Melodic
Big data cloud

Title:
Explanation and education materials

Abstract:
The success of any software platform and its penetration to the target market and software communities, either commercial or research-based, is heavily dependent on the availability and quality of project documentation and training materials. These types of materials guide prospective users and developers to understand the platform features, demonstrate how the main functionality of the platform can be exploited to achieve user goals and requirements, and more importantly accelerate the adoption and dissemination of the platform. The purpose of this deliverable is to describe training materials that are made available by the Melodic project until the midterm of the project, including materials that are presented online at the Melodic website[1]. The training materials include: instructions on how to deploy and configure the Melodic platform and two use case applications (i.e., the SmartDesign and Genome applications), instructions on how to deploy and configure Cloudiator V2 as the ExecutionWare component of Melodic, hands-on session instructions for Melodic installation, and some educational documents on Melodic prepared by the Melodic industry partners. This deliverable will serve as a basis for deliverable D6.6, Guidebooks, which will be provided towards the end of the project.

Multi-cloud Execution-ware for Large-scale Optimised Data-Intensive Computing

H2020-ICT-2016-2017
Leadership in Enabling and Industrial Technologies; Information and Communication Technologies

Grant Agreement No.:
731664

Duration:
1 December 2016 -
30 November 2019

www.melodic.cloud

Deliverable reference:
D7.2

Date:
22 June 2018

Responsible partner:
University of Oslo

Editor(s):
Amir Taherkordi

Author(s):
Amir Taherkordi

Approved by:
Ernst Gunnar Gran

ISBN number:
N/A

Document URL:
http://www.melodic.cloud/deliverables/D7.2 Explanation and education materials.pdf

[1] http://melodic.cloud/education.html

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Document	
Period Covered	M8-18
Deliverable No.	D7.2
Deliverable Title	Explanation and education materials
Editor(s)	Amir Taherkordi
Author(s)	Amir Taherkordi
Reviewer(s)	Feroz Zahid and Florian Held
Work Package No.	7
Work Package Title	Exploitation and sustainability
Lead Beneficiary	University of Oslo
Distribution	PU
Version	1.0
Draft/Final	Final
Total No. of Pages	28 + Appendices

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

List of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1 Introduction

The Melodic platform may be exploited by different organisations with various goals. Some organisations might just desire to exploit the main features of the platform, such as data life cycle management or Cross-Cloud application deployment. The research community may use Melodic and extend it for research purposes, while others may intend to adopt and possibly extend it in order to develop a cross-cloud platform in the form of a business product, which can make a differentiation in the market and thus increase their market share. However, in order to expedite exploitation of the Melodic platform, sufficient documentation, as well as educational and training materials should be made available. They will present the various possible uses of the platform and details of all necessary steps and required knowledge that are needed to perform the appropriate steps towards its utilisation and fast adoption. This is basically the main purpose of this deliverable which is providing explanation and educational materials that can help Melodic adopters to exploit the Melodic platform.

The types of user that can benefit from the material presented in this document are the following:

- System admins: They can inspect the configuration and building guidelines provided in order to build and deploy the Melodic platform or its modules. Their specialized knowledge enables them not only to comprehend such guidelines, but also to implement them by also respecting their organisations' technical requirements, peculiarities and regulations and bypass any technical obstacles. This group of users are also in charge of finding the right way to check the operation status of the platform and perform the right actions to recover it in case of failures.
- Business users: Such users, which might not have a technical background in IT, can benefit from the material provided in order to obtain and share knowledge, and publish their application models. The models can be materialized into concrete CAMEL models by supporting the expression of business requirements for user applications in collaboration with other types of users from the same organisation. The CAMEL models specify deployment, scalability, quality and security requirements (more details about CAMEL is available in the deliverable "D2.2 Architecture and initial feature definitions" [1]). These models can then be issued into the Melodic platform for the proper modelling and management of respective applications in the cloud.
- Application developers/engineers: They can exploit the guidelines to obtain and share application design knowledge as well as publish application models for their organisations. They are also guided in the provision of requirements for their applications in terms of concrete CAMEL models. Such users are also responsible for the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

operation of the multi-cloud applications which means that they need to monitor the state of the application and perform any necessary steps to correct it, if needed. Moreover, they can exploit the presented material in order to extend particular Melodic platform components and produce new components that can be fitted into the platform providing added-value functionality.

1.1 Structure of the Document

The set of materials presented in this deliverable is structured into the following different sections:

- Section 2 presents material which explains how to configure, build and deploy the Melodic platform. This material will be of interest for organisations that want to create and configure their own platform, through which their data-intensive applications can be deployed across different clouds. This material is mainly intended for *system* admins involved in such organisations as it contains low-level technical details which might not be understandable, e.g., by business users. However, this does not mean that other types of users cannot exploit it to fulfil their respective tasks (e.g., when application developers need to deploy the platform for debugging or test purposes) as the description of the platform configuration and deployment process is quite straightforward.
- Section 3 presents material which explains how to install, configure and interact with Cloudiator. This material will be of interest for Melodic users who want to learn and understand how the ExecutionWare component (cf. Deliverable D2.2 [1]) works and how resource management is performed in Melodic. This material is mainly intended for system admins involved in Melodic platform deployment and configuration. Devops users can also use this material for resolving errors. It is also useful for users who just want to use Cloudiator as an orchestration tool and use its cross-cloud deployment feature.
- Section 4 presents step-by-step instructions for installing a sample application on the Melodic platform. It gives useful insights to the concept of application and application components.
- Section 5 and 6 are complementary to Section 4; these two sections provide complete and detailed instructions about the deployment of two use case scenarios, namely, the CRM frontend 'SmartDesign' and the Genome Application (more detail about these applications is available in deliverable "D2.1 System specification document" [2]). These two applications are designed and developed by the Melodic partners CAS and 7bulls, respectively.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Section 7 is dedicated to some online and offline materials published for Melodic educational purposes. The online materials are related to the documentation of Melodic and Melodic related tools, such as Cloudiator. In the case of Cloudiator, the material in Section 3 presents only the installation guide, while the online material provides a complete description of Cloudiator and its features. The offline materials are presentations and articles that are prepared by Melodic industry partners to be presented to interested customers.
- Section 8 provides conclusions and summary.

2 Melodic platform installation

This section guides installation of the Melodic platform.

2.1 Prerequisites

No prerequisite is required for installing the Melodic platform.

2.2 Requirements for Melodic's machine

Following are the requirements for the Melodic's machine:

1. Ubuntu 16.04 or higher operating system
2. The machine should have at least 8 GB of RAM
3. The ports listed in Table 1 should be accessible on that machine:

Table 1: Port Requirements for the Melodic's Machine

Port	Protocol	Component	Purpose
22	TCP	ssh	Console
8080-8099	TCP	components	REST endpoints of Melodic components (CP-Generator, CP-Solver, Solver2Deployment, Adapter, ESB, BPM)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

9001-10000	TCP		
2036, 3306	TCP	CDO	MySQL database
80	TCP	UI	Cloudiator's web interface
4001	TCP	Lance	etc registry
9000	TCP	Colosseum	Cloudiator's REST API
8080	TCP	Axe	Time-series database
33034	TCP	Lance	rmi registry

2.3 Installation steps

1. Perform SSH login into the machine (Ubuntu 16.04) with the default username e.g.'ubuntu'
2. Run the following command (this will download installation files):

```
git clone https://bitbucket.7bulls.eu/scm/mel/utils.git
```

3. Open Melodic's installation script using the editor of your choice, e.g. vi

```
vi ~/utils/melodic_installation/installMelodic.sh
```

4. Change the value of the NODEGROUP variable to, e.g., your own login (new login must use only lowercase letters).

```
NODEGROUP=jdoe
```

5. Run the Melodic's installation script

```
~/utils/melodic_installation/installMelodic.sh install
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6. After installation, a new ".profile" is created in the home directory of the user. This profile must be loaded by executing the following command:

```
cd ~/
. .profile
```

7. Verify that the Melodic configuration files use the right IP addresses (check files under ~/conf). Especially if you are running Cloudiator on a different host you need to update Cloudiator's IP in the eu.melodic.integration.mule.properties file). The list of files to be checked is:

```
eu.melodic.integration.bpm.properties
eu.melodic.integration.mule.properties
eu.melodic.upperware.adapter.properties
eu.melodic.upperware.cpSolver.properties
eu.melodic.upperware.generator.properties
eu.melodic.upperware.solverToDeployment.properties
```

This checking can be conducted by the following command (The command must not return any match)

```
cd ~/conf
grep "MELODIC_IP" *.properties;
```

8. Add Cloud Provider credentials to allow Melodic to download Provider Offers in the Adapter configuration file. For instance, if you want Melodic to manage Amazon AWS machines, you need to edit the file:

```
~/conf/eu.melodic.upperware.adapter.properties
```

and provide values to the following attributes:

```
clouds.endpoints.ec2=
clouds.logins.ec2=
clouds.passwords.ec2=
```

9. Add the Cloudiator's API key to the Generator's config file (this API key allows access to cloudiator's API). It can be received from the Cloudiator installation.

```
cloudiatorV2.apiKey=abcdefghijk12345687
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

10. Now the machine is ready to download and run the latest docker images of the Melodic components from the Melodic artefact repository. To download and start the components simply use the following command:

```
ddeploy
```

11. Running this for the first time can take more time as docker swarm is being initialised. After successful execution of the above installation scripts, all the Melodic platform components should have been started. You can check the status by running the following commands. A screenshot of expected CLI is given in Figure 1

```
dps
mping
```

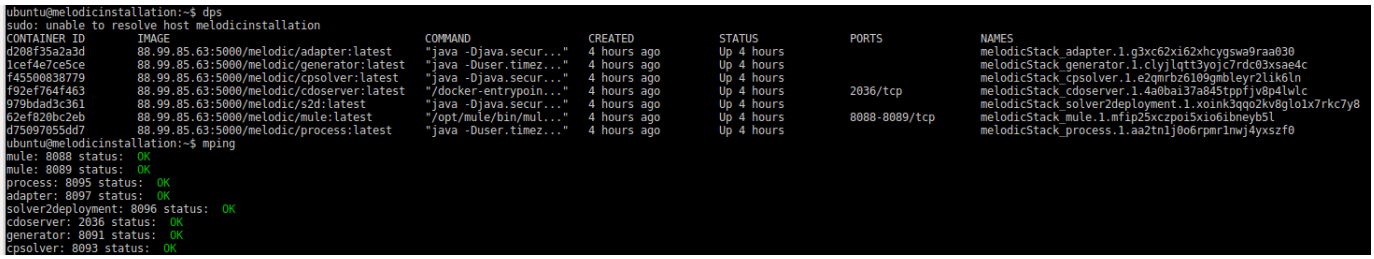


Figure 1 - Checking the status of Melodic installation by running dps and mping commands

The process GUI should be now available under:

```
http://{PUBLIC_MELODIC_IP}:8095 (admin:admin are default userid/password)
```

12. The status of the Colosseum (Cloudiator) service can be checked with:

```
sudo service colosseum status
```

13. The log files can be found under /var/log/colosseum.log, and the Cloudiator GUI should be available under:

```
http://{PUBLIC_CLOUDIATOR_IP}/ui (john.doe@example.com : admin : admin)
```

14. Script to restart Cloudiator with a full wipe out of Colosseum data is available here:

```
~/cloudiator_reset.sh
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2.4 Useful aliases

Below you find useful commands to manage Melodic components:

dps - displays docker containers running (alias for sudo docker images)
mping - tests connection to each of the components
drestart - stops and then starts all of the Melodic's components
dundeploy - stops all of the components
ddeploy - starts all of the components

3 Cloudiator v2 installation guide

This section describes how to install Cloudiator v2.

3.1 Requirements for the machine

Following are the requirements for the Cloudiator machine:

1. Ubuntu 16.04 or higher should be the operating system of the machine in which Cloudiator will be installed
2. The machine should have at least 8 GB of RAM
3. The required ports, as given in Table 2 should be accessible

Table 2: Port requirements for the Cloudiator machine

Port	Protocol	Component	Purpose
22	TCP	ssh	Console
80	TCP	UI	Cloudiator's webinterface
4001	TCP	Lance	etcd registry
9000	TCP	Colosseum	Cloudiator's REST API
8080	TCP	Axe	Time-series database
33034	TCP	Lance	rmi registry

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3.2 Installation steps

1. Login into the machine using SSH, with the default user name, e.g. 'ubuntu'
2. Run the following commands (this will download installation files):

```
git clone https://bitbucket.7bulls.eu/scm/mel/utils.git
```

3. Run the Cloudiator's installation script:

```
sudo ~/utils/melodic_installation/installCloudiatorV2.sh install
```

4. To start the Cloudiator execute the following commands:

```
cd ~/cloudiator/
sudo docker-compose create && sudo docker-compose up
```

5. To stop the Cloudiator use the following command:

```
sudo docker-compose down
```

3.3 REST interface

On the following page you will find the description of common REST requests from an operational point of view.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
{
        "endpoint": "",
        "cloudType": "PUBLIC",
        "api": {
            "providerName": "aws-ec2"
        },
        "credential": {
            "user": "YOURAWSUSER",
            "secret": "AWSSECRET"
        },
        "cloudConfiguration": {
            "nodeGroup": "cloudiator",
            "properties": [
                {
                    "key": "sword.ec2.ami.cc.query",
                    "value": "image-id=ami-58d7e821"
                },
                {
                    "key": "sword.ec2.ami.query",
                    "value": "image-id=ami-58d7e821"
                }
            ]
        },
        "id": "7697dce7103d5926a4027d17ad975446"
    }
```

In order to feed the Cloudiator with Cloud Providers data, simply send a POST request to the http://{{cloudiator-host}}:9000/clouds where the {{cloudiator-host}} is the IP address of your Cloudiator machine. A sample BODY (change the user/secret and maybe image-id with values that suit you) is given below.

```
Headers:
Accept:application/json
 Content-Type:application/json
 x-api-key:{{x-api-key}}
```

4 Hands-on Session Instructions

4.1 Application overview

A two components application named as TwoComponentApp will be installed with the following configuration (mainly focusing on its components):

1. Application server: REST-controller (with the ability to read/add objects from the web interface)
2. DB: M database storing application data

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.2 Prerequisites

1. A clean Ubuntu 16.04 LTS machine available with SSH access (or an AWS account with the ability to create t2.large machines for Melodic) and Melodic's required ports opened (as described in Section 2.1)
2. An AWS account exists to allow provisioning of machines for application deployment with the use of Melodic
3. The postman tool has been installed

4.3 Melodic installation

The Melodic installation guide is provided in Section 2.

