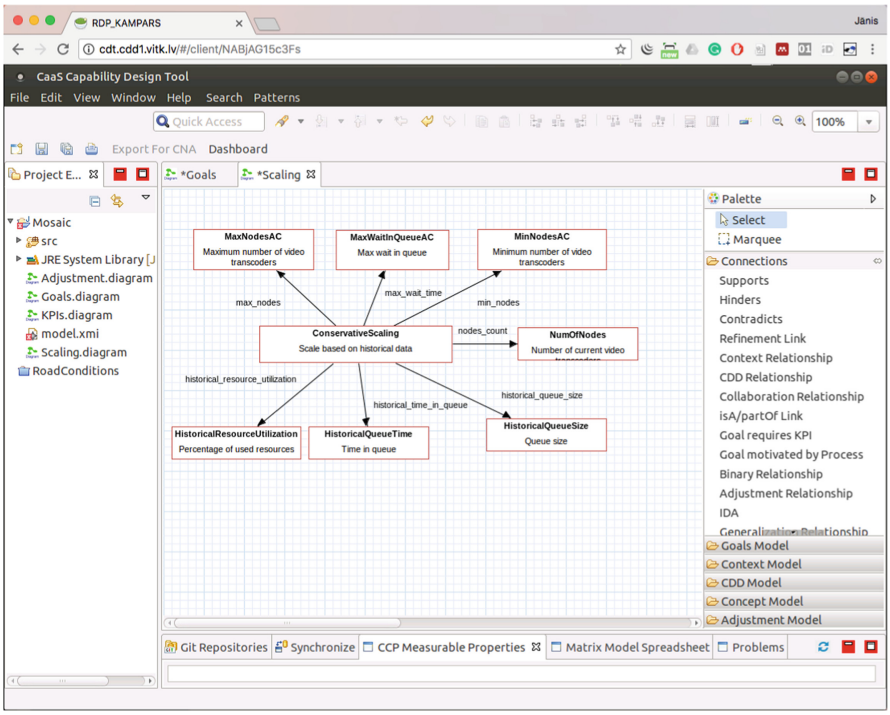

Fig. 4. User interface of the cloud-based CDT

4 Scalability Capability

The CDD methodology in combination with cloud-enabled capability management allows to develop highly scalable applications. That is demonstrated by development of a simplified auto-scaling capability using the cloud-based CDD environment and supported by cloud native CDA. It serves as a demo project that is shipped together with the cloud-based CDT. The CDA of the demo capability is a NodeJS[4] and AngularJS[5] based web application that can be used to generate a mosaic from image and keyword provided by a user. The logic of the CDA is shown in Fig. 5.

Once the user has submitted the mosaic generation form, the data about the mosaic generation job is added to a RabbitMQ[6] message queue. One of the worker nodes, implemented as Docker containers, picks up this job and starts the mosaic generation process. In order to find the small tiles that correspond to the user provided keyword it queries the Flickr API[7]. The list of relevant images is downloaded, they are resized and

[4] https://nodejs.org/.

[5] https://angularjs.org/.

[6] https://www.rabbitmq.com/.

[7] https://www.flickr.com/services/api/.

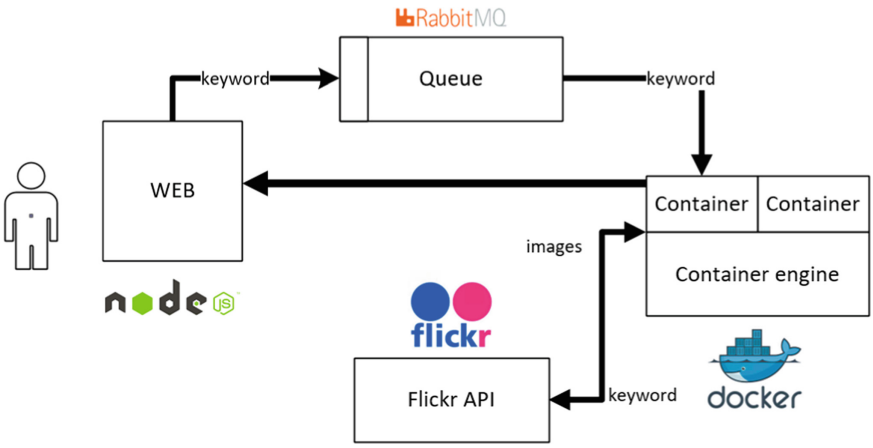

Fig. 5. Capability delivery application logic

matched with sections of the user provided image. The most similar image tiles are overlaid on top of the user provided image thus forming a mosaic. Finally, the generated mosaic is presented to the user of the CDA and user is asked to rate the experience. Statistics from the CDA like time in queue, rating, queue size, data retrieval time from Flickr, mosaic generation time, number of current nodes, number of busy nodes are made available to the CCP via a series of REST (Representational state transfer) web services. The corresponding configuration of the CCP is shown in Fig. 6.

The CDT model containing goals, KPIs, context set, context ranges, context elements and measurable properties is presented in Fig. 7.

Reactive

#	Name	Protocol	Measurable Property	Frequency	URI	Encryption Type	Created by
1675368	Queue size	REST	queue_size	10000ms	http://mosaic.vitk.lv/api/mp/queue_length	None	industry
1666381	Rating	REST	rating	10000ms	http://mosaic.vitk.lv/api/mp/rating	None	industry
1666395	Queue	MQTT	queue_time	10000ms	http://mosaic.vitk.lv/api/mp/queue	None	industry
1666401	Flickr	MQTT	flickr_time	10000ms	http://mosaic.vitk.lv/api/mp/flickr	None	industry
1666397	Mosaic	MQTT	mosaic_time	10000ms	http://mosaic.vitk.lv/api/mp/mosaic	None	industry
1666436	Nodes	REST	nodes	10000ms	http://mosaic.vitk.lv/api/mp/nodes	None	industry
1680892	Busy nodes	REST	busy_nodes	10000ms	http://mosaic.vitk.lv/api/mp/busy	None	industry

Fig. 6. CCP configuration for the scalability capability

The main goal of the capability is to ensure scalability of the mosaic generation application through minimizing cloud resource consumption and maximizing the Quality of Service. The number of busy nodes (Docker containers currently performing mosaic generation), queue size (unprocessed mosaic generation jobs stored in the

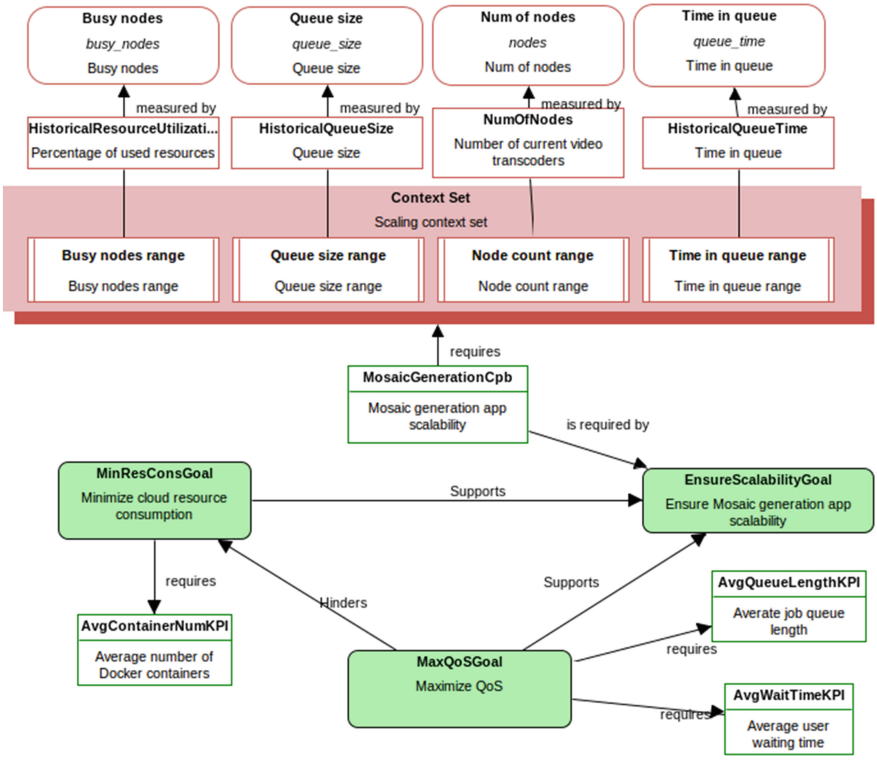

Fig. 7. Scalability capability

message queue), number of nodes (running Docker containers) and average time in queue serve as the context for the scalability capability. A scheduled adjustment is created to monitor the values of the context elements and to scale the mosaic generation application accordingly (see Fig. 8). Besides previously documented context elements it uses three adjustment coefficients that can be altered during run-time to change the scaling algorithm behavior (see Fig. 9). The scheduled adjustment is implemented as a Java class which makes a decision whether the mosaic generation application should be scaled down, up or left intact. The names on the arrows in Fig. 8 are equal to the names of variables that are made available in the adjustment for retrieving values of context elements and adjustment constants.