4.4 Application deployment

1. Download the test data (xmi files, cdo uploader)

```
~/utils/melodic_installation/store.sh init
```

2. Upload xmi files for the TwoComponentApp

```
~/utils/melodic_installation/store.sh storeTwoComponentApp
```

3. Log in to GUI (process GUI, Executionware GUI)
4. Launch Postman and send deployment request:

```
POST http://{{melodic-host}}:8088/api/frontend/deploymentProcess
header: Content-Type: application/json

Sample body:
{
     "applicationId": "TwoComponentApp",
     "watermark": {
         "user": "myName",
         "system": "UI",
         "date": "2017-27-11T16:41:41+0000",
         "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
     }
 }
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. Check the result of the deployment process via the process GUI, ExecutionWare GUI, AWS console.

4.5 Verification of the application installation

The verification is doing by accessing the following URL:

```
http://{{application-host}}:9999/demo/add?name=Pawel&email=pszkup2@o2.pl
http://{{application-host}}:9999/demo/all
```

5 Deploying the SmartDesign Application via Melodic

To deploy an application via Melodic, the (authenticated) evaluating person uses the UI provided by Melodic to perform all necessary steps, which include:

- Defining the CAMEL model for the application
- Making the binaries of the application available
- Storing the CAMEL model in the platform.

The scenario can be considered 'executed', when the application is 'setup' and deployable.

5.1 Description of Implementation

Based on the description of the CRM use case application in the deliverable "D6.1 Evaluation Framework and Use Case Planning" [3], a refined CAMEL model was created. This model describes the CRM frontend 'SmartDesign' as an application with the frontend itself as the only component (cf. Figure 2). It furthermore describes that this component supplies the port '9494' for communication and need to be installed on a machine for which the following set of hardware requirements needs to hold:

- 8 GB RAM
- 2 CPU Cores
- Ubuntu Linux as operating system

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

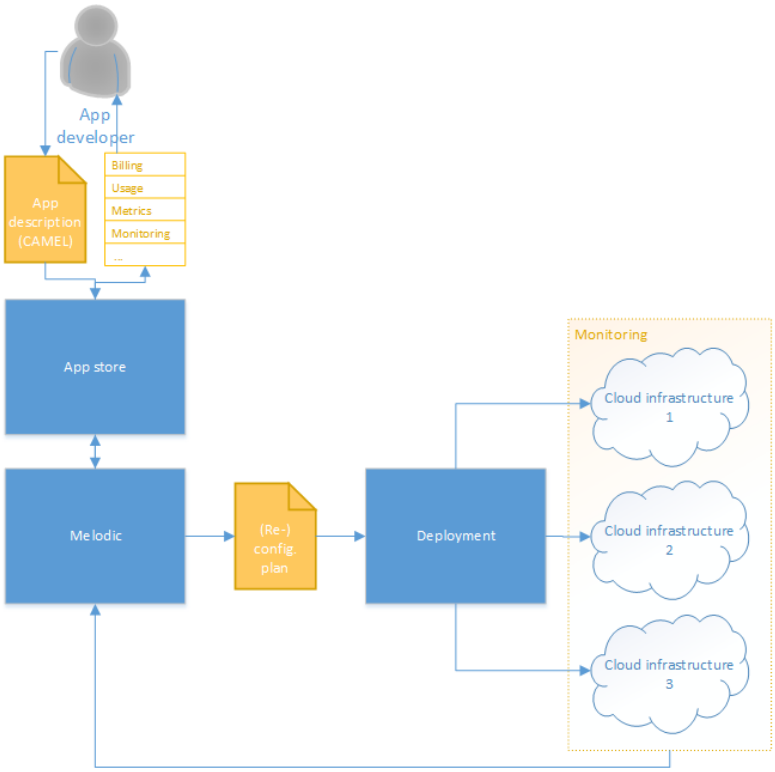


Figure 2 - CAS SmartDesign app store architecture with Melodic

The configuration of the component lifecycle includes three commands that need to be executed during application deployment; the *downloadCommand*, the *installCommand*, and the *startCommand:*

downloadCommand:

The download command shown in Listing 1 ensures that the application artefact code, additional configuration and the runtime environment are downloaded onto the allocated target machine:

- SmartDesign frontend Java artefact, ~135MB
- SmartDesign configuration providing custom application settings
- SmartDesign resources containing style/custom theme
- Java Runtime Environment 8 (Update 172)

```
sudo apt-get -y install wget && wget http://$OPERATOR/vaadin-smartdesign.war
    && wget http:// $OPERATOR/smartdesign.config && wget http:// $OPERATOR/sd-
resources.zip
    && sudo wget http:// $OPERATOR/jre-8u172-linux-x64.tar.gz -O java.tar.gz
```

Listing 1 - CRM download command

installCommand:

The install command shown in Listing 2 creates directories on the target machine and extracts the previously downloaded archives. Two of these directories are application-specific:

- 'sd-apps/' is an initially empty folder where external apps are placed
- 'sd-resources/' contains custom themes that are applied on application start

```
sudo apt-get install unzip && sudo mkdir java8 && sudo tar zxvf java.tar.gz -C java8
--strip-components=1
    && sudo mkdir sd-apps && unzip sd-resources.zip -d sd-resources
```

Listing 2 - CRM install command

startCommand:

This command as shown in Listing 3 performs the actual application start by executing the previously downloaded SmartDesign application artefact with the respective executable file within the Java 8 runtime environment. Execution-related configuration parameters are directly applied as program arguments, e.g.:

- The TCP port under which the application should be reachable
- The backend service and WSDL URLs the application connects to
- Tenant and user details that will be prefilled on the application's login page

```
java8/bin/java -XX:+UseG1GC -Xms512m -Xmx2048m -jar vaadin-smartdesign.war
-verbose true
-port 9494
-serverUrl http://$SMARTWE-BACKEND:8080/SmartWe/services
-serverWsdlUrl http://$SMARTWE-BACKEND:8080/SmartWe/eim.wsdl
-serverType OPEN -database melodic -username "Robert Glaser"
```

Listing 3 - CRM start command

The CAMEL model is then converted into its XMI (XML Metadata Interchange) representation by using the Eclipse-based CAMEL editor and can be uploaded to the model repository that runs under the internal name 'CDOServer' on the Melodic machine. The upload is done via the java-based CDOUploader that is available within the Melodic project. At CAS Software AG a UI wrapper was created to facilitate the handling of the model maintenance by offering an interactive way to upload models and to track the state of each upload. Figure 3 shows the program with the redirected console output of the original CDOUploader on the left side, an upload area in the middle that allows convenient model uploads via drag'n'drop, and an overview area on the right side showing the upload history and the configuration used. Once the model is successfully added to the model repository on the Melodic machine, a success message is printed in the console and the status changes from 'PROCESSING..' to 'SUCCESS'. In case of an error, the status becomes 'ERROR' and the console shows a more detailed explanation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

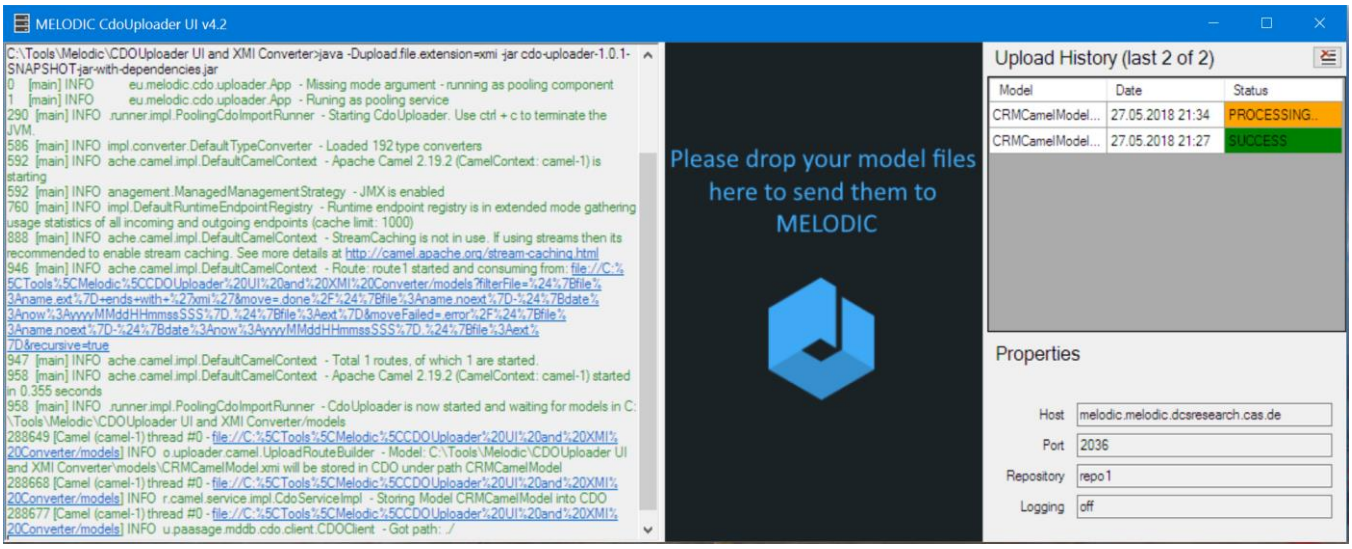


Figure 3 - UI wrapper for the java-based CDOUploader

After a successful model upload the application is considered being added and potentially ready for deployment. On the Melodic server, a good way to verify a properly uploaded model is to watch the CDOServer log file. As shown in Figure 4, a successful upload on a blank CDOServer database would state that there were tables missing and that the schema was updated.

```
root@melodic:~/logs# tail cdoserver.log -n5
2018-05-27 11:55:22.725  INFO 24 --- [net4j-Thread-1] java.sql.DatabaseMetaData                : HHH000262: Table not found: camel_virtualmachineprofile
2018-05-27 11:55:22.726  INFO 24 --- [net4j-Thread-1] java.sql.DatabaseMetaData                : HHH000262: Table not found: camel_warprofile
2018-05-27 11:55:22.727  INFO 24 --- [net4j-Thread-1] java.sql.DatabaseMetaData                : HHH000262: Table not found: camel_webserverprofile
2018-05-27 11:55:22.727  INFO 24 --- [net4j-Thread-1] java.sql.DatabaseMetaData                : HHH000262: Table not found: camelcontinentupperware
2018-05-27 11:55:35.300  INFO 24 --- [net4j-Thread-1] org.hibernate.tool.hbm2ddl.SchemaUpdate  : HHH000232: Schema update complete
```

Figure 4 - CDOServer log file

It turned out to be best practice to reset the CDOServer to an empty model repository whenever Melodic is restarted. To perform this reset conveniently, a bash script that removes corresponding tables was written during the use case implementation and used directly from the Melodic machine's command line.

5.2 Deploy Application

The application is considered deployed when Melodic confirms the deployment (i.e. by checking the UI, Monitoring) and the application (and its components) is (are) up and accessible (application specific verification).

With an existing, configured and running Melodic environment and the application model successfully uploaded to it, the initial deployment of the application itself can be performed anytime. Melodic exposes a REST endpoint named 'deploymentProcess' that can be used to trigger the actual deployment process. The message body expects a predefined JSON object as

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

shown in Listing 4. Besides the application ID, some additional meta information is expected that does not yet have any impact on the deployment itself. Melodic uses the application ID to load the correct application model and initiates the deployment on all relevant components. The application ID corresponds to the file name of the XMI representation of the application's CAMEL model. The response body contains information about the state (e.g., success) and the unique process ID that identifies the ongoing deployment process.