The source-code of the scheduled adjustment is given in Fig. 10. The method `this.scale()` is used for calling a REST scaling web-service that changes the number of running Docker containers during run-time.

The end results from the demo CDA and list of running containers retrieved from the Docker engine are shown in Fig. 11.

The results from command `docker ps` show that there are four running Docker containers. This information is also visible in the user interface of the CNA together with other context indicators like average waiting time and current queue length.

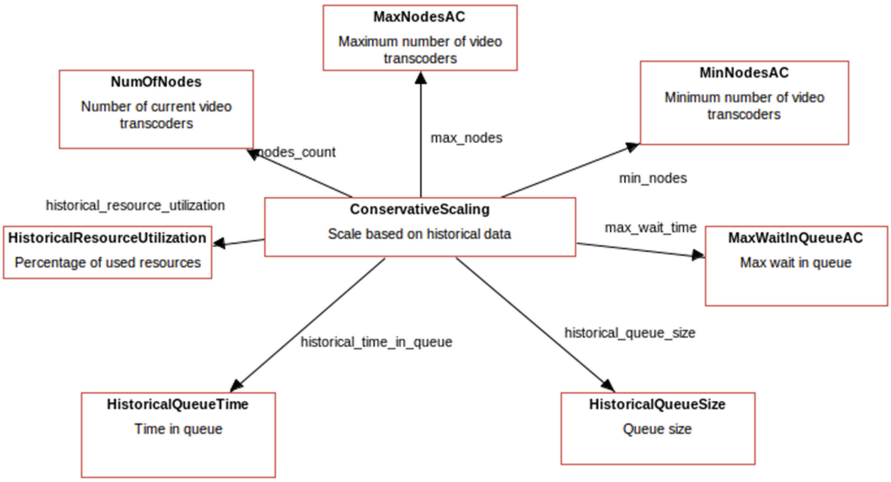

Fig. 8. Input data for the scheduled adjustment

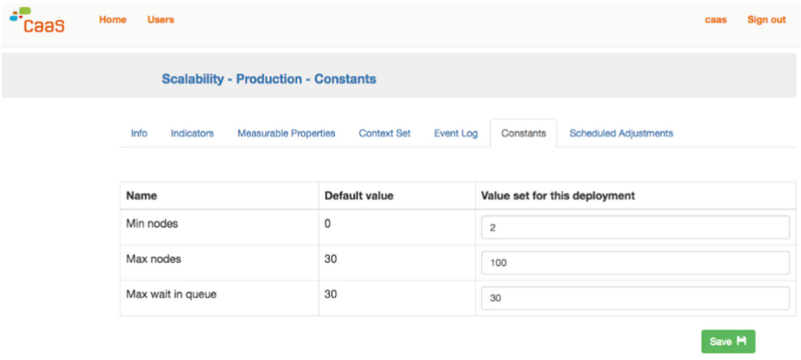

Fig. 9. Changing adjustment coefficients during run-time

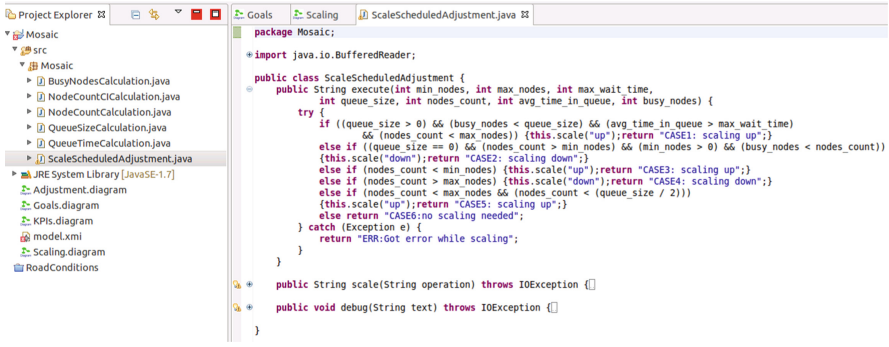

Fig. 10. Implementation of a scheduled adjustment

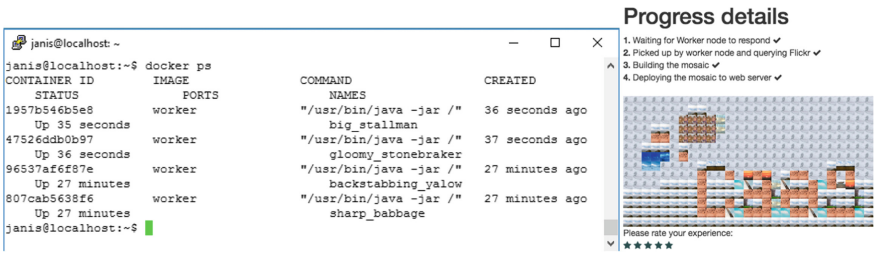

Fig. 11. Results from CDA and Docker engine status

5 Conclusion

This paper has described cloud-based capability management as an enabler of the CaaS approach. In comparison with typical service-oriented approaches, CaaS still requires a relatively high degree of collaboration between the capability provider and the capability consumer. Both parties are required to commit their abilities and resources to capability delivery. Additionally, one can argue that capabilities are traditionally viewed as a company's internal asset. Some of competencies and resources can be procured from providers, however, capability consumers are still expected to evolve the capabilities by themselves at least partially. Therefore, the private capability delivery mode involving capability design customization is suitable for the CaaS approach. The capability design customization leads to challenges associated with model management and handling of different versions of the capability design in a distributed environment. The service consumers also must have sufficient incentives for information sharing and a greater degree of customization potentially leads to lower returns on information sharing. This challenge relates to the overall issue of valuing and trading data what becomes more and more relevant in the area.

References

1. Cohen, B.: PaaS: new opportunities for cloud application development. Computer **46**(9), 97–100 (2013)
2. Gass, O., Meth, H., Maedche, A.: PaaS characteristics for productive software development: an evaluation framework. IEEE Internet Comput. **18**(1), 56–64 (2014)