```
{
  "applicationId": "CRMCamelModel",
  "watermark": {
    "user": "RobertGlaser",
    "system": "CRM",
    "date": "2018-02-28T16:00:00+0000",
    "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
  }
}
```

Listing 4 - POST message body to trigger application deployment

Within the pilot implementations at CAS Software AG, a UI-based tool was developed that facilitates triggering the deployment process during the development. Figure 5 shows the UI consisting of connection parameters, a status window, a drag'n'drop area to load the model, followed by a 'Deploy Application' command button. The model is currently used to derive the application name that is used within the 'applicationID' field of the JSON request body.

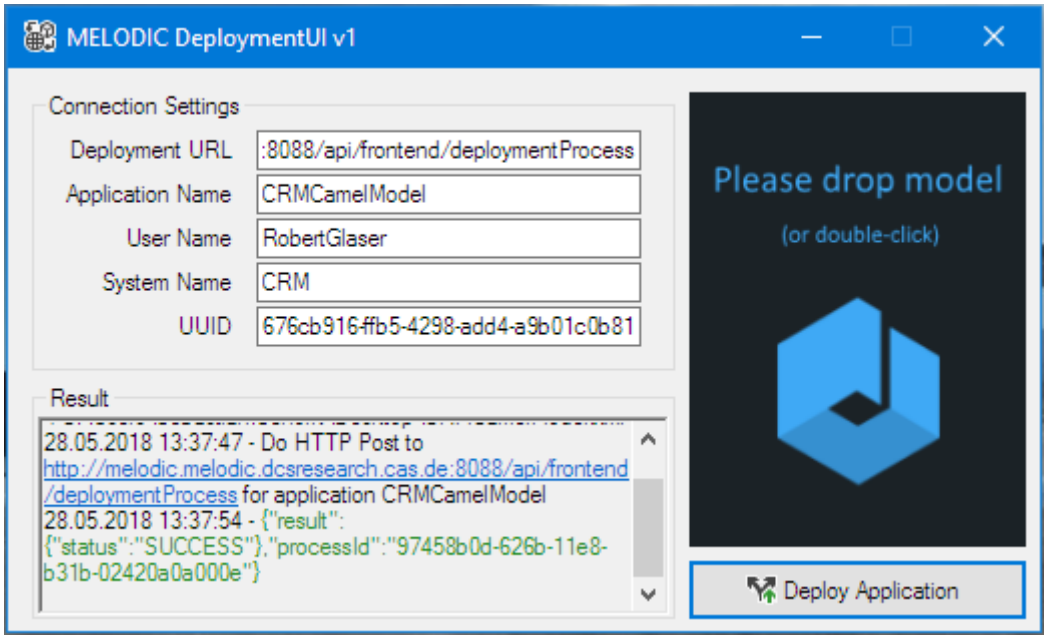


Figure 5 - UI-based deployment tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Nevertheless, the validation of a deployment has to be done manually. During the deployment process of the use case application, the log files provided by Melodic for each component (including, e.g., CP generator, solver) were used to keep track of the current deployment step and to debug failed deployments from a low-level perspective. For a more high-level view on the deployment status, the Camunda BPM cockpit hosted on the Melodic machine was used since it offers a quick overview of the currently active step or component.

A more application-related validation of a successful deployment was done by using the Cloudiator frontend provided by Melodic as shown on Figure 6. This UI originates from the PaaSage project[2] and was slightly changed in its appearance (favicon, title and logo) by the pilot partner to meet both the projects branding and the pilot requirements of a visually unified UI. It allows browsing through the following information:

- The application and its components (as described in the CAMEL model),
- The discovered cloud offerings,
- The current deployment (if successful), like the virtual machines allocated and their IP addresses.

Based on the provided IP addresses, the application VMs can be reached via SSH and in case of a successful deployment as well under the application-specific port they expose. In case of SSH, the generated private key for the machine has to be queried beforehand. This is done by using a shell script directly on the Melodic machine that outputs the key from the underlying MySQL database.

[2] https://paasage.ercim.eu/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

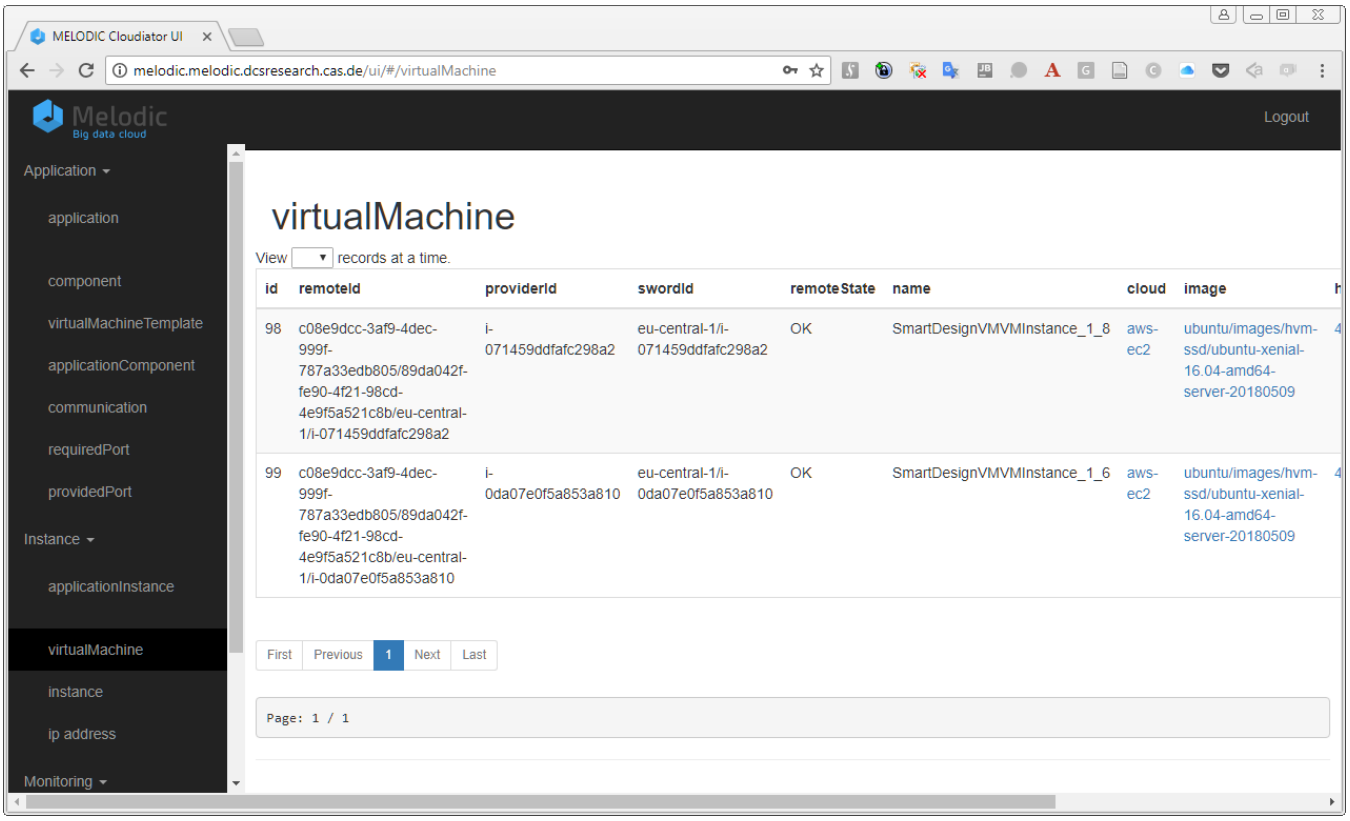


Figure 6 - Cloudiator UI served by Melodic

The application itself was then validated by manually mocking user interaction that covers both basic and component-specific functionality.

To remove existing application deployments, the management console of the cloud provider (e.g., Amazon AWS) was used. Once logged in, existing instances can be revised and if necessary terminated. Figure 7 shows the management console of Amazon AWS with two instances of the SmartDesign CRM application. Having two initial instances of SmartDesign is the result of the horizontal scale requirement that demands a minimum of 2 and a maximum of 6 machines for the sole component of this application.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
casmelodic-d8x0n-SmartDesignVMVMInstance_1_8	i-071459ddfafc298a2	m4.large	eu-central-1c	running	2/2 checks ...
casmelodic-s8mus-SmartDesignVMVMInstance_1_6	i-0da07e0f5a853a810	m4.large	eu-central-1c	running	2/2 checks ...

Figure 7 - Amazon AWS management console with 2 application instances

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6 Deploying Genome Application via Melodic

To deploy an application via Melodic, the (authenticated) person uses the UI provided by Melodic to perform all necessary steps:

- Defining the CAMEL model for the application
- Making binaries of the application available
- Storing the CAMEL model in the platform.

6.1 Description of Implementation

The Genome app is modelled as a two-component application comprising Master and Worker components. The Master component should have exactly 1 instance, while the worker components can have from 2 to 10 instances (cf. Figure 8).

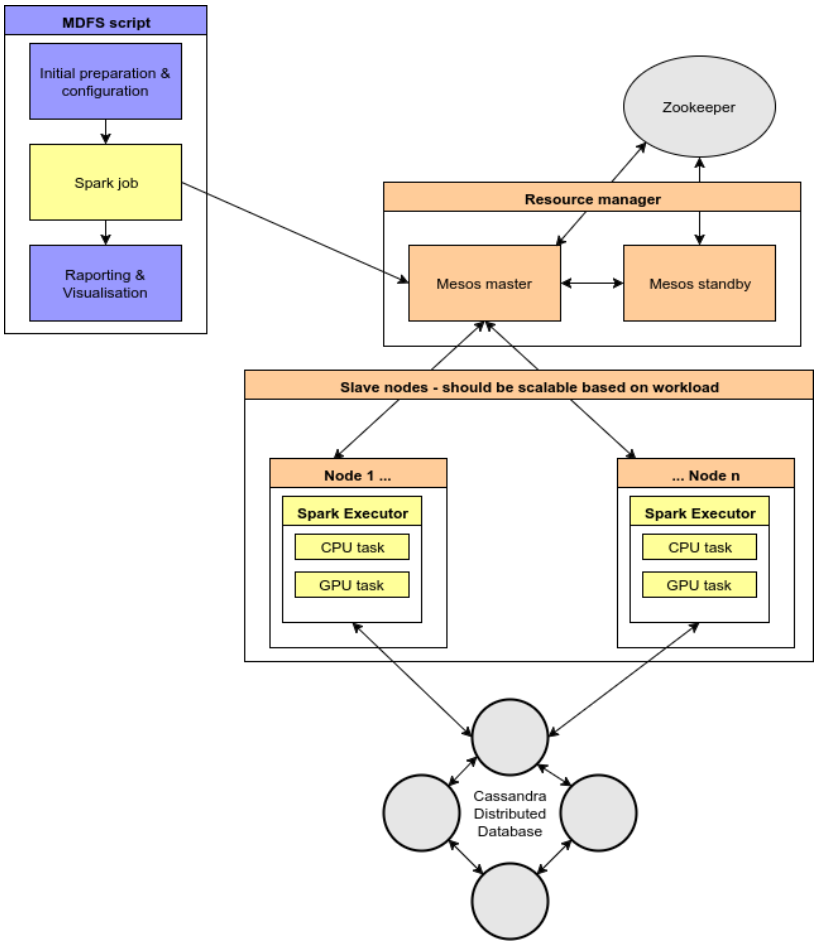


Figure 8- Preliminary application architecture for gnome application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

The following requirements were set up in the CAMEL description:

1. For Master component:
   a. Number of instances: 1
   b. Number of cores per instance: 1..2
   c. Amount of memory (MB): 8000..9000
   d. Custom image (Ubuntu 16.04 with pre-installed software)
2. For Worker component:
   a. Number of instances: 2..10
   b. Number of cores per instance: 2
   c. Amount of memory (MB): 8000..9000
   d. Custom image (Ubuntu 16.04 with pre-installed software)

6.2 Installation and deployment on a single Cloud provider

Input Conditions:

3. Installed and configured Melodic platform, without any application related artefacts.
4. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
5. Meta solver configured to use CP Solver for that case.
6. Cloudiator properly connected to the given Cloud Provider.
7. A complete CAMEL model of the application (which includes the definition of these two components and their installation/maintenance scripts) should have been prepared.

Steps to Complete:

1. Login to the machine with installed MELODIC using SSH:

```
ssh melodic@<VM IP>
```

2. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from:

```
https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
```

3. Upload models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi, cpGenerator-providerTypes.xmi, cpGenerator-operatingSystems.xmi into the "/home/user/models" directory
4. Upload Provider model AmazonEC2.xmi into the "/home/models/upperware-models/fms" directory

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5. Upload the CAMEL model TwoComponentApp.xmi into the "/home/models" directory
6. Using SoapUI tool, execute following steps:
    a. Create new REST project with URL:

```
http://<VM IP>:8088/api/frontend/deploymentProcess
```

    b. Using POST method start process
7. Start the deployed application on page:

```
http://"public ID of created Virtual Machine":9999/demo/all
```

Expected results:

1. Two virtual machine instances (of the same VM flavour/offering) should be created using the selected Cloud Provider.
2. The application should be installed on those VM instances (one business logic component instance should be installed on the first VM instance and the database component instance should be installed on the second VM instance).
3. The application should run properly, which actually involves that proper communication between application components has been established (the Wordpress page from the database should be displayed correctly).

6.3 Installation and deployment on two Cloud providers

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.
2. At least two cloud providers integrated with MELODIC platform, where the user has provided his/her own credentials for both of them (included in the CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to given Cloud Providers.
5. Complete CAMEL model of the two-component application (which includes the definition of these two components and their installation/maintenance scripts). One component can map to the main business logic of the application and the other to the underlying database used. An application is Wordpress which also includes an underlying MySQL database.
6. CAMEL model of each Cloud Provider prepared with at least one virtual machine offer provided per each provider.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

7. There should be a proper configuration of the virtual machine both in the CAMEL Providers model and on the Cloud Providers sides.
8. There should be a requirement in the application CAMEL model to use different Cloud Providers (this could be done in different ways; for example, by placing a location requirement that is then referenced in the virtual machine requirement set).

Steps to Complete:

1. Login to the machine with installed Melodic by using the following steps:

```
ssh melodic@<VM IP>
. .profile
```

2. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from:

```
https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
```

3. Upload the models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi, cpGenerator-providerTypes.xmi, cpGenerator-operatingSystems.xmi into the "/home/user/models" directory
4. Upload Providers AmazonEC2.xmi and OpenStackUlm.xmi into the "/home/models/upperware-models/fms" directory

5. Upload the CAMEL model TwoComponentApp.xmi into the "/home/models" directory
6. Using the SoapUI tool, execute the following steps:
   a. Create new REST project with URL:

```
http://<VM IP>/api/frontend/deploymentProcess
```

   b. Using the POST method start process
7. Start deploying of application

```
http://"public ID of created Virtual Machine":9999/demo/all
```

Expected results:

1. Two VM instances should be created, one instance per each Cloud Provider.
2. The application should be installed on those two VM instances (one business logic component instance should be installed on the first VM instance and the database component instance should be installed on the second VM instance).
3. The application should be run properly (The Wordpress web page from database should be displayed correctly).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

7 Other Materials

7.1 Online materials

There are a number of educational materials which are available online and are related to Melodic. In this subsection, we mention these types of materials.

On the website of Cloudiator, we have provided a detailed explanation of Cloudiator V2[3]. As shown in Figure 9, it is categorized into the following sections: main architectural concepts of Cloudiator, its installation and configuration, the modelling aspects, and a tutorial which provides a detailed step-by-step guide, helping the reader to deploy a simple application using the Cloudiator toolset.

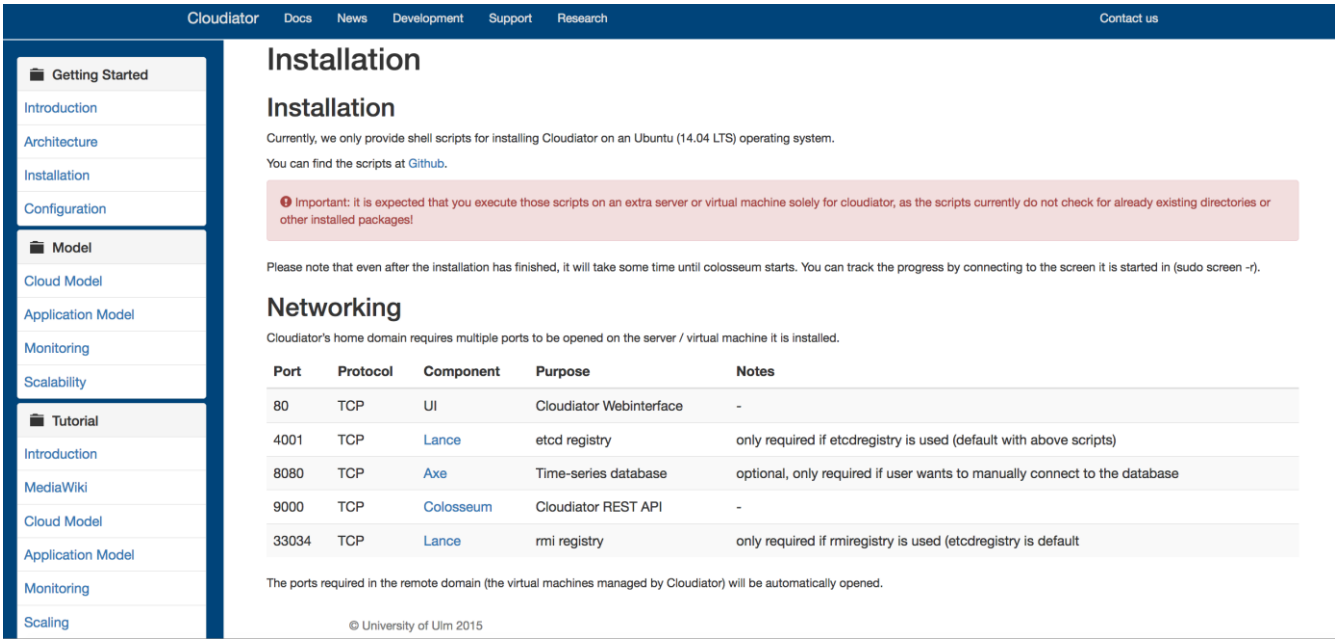


Figure 9 – A snapshot of Cloudiator V2 documentation webpage

7.2 Offline materials for presentation purposes

The industry partners in Melodic have been actively presenting and educating Melodic and its features to interested customers. Below, we list the materials presented, along with the PDF

[3] http://cloudiator.org/docs/introduction.html

version of each material attached to the end of this document. First, all documents in their original language are attached, then the English translated versions are attached at the end.

1. *Cloud Computing from AWS to Melodic (original language: Polish):* Materials for workshop dedicated to fundamentals of cloud computing and evolution of cloud computing, from the simplest models like IAAS through PaaS, CaaS, FaaS till multicloud and Melodic.
2. *Cloud Computing security (original language: Polish):* Materials for a dedicated workshop for Polish Bank Association about Cloud Computing solutions security with presentation of security aspects of Melodic.
3. *Multicloud: automation and optimization of processing in cloud environment (original language: Polish):* Article was published in the IT industry magazine ITWIZ (paper edition) which describes Melodic and the multicloud approach.
4. *Multicloud – diversification, optimization and security in cloud environment (original language: Polish):* This article presents Melodic and some security aspects of cloud computing which have been published in the Polish Bank Association journal.

8 Conclusions

This deliverable has provided training and educational material as well as documentation which can be used as a guide for various types of users in order to perform a variety of tasks, such as the deployment and configuration of the Melodic platform and Cloudiator, learning quickly how to deploy an application on Melodic, the steps to deploy real use case scenarios, and other on-line and off-line materials related to educating and presenting Melodic to interested parties.

While the above materials constitute an initial effort in guiding users on how to exploit the Melodic platform, which is in line with the deployment status of the platform itself, it is expected that the existing material will be further enhanced, while new material will be developed and be reported in the upcoming deliverable D6.6 “Guidebooks”.