3. Papazoglou, M.P., van den Heuvel, W.-J.: Blueprinting the cloud. IEEE Internet Comput. **15**(6), 74–79 (2011)
4. Barton, T., Seel, C.: Business process as a service - status and architecture. In: Proceedings - Series of the Gesellschaft fur Informatik (GI). Lecture Notes in Informatics (LNI), p. 145 (2014)
5. Domaschka, J., Griesinger, F., Seybold, D., Wesner, S.: A cloud-driven view on business process as a service. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science, CLOSER 2017, p. 739 (2017)
6. Taher, Y., Haque, R., Van Den Heuvel, W.-J., Finance, B.: α BPaaS - a customizable BPaaS on the cloud. In: Proceedings of the 3rd International Conference on Cloud Computing and Services Science, CLOSER 2013, pp. 290–296 (2013)

7. Taher, Y., Haque, R., Parkin, M., van den Heuvel, W.-J., Richardson, I., Whelan, E.: A multi-layer approach for customizing business services. In: Huemer, C., Setzer, T. (eds.) EC-Web 2011. LNBIP, vol. 85, pp. 64–76. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23014-1_6
8. Berzisa, S., Bravos, G., Gonzalez, T., Czubayko, U., España, S., Grabis, J., Henkel, M., Jokste, L., Kampars, J., Koç, H., Kuhr, J., Llorca, C., Loucopoulos, P., Juanes, R., Pastor, O., Sandkuhl, K., Simic, H., Stirna, J., Valverde, F., Zdravkovic, J.: Capability driven development: an approach to designing digital enterprises. Bus. Inf. Syst. Eng. **57**(1), 15–25 (2015)
9. Kampars, J., Stirna, J.: A repository for pattern governance supporting capability driven development. In: Johansson, B. (ed.) Joint Proceedings of the BIR 2017 pre-BIR Forum, Workshops and Doctoral Consortium co-located with 16th International Conference on Perspectives in Business Informatics Research (BIR 2017). CEUR-WS.org (2017)
10. Kampars, J., Pinka, K.: Auto-scaling and adjustment platform for cloud-based systems. In: Environment. Technology. Resources: Proceedings of the 11th International Scientific and Practical Conference, 15–17 June, vol. 2, pp. 52–57 (2017)
11. Henkel, M., Kampars, J., Hrvoje, S.: The CDD environment architecture. In: Sandkuhl, K., Stirna, J. (eds.) Capability Management for Digital Enterprises. Springer, Cham (2018)
12. Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and Linux containers. Technology **25482**, 171–172 (2014)
13. Kampars, J., Grabis, J.: Near real-time big-data processing for data driven applications. In: Proceedings of the 3rd International Conference on Big Data Innovations and Applications, Innovate-Data 2017, Czech Republic, Prague, 21–23 August 2017, pp. 35–42. IEEE, Piscataway (2017)