References

[1] Y. Verginadis, G. Horn, K. Kritikos, F. Zahid, D. Baur, P. Skrzypek, D. Seybold, M. Prusiński and S. Mazumdar, *D2.2 Architecture and Initial Feature Definitions,* The Melodic H2020 project, 2018.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731664

[2] Y. Verginadis, W. Zolnieroiwcz, P. Skrzypek, D. Seybold, K. Kritikos, S. Mazumdar, A. Schwichtenberg, F. Zahid, J. Domaschka, G. Horn, E. G. Gran, D. Baur, H. Masata and P. Gora, *D2.1 System specification document,* The Melodic H2020 project, 2017.

[3] S. Kicin, S. Schork, A. Schwichtenberg, G. Horn, P. Gora, T. Przeździęk and M. Semczuk, *D6.1 Evaluation Framework and Use Case Planning,* The Melodic H2020 project, 2018.

Appendices

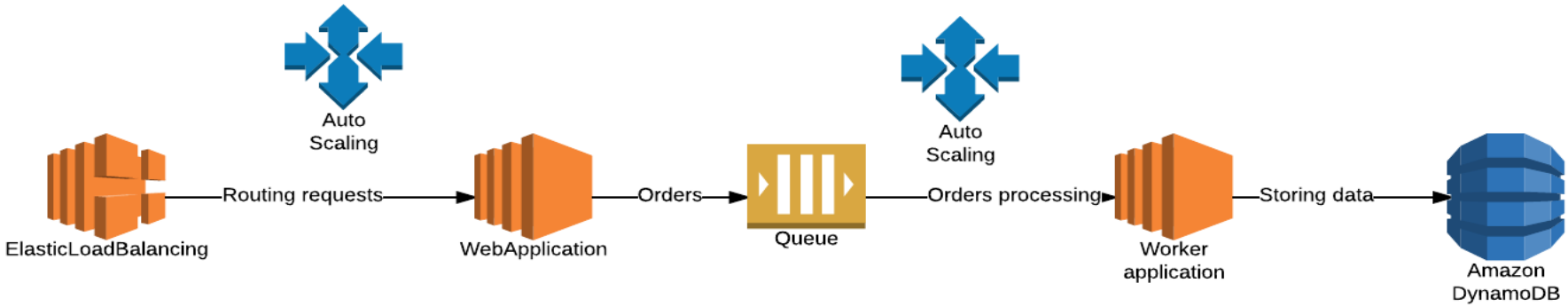
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloud Computing
Od AWS po MELODIC

Katarzyna Materka

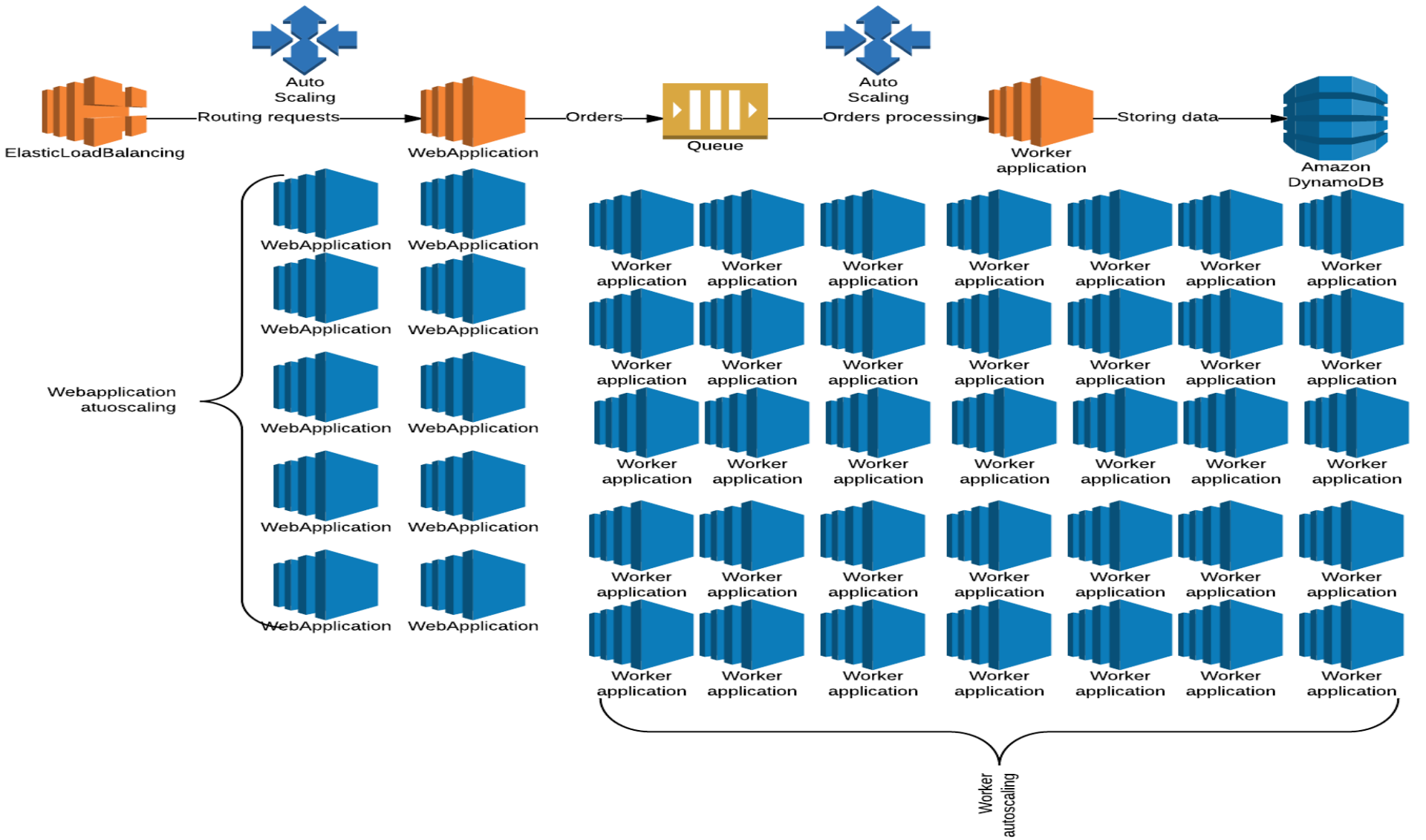
Paweł Skrzypek

Architektura systemu przetwarzającego obrazy



Przetwarzanie około 5000 requestów na godzinę przez jedną instancję workera (4 core, 16 GB RAM)

Architektura systemu przetwarzającego obrazy



Po autoscalingu przetwarzanie 175 000 requestów na godzinę, łącznie 140 core,560 GB RAM. Czas podniesienia konfiguracji 5 do 10 minut.

Wszyscy robią chmurę –

ale,

co to jest,

jak to zrobić,

a przede wszystkim po co?

Cloud Computing? Chmura? Co i jak?

- SaaS
- IaaS
- PaaS
- CaaS
- FaaS
- StaaS
- I inne

- Virtual machines
- VMWare
- Docker
- Kubernetes
- OpenStack
- OpenShift
- I inne

- AWS
- Azure
- Google Cloud
- Heroku
- Hetzner
- Aruba
- I inne

- EC2
- Dynos
- SmartCont
- EBS
- Lambda
- VPS
- Bare metal
- I inne

Co wyniosę ze szkolenia?

1. **Zrozumienie co to jest „chmura”**
2. Wiedzę **jak wykorzystać możliwości** – racjonalnie
3. Rozwiać mity o chmurach

?

Cloud Computing - mity

1. Bezpieczeństwo
2. Dane (w szczególności poufne)
3. Vendor lock
4. Rozwiązania hybrydowe
5. Mitologiczny marketing

Podstawowe pojęcia – zanim dojdziemy do Chmury

1. **Podstawowe pojęcia**
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

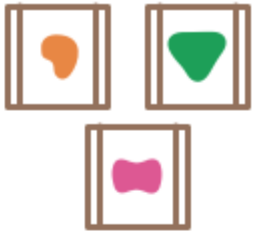
Zanim dojdziemy do Chmury

1. Architektura microservices vs SOA
2. Wirtualizacja
3. Kontenery
4. Platforma dla aplikacji
5. Orkiestracja usług
6. DevOps

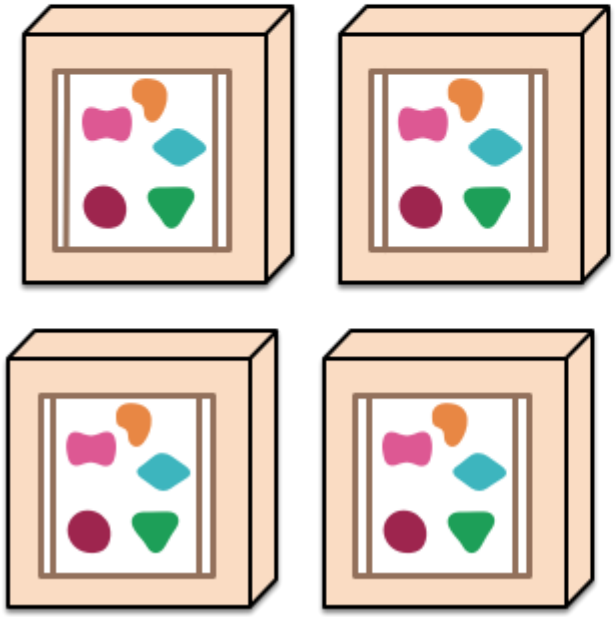
Architektura microservices

A monolithic application puts all its functionality into a single process...

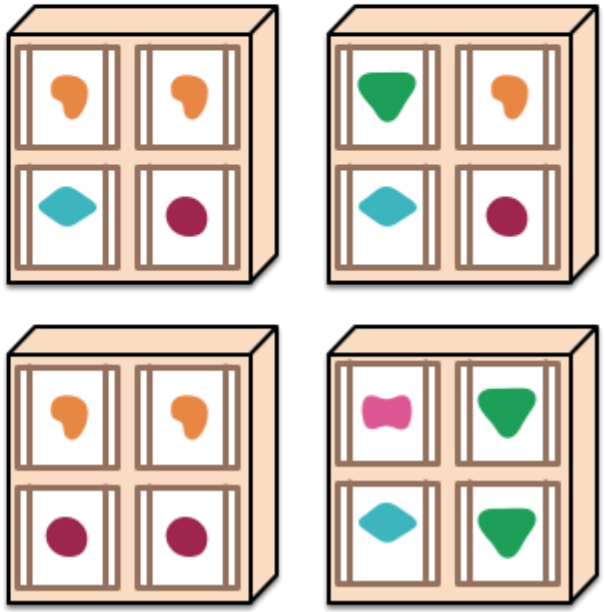
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers



... and scales by distributing these services across servers, replicating as needed.



Microservices to nie SOA

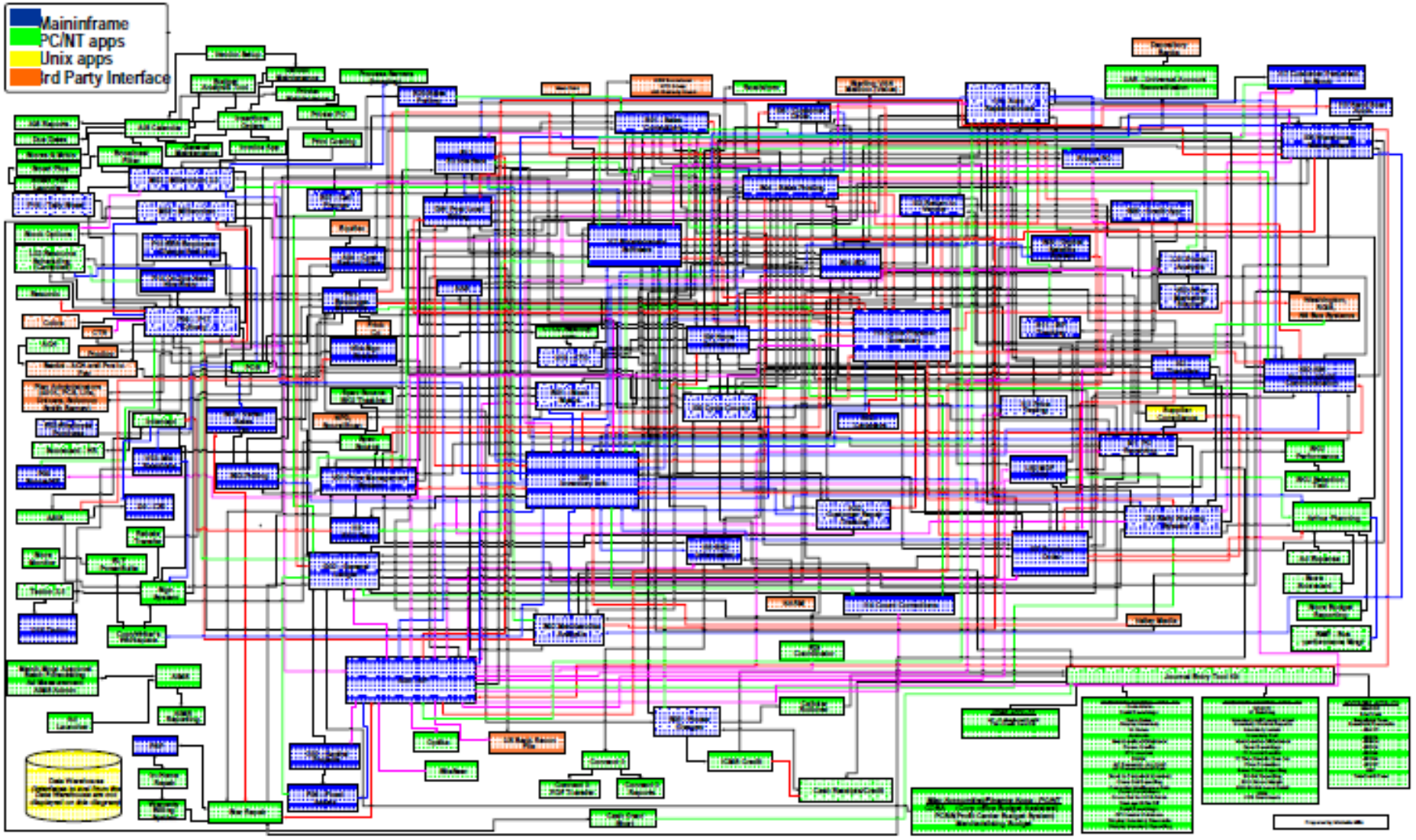
Microservices

- Jedna usługa = jeden komponent
- Komponenty niezależne
- Komponenty replikowalne i dystrybuowalne
- Komponenty o małym rozmiarze

SOA

- Wiele usług z jednego monolitycznego systemu
- Usługi zależne – jeden system
- Niemożność replikacji i dystrybucji usług SOA
- Za usługami zwykle monolityczny system

Microservices ?



Wirtualizacja (hardware virtualization)

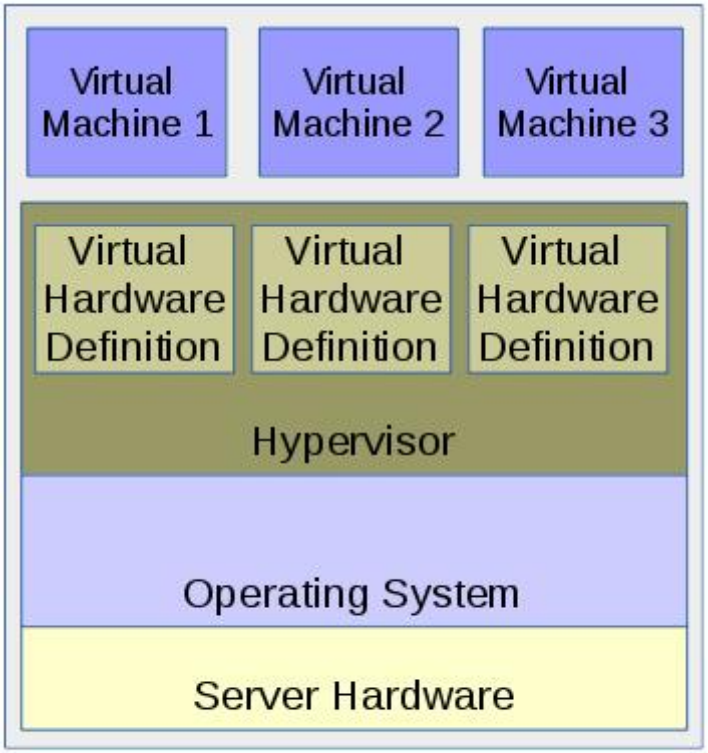
Uruchomienie **wielu systemów wirtualnych** na jednym serwerze sprzętowym, z przydziałem procesorów, pamięci i innych zasobów.

Zalety:

- **Pełna kontrola** i odwzorowanie.

Wady:

- Trudne zarządzanie, przenoszenie, modyfikacja - **obrazy wielkie.**



Hardware virtualization

Kontenery (OS virtualization)
Application delivery

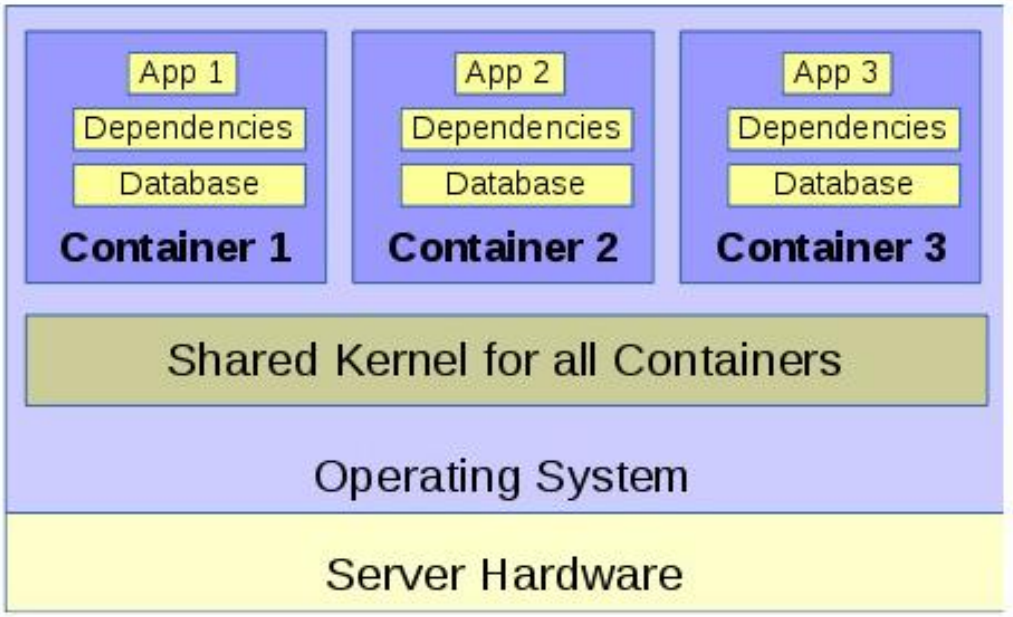
Uruchomienie **wielu systemów kontenerowych** na jednym serwerze, z możliwością przydziału procesorów, pamięci i innych zasobów.

Zalety:

- **Łatwość użycia**, stosunkowo mały rozmiar.

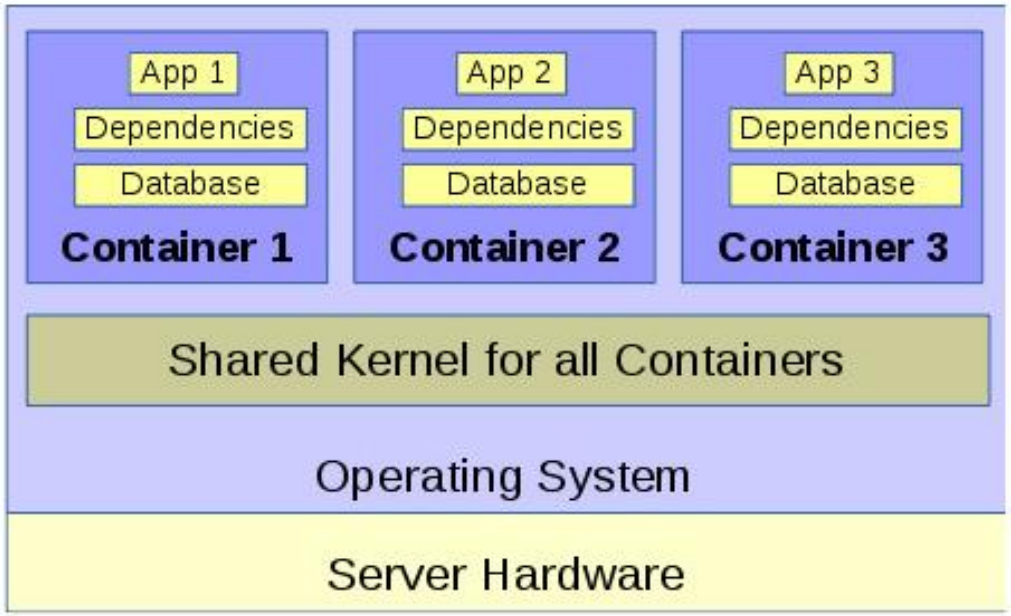
Wady:

- **Brak pełnej kontroli** (kernel wspólny).

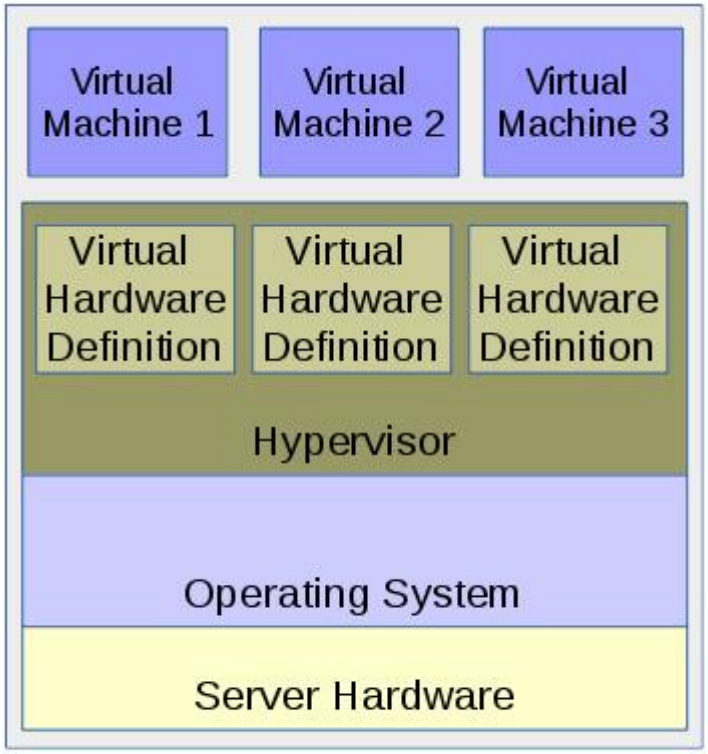


OS level virtualization

Kontenery a pełna wirtualizacja



OS level virtualization



Hardware virtualization

Platforma aplikacji

- Platforma umożliwiająca uruchomienie wielu komponentów aplikacyjnych i innych elementów systemu **bez narzutu VM/kontenerów**
- Serwery aplikacji – **i dlaczego wyginęły**
- POJO i Spring, a także inne języki
- Co dalej?

Orkiestracja komponentów systemu

Zarządzanie deploymentem poszczególnych komponentów w uporządkowany sposób:

- Aplikacji
- Infrastruktury

Monitoring i optymalizacja działania

- Security
- Logowanie i debugging

Szczególnie istotne w Chmurze!

DevOps

Metodyka zespolenia rozwoju (development) i eksploatacji (operations) oraz zapewnienia jakości (quality assurance), mająca na celu jak najbardziej **efektywne dostarczanie nowych wersji oprogramowania.**

- Jedno z najbardziej wypaczonych podejść w świecie IT
- **Jeden zespół**
- Właściwe rozumienie celów

Bazy danych SQL vs NoSQL

- **Rosnące znaczenie NoSQL**, idealne dla przetwarzania rozproszonego dużych ilości danych
- **NoSQL naturalny dla Cloud Computing**
- Konieczność wykorzystania baz relacyjnych, transakcje
- **Rozwiązania hybrydowe** w zakresie przechowywania danych

Cloud Computing – rodzaje i modele

1. Podstawowe pojęcia
2. **Co to jest Cloud Computing**
3. **Rodzaje i modele Cloud Computing**
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Co to jest „Chmura” i jej typy

Chmura – Cloud Computing - **możliwość wykorzystania infrastruktury** (serwery, dyski, sieć, inne) znajdującej się w serwerowni operatora usług chmurowych.

Główne rodzaje to:

- **IaaS** – Infrastructure as a Service
- **CaaS** – Container as a Service
- **PaaS** – Platform as a Service
- **PaaS serverless** (FaaS)
- **SaaS** – Software as a Service

IaaS – Infrastructure as a Service

Udostępnienie **wybranych elementów infrastruktury IT** w modelu cloud z wykorzystaniem zwirtualizowanych zasobów.

Typowe komponenty udostępniane w tym modelu:

- **Serwery wirtualne** (VM/VPS)
- Pamięć masowa (dyski)
- Elementy sieciowe (load balancing, routing)

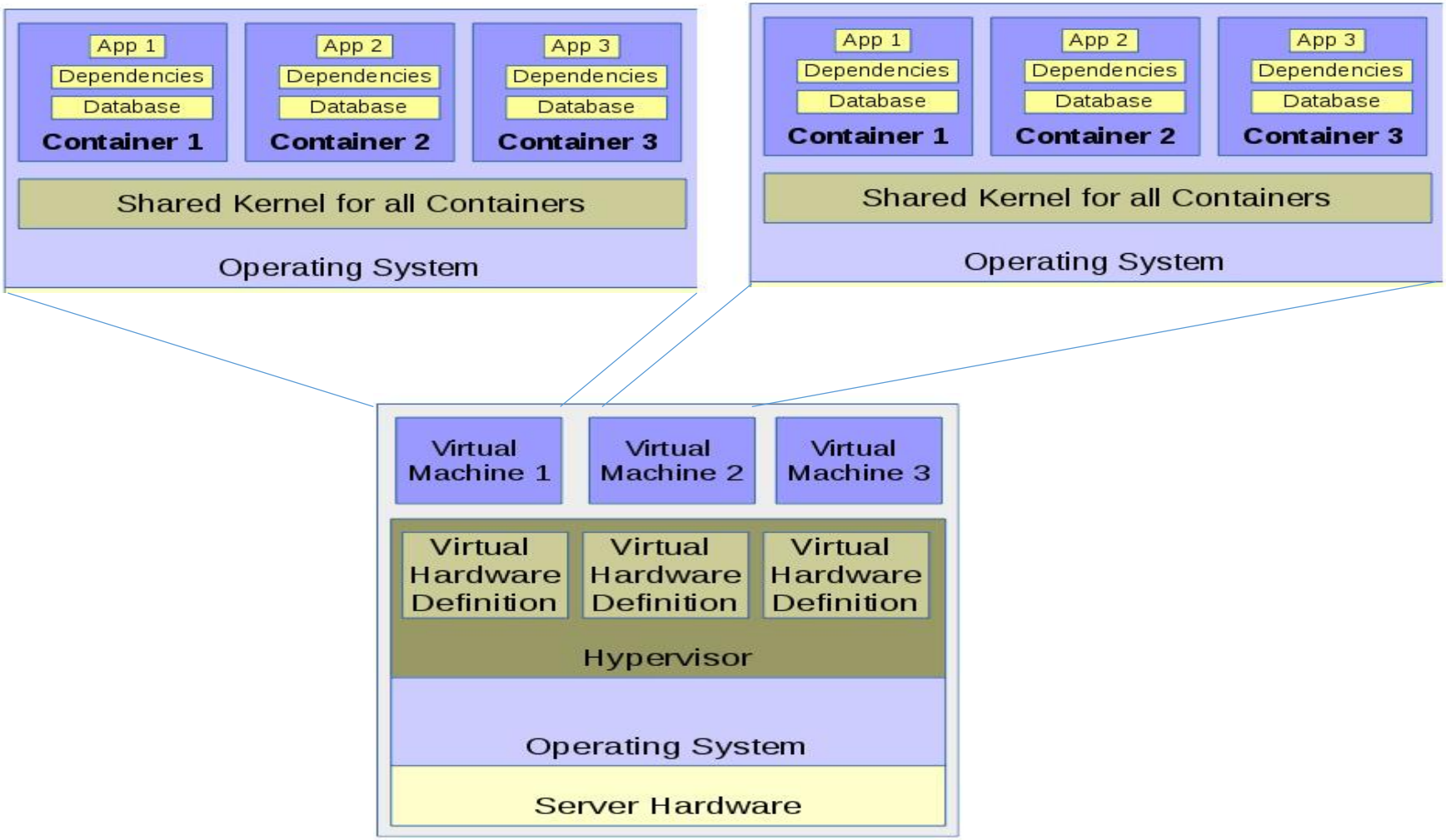
CaaS – Container as a Service

CaaS – Container as a Service – operator usług CaaS udostępnia platformę umożliwiającą **uruchamianie kontenerów** według zadanych parametrów.

Nie kupujemy poszczególnych usług typu serwery, dyski lecz usługę platformy dla kontenerów, a elementy infrastruktury obsługuje dostawca platformy.

Everybody loves DOCKER!

CaaS – wady



Hardware virtualization

PaaS – Platform as a Service

PaaS – Platform as a Service – operator usług PaaS udostępnia platformę **umożliwiającą uruchamianie aplikacji** według zadanych parametrów.

Dzięki temu nie dokonujemy zakupu poszczególnych usług typu serwery, dyski jak w modelu IaaS, **kupujemy usługę platformy dla aplikacji**, a elementy infrastruktury obsługuje dostawca platformy.

PaaS serverless (FaaS)

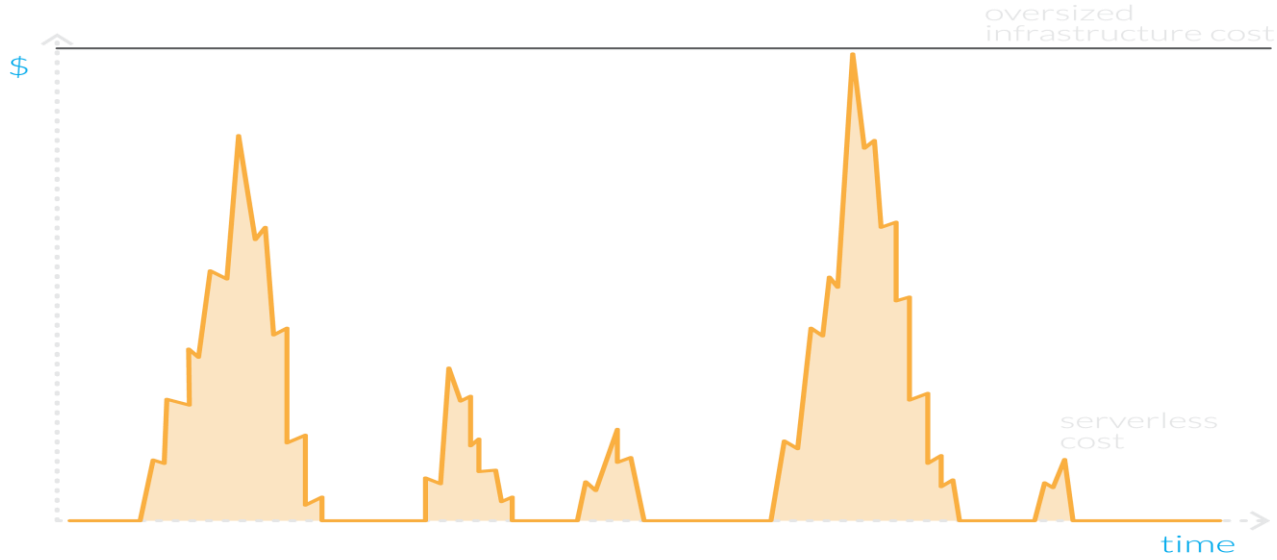
Usługa PaaS może być udostępniana również w wersji serverless, czyli takiej w której **opłata jest ponoszona za liczbę wywołań** danej aplikacji (lub jej funkcji).

Dzięki temu uzyskuje się **największą efektywność kosztową**, bo płaci się za faktyczne wykorzystanie systemu - brak opłat w przypadku braku korzystania z systemu.

PaaS serverless - zalety

Takie jak w przypadku PaaS plus dodatkowo:

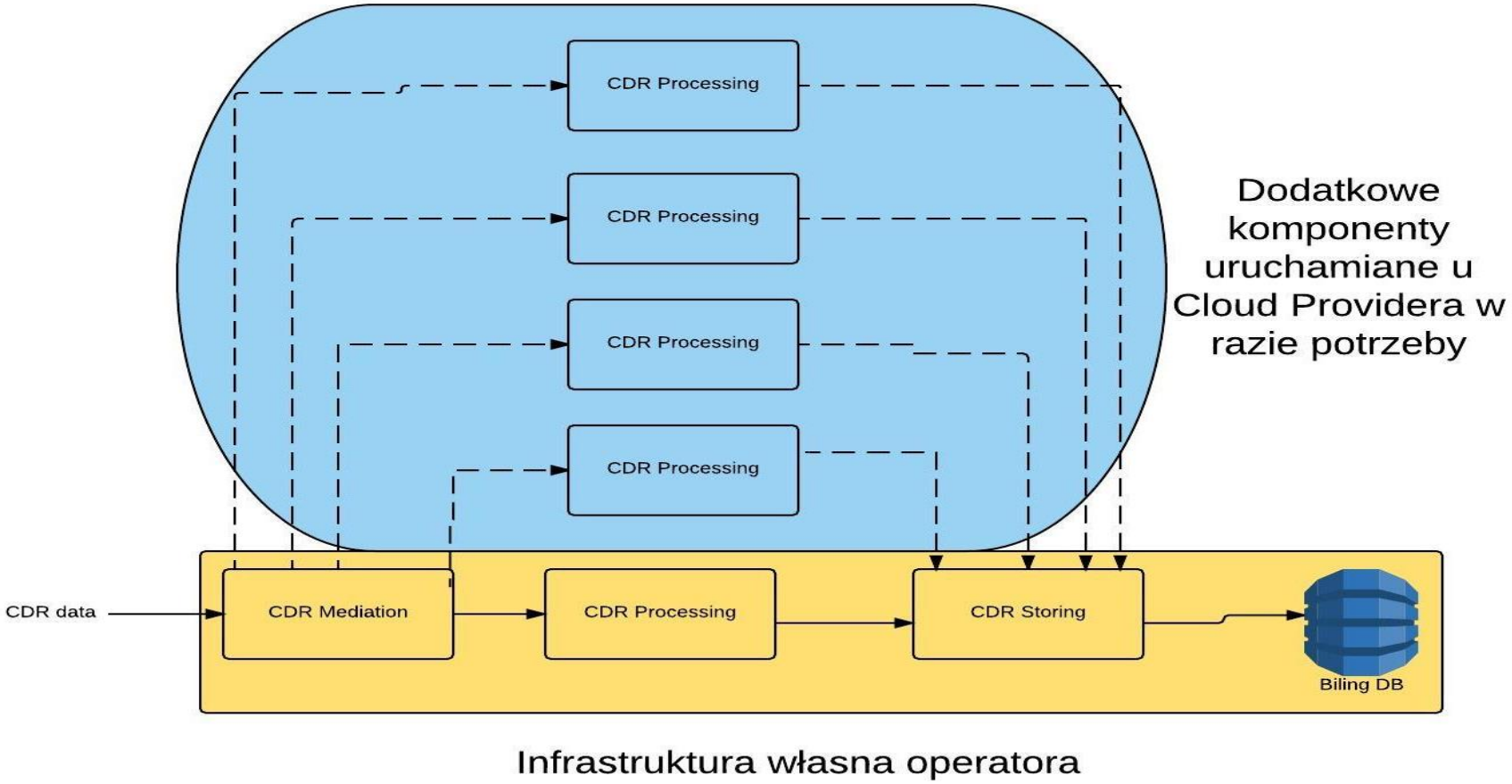
- Opłaty **tylko za faktyczne wykorzystanie systemu**, brak opłat w przypadku nie korzystania z systemu.
- **Łatwiejsze skalowanie** do wielu instancji – konstrukcja systemu wymusza przystosowanie do skalowanie.



PaaS serverless - wady

- Ograniczenia platformy.
- **Trudny development** i deployment.
- **Trudne zarządzanie** i monitorowanie.
- Nowa usługa na rynku – niedopracowana, **tylko jedna komercyjna implementacja.**

Serverless – Biling use case



SaaS – co to jest

Software as a Service – **udostępnienie aplikacji jako całej w chmurze**, dostęp przez Internet.

- To **nie Cloud Computing**, a raczej model płatności za aplikacje.
- Właściwe zrozumienie zasadności wykorzystania.

Cloud Computing – orkiestracja – jak to działa

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. **Cloud Computing - orkiestracja**
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Orkiestracja w chmurze - rozwiązania

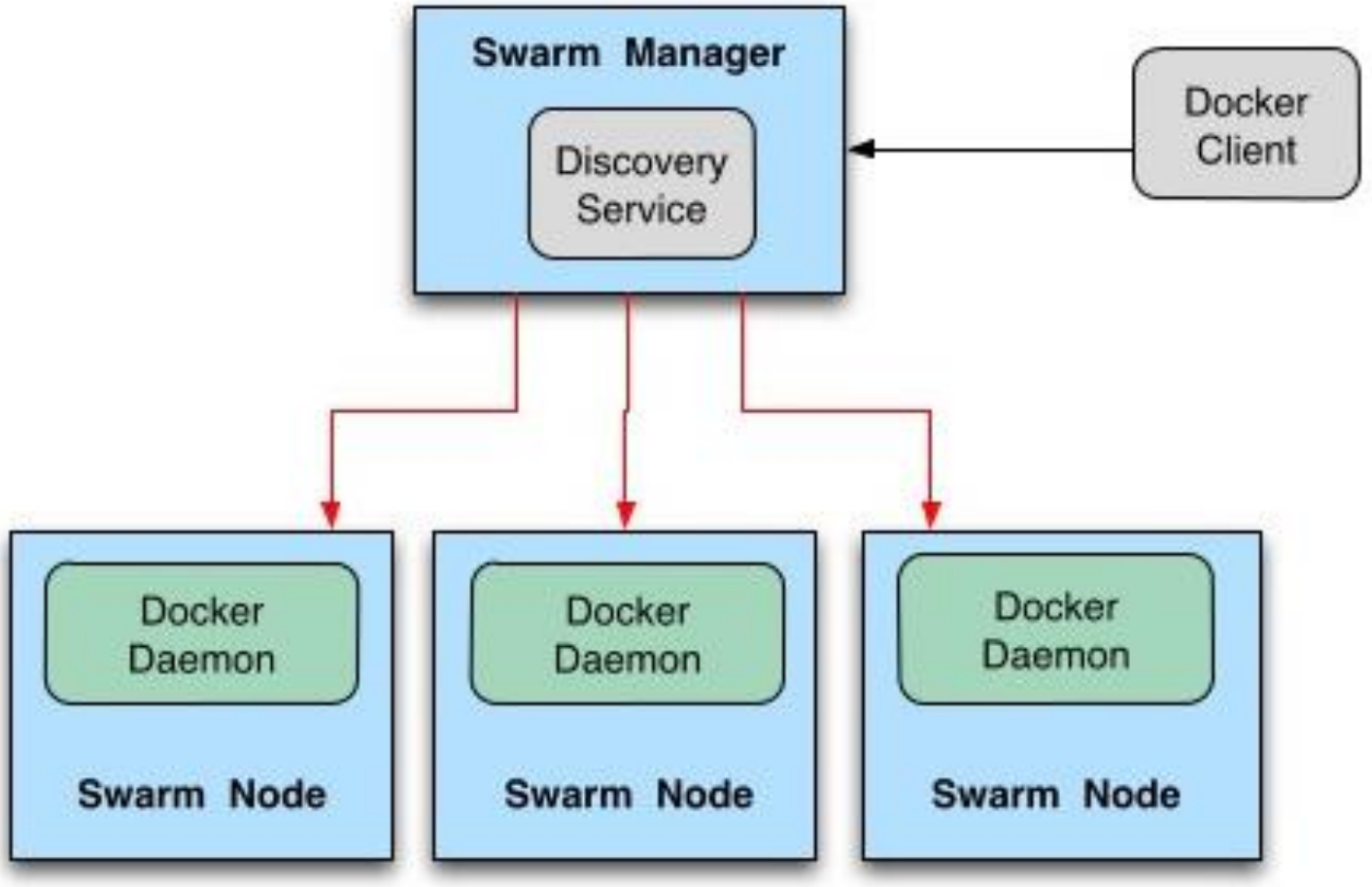
- Open Stack
- Kubernetes
- Docker Swarm
- TerraForm
- Tosca/Cloudify
- Heat

Docker Swarm – klaster kontenerów

Natywny dla Docker system obsługi orkiestracji

- Zarządzanie pulą hostów z kontenerami z jednego kontrolera.
- Zarządzanie zasadami umieszczania kontenerów na hostach.

Swarm Architecture



Docker Swarm – zalety i wady

Zalety

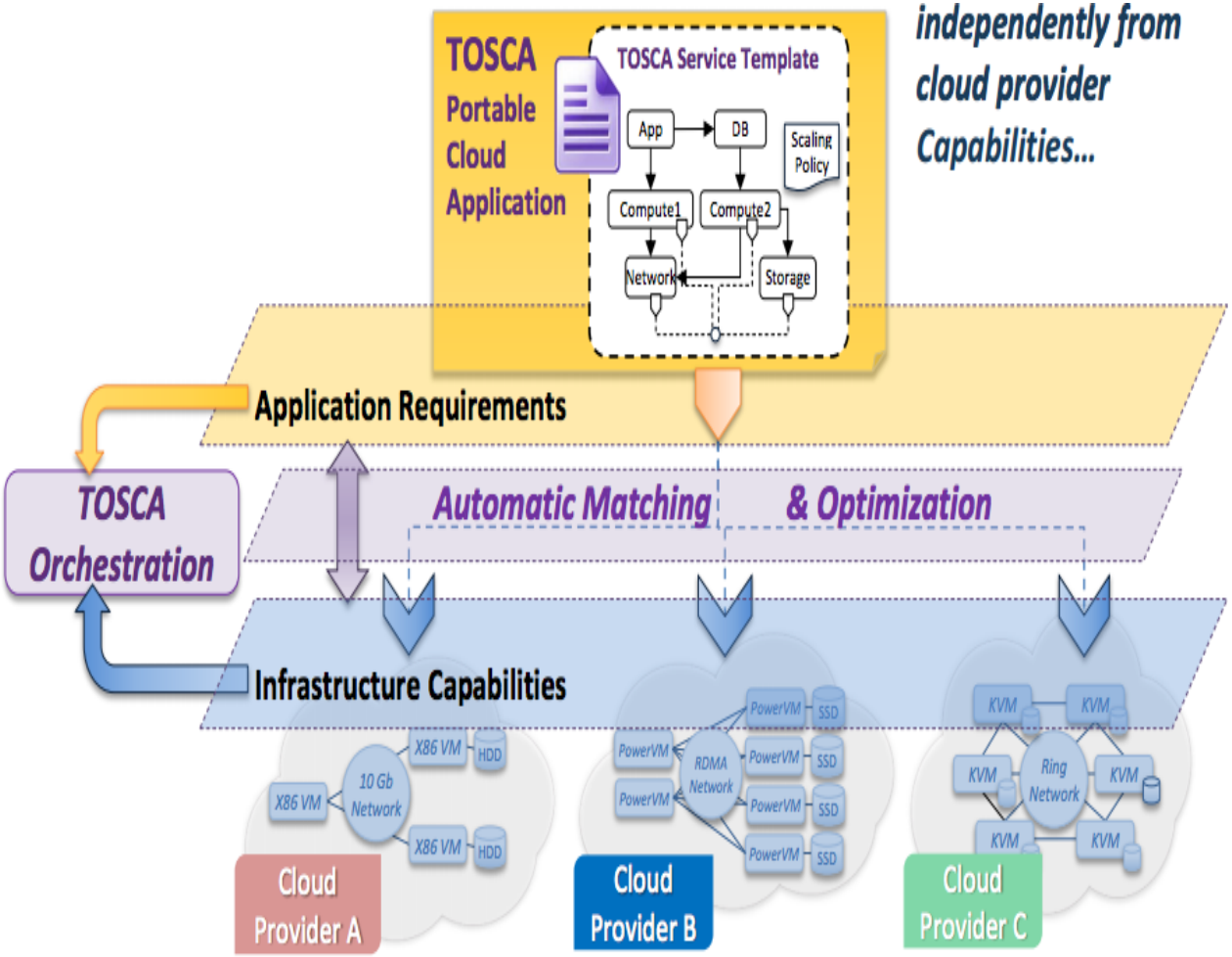
- Łatwość użycia z Docker,
- Możliwość elastycznego zarządzania lokalizacją kontenerów.

Wady

- Pierwsza wersja,
- Ręczna konfiguracja wielu instancji,
- Brak procesu deploymentu
- Brak monitoringu, automatycznego skalowania itd.
- Brak obsługi wielu Cloud Providers.

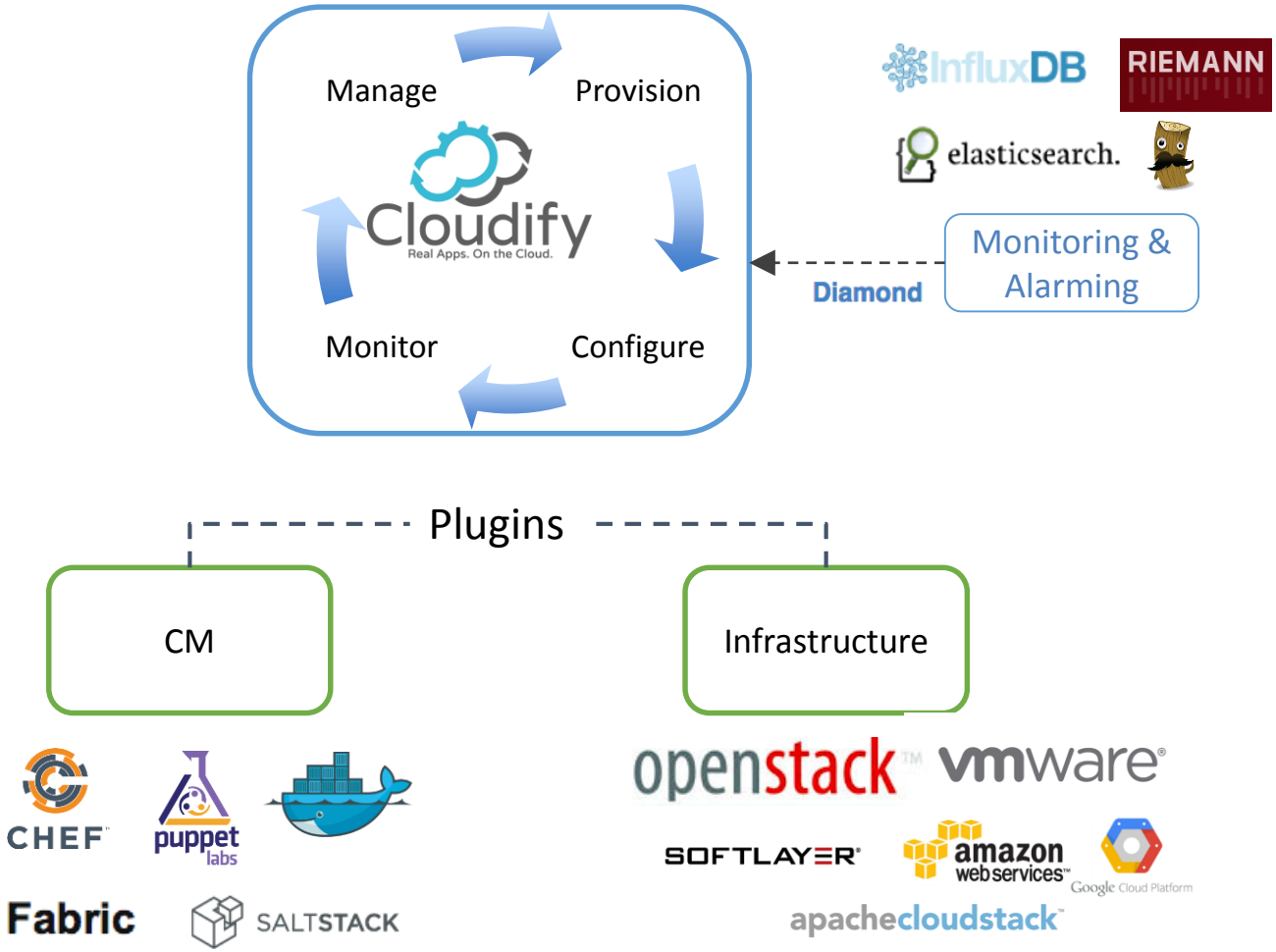
TOSCA

Topology and Orchestration Specification for Cloud Applications



Kompleksowe definiowanie architektury cloud dla aplikacji i komponentów, z relacjami, zależnościami, wymaganiami i możliwościami poprzez DSL

Cloudify – wersja open source TOSCA



Podsumowanie Cloudify/TOSCA

Zalety

- Agnostyczna względem infrastruktury i frameworków
- Pełen cykl życia architektury
- Obsługuje infrastrukturę i aplikacje
- Pełny zakres orkiestracji:
  - Monitoring
  - Proces deploymentu
  - Polityki deploymentu
  - Logowanie

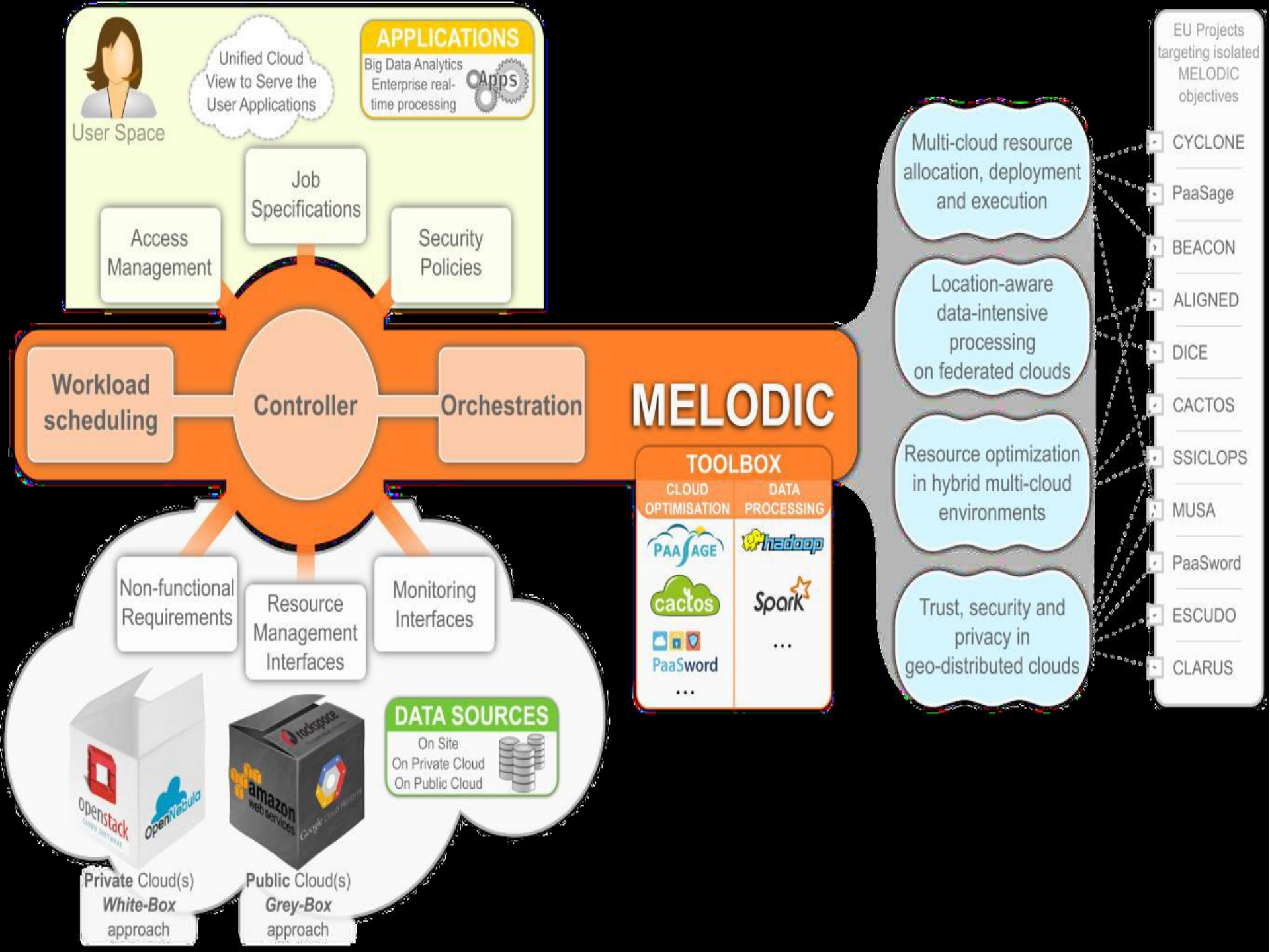
Wady

- W trakcie tworzenia
- Brak dynamicznego zarządzania skalowaniem i HA
- Ograniczony zestaw narzędzi

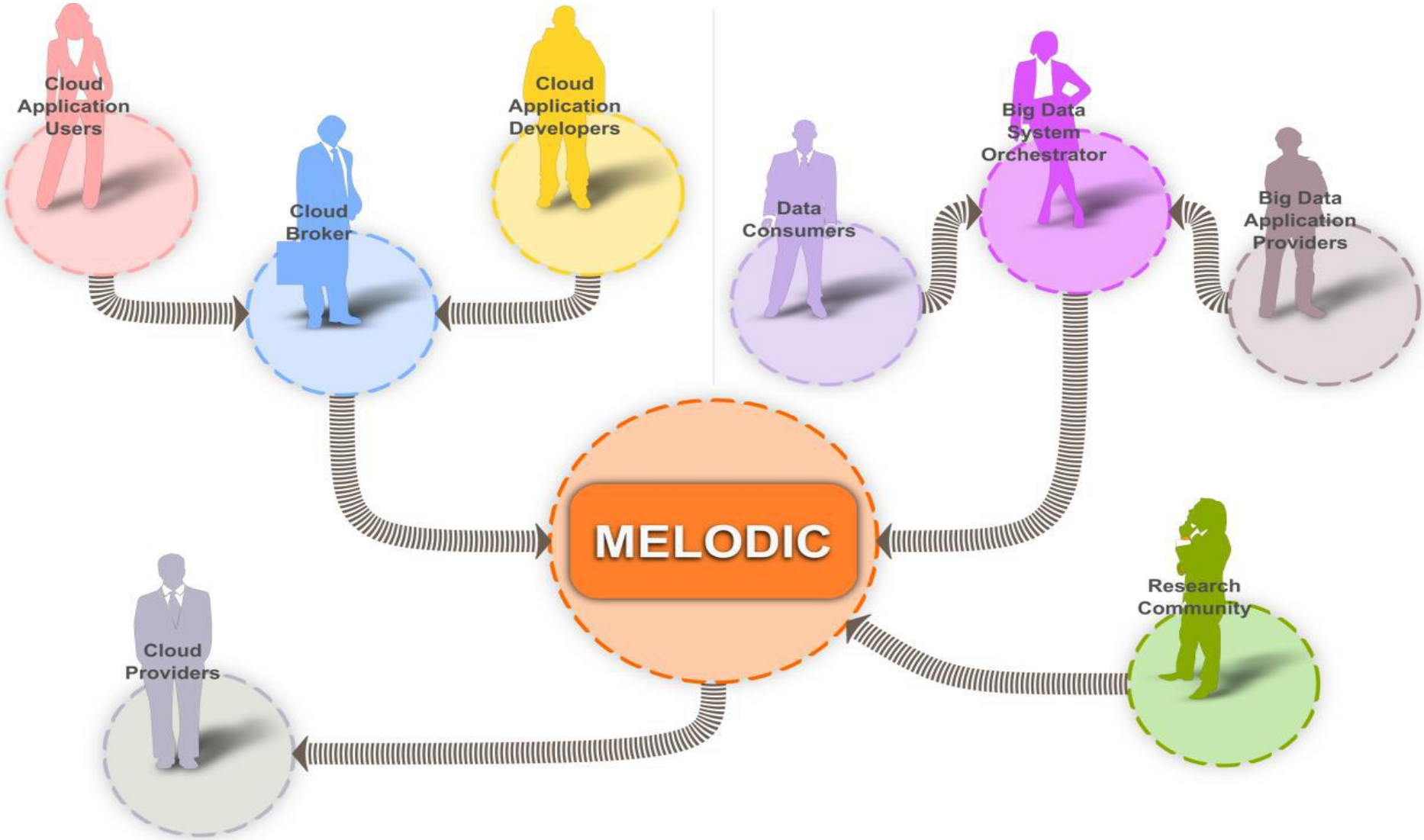
MELODIC

Multicloud **E**xecution-ware for **L**arge-scale **O**ptimized **D**ata **I**ntensive **C**omputing

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. **MELODIC**
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania


User Space
Unified Cloud View to Serve the User Applications
APPLICATIONS
Big Data Analytics
Enterprise real-time processing
Apps
Job Specifications
Access Management
Security Policies
Workload scheduling
Controller
Orchestration
MELODIC
TOOLBOX
CLOUD OPTIMISATION
DATA PROCESSING
PAASAGE
hadoop
cactos
Spark
PaaSword
...
...
Non-functional Requirements
Resource Management Interfaces
Monitoring Interfaces
DATA SOURCES
On Site
On Private Cloud
On Public Cloud
openstack
OpenNebula
rackspace
amazon web services
Google Cloud Platform
Private Cloud(s)
White-Box
approach
Public Cloud(s)
Grey-Box
approach
Multi-cloud resource allocation, deployment and execution
Location-aware data-intensive processing on federated clouds
Resource optimization in hybrid multi-cloud environments
Trust, security and privacy in geo-distributed clouds
EU Projects targeting isolated MELODIC objectives
CYCLONE
PaaSage
BEACON
ALIGNED
DICE
CACTOS
SSICLOPS
MUSA
PaaSword
ESCUDO
CLARUS

MELODIC – dla kogo?



New application

Legacy application

Speculative profiler

Community expertise

PaaSage Platform

CAMEL

- Provisioning and deployment requirements
- Scalability rules
- Service-level objectives
- Provider models

Design time optimisation loop

Metadata

Speculative profiler

Intelligent reasoner

Extra functional adaption

Execution optimisation loop

Metadata collection

Execution control

Execution monitoring

Platform specific mapping

Execution environments

MELODIC - cechy

1. Jedna uniwersalna platforma instalacji, uruchomienia i zarządzania aplikacjami w chmurze.

2. Możliwość migracji aplikacji pomiędzy dostawcami usług Cloud.

3. Możliwość wyboru najbardziej optymalnej opcji uruchomienia aplikacji na bazie charakterystyki aplikacji.

4. Bezpieczeństwo - uruchomienie aplikacji u więcej niż jednego dostawcy.

5. Brak Vendor – lock.

MELODIC – jak skorzystać

1. Instalacja aplikacji na platformie Melodic.
2. Wybór dostawców usług u których ma być zainstalowana aplikacja.
3. Uruchomienie i działanie aplikacji.
4. Analiza działania aplikacji.
5. Optymalizacja kosztowa i wydajnościowa aplikacji na bazie jej charakterystyki.
6. Przeniesienie do najbardziej optymalnego dostawcy usług Cloud.

MELODIC – co może dać Asseco?

1. Zunifikowaną platformę do deploymentu aplikacji – jednolite instrukcje wdrożeniowe
2. Możliwość multcloud i hybrid cloud – unikalna oferta na rynku
3. Optymalizacja (w tym kosztowa) infrastruktury i aplikacji – szczególnie istotna w chmurach

MELODIC – status i rola 7bulls.com

1. Projekt wystartował 1.12.2016, czas trwania 3 lata, pierwsza iteracja po 18 miesiącach.
2. Cały system będzie dostępny jako Open Source.
3. **7bulls.com odpowiedzialny za Quality Assurance i Integration.**
4. Poszukiwani partnerzy:
   - Cloud Services providers
   - Klienci/użytkownicy – early adopters

Praktyczne omówienie wybranych rozwiązań

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. **Praktyczne omówienie wybranych rozwiązań**
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Amazon Webservices

Obecnie najbardziej kompletna platforma usług Cloud różnego rodzaju

Zalety:

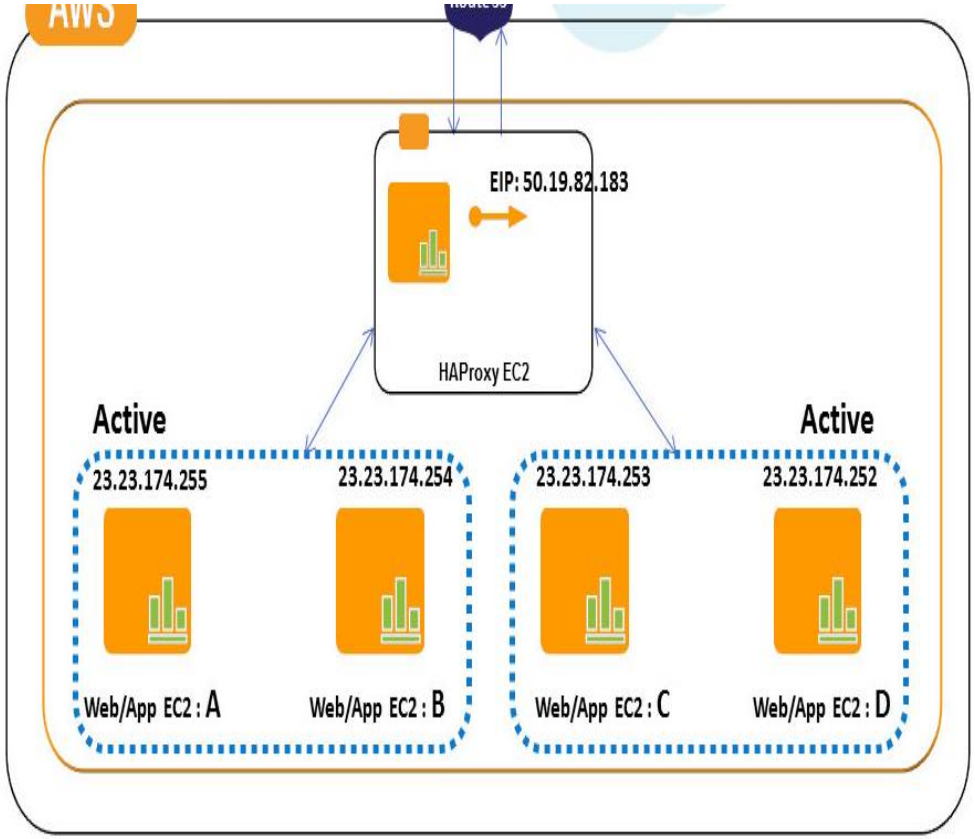
- Duży wybór komponentów,
- Jedyna platforma serverless,
- Rozwiązania dla monitoringu, logowania, deploymentu,
- Zintegrowany system uprawnień,
- Autoscaling różnego typu,
- Opłata za czas (minutowo) i możliwość zakupu z góry,
- Wiele datacenter.

Wady:

- Część rozwiązań nieoptymalnych,
- Rozwiązania specyficzne dla AWS - silny vendor lock,
- Uwaga: Ceny.

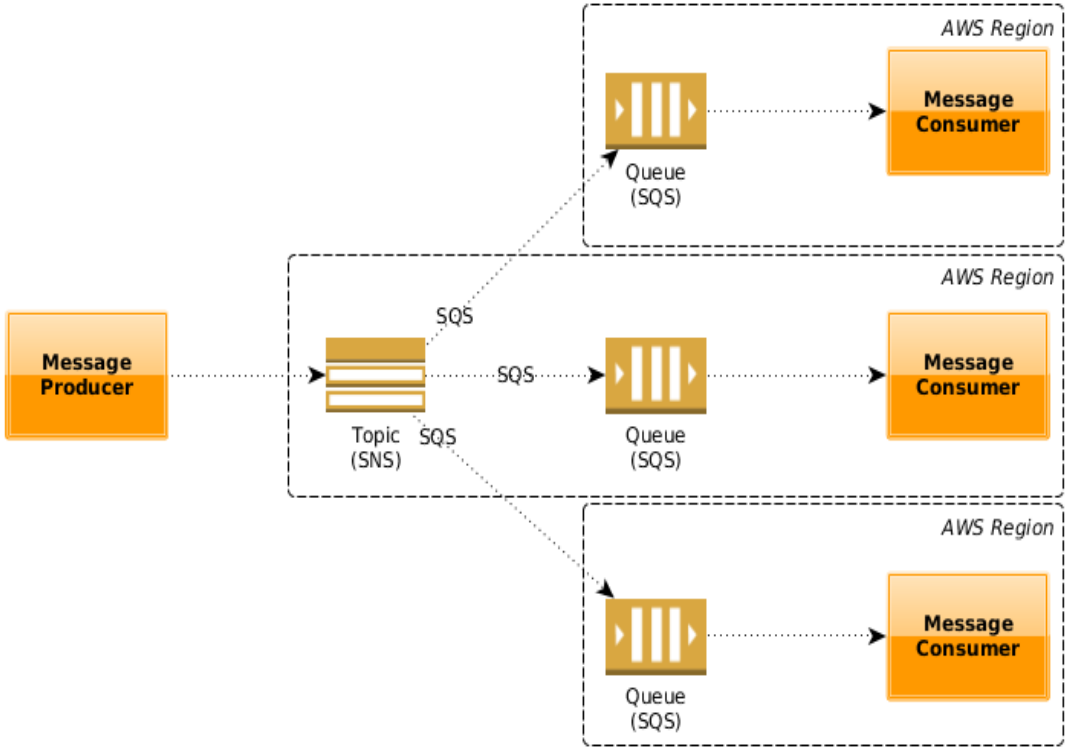
EC2 - serwer wirtualny

- IaaS na KVM.
- Tworzenie obrazów i backup.
- Wiele dostępnych typów maszyn, w tym GPU.
- Autoscaling.
- Opłata za czas (minutowo) i możliwość zakupu z góry.
- Szybkość rekonfiguracji.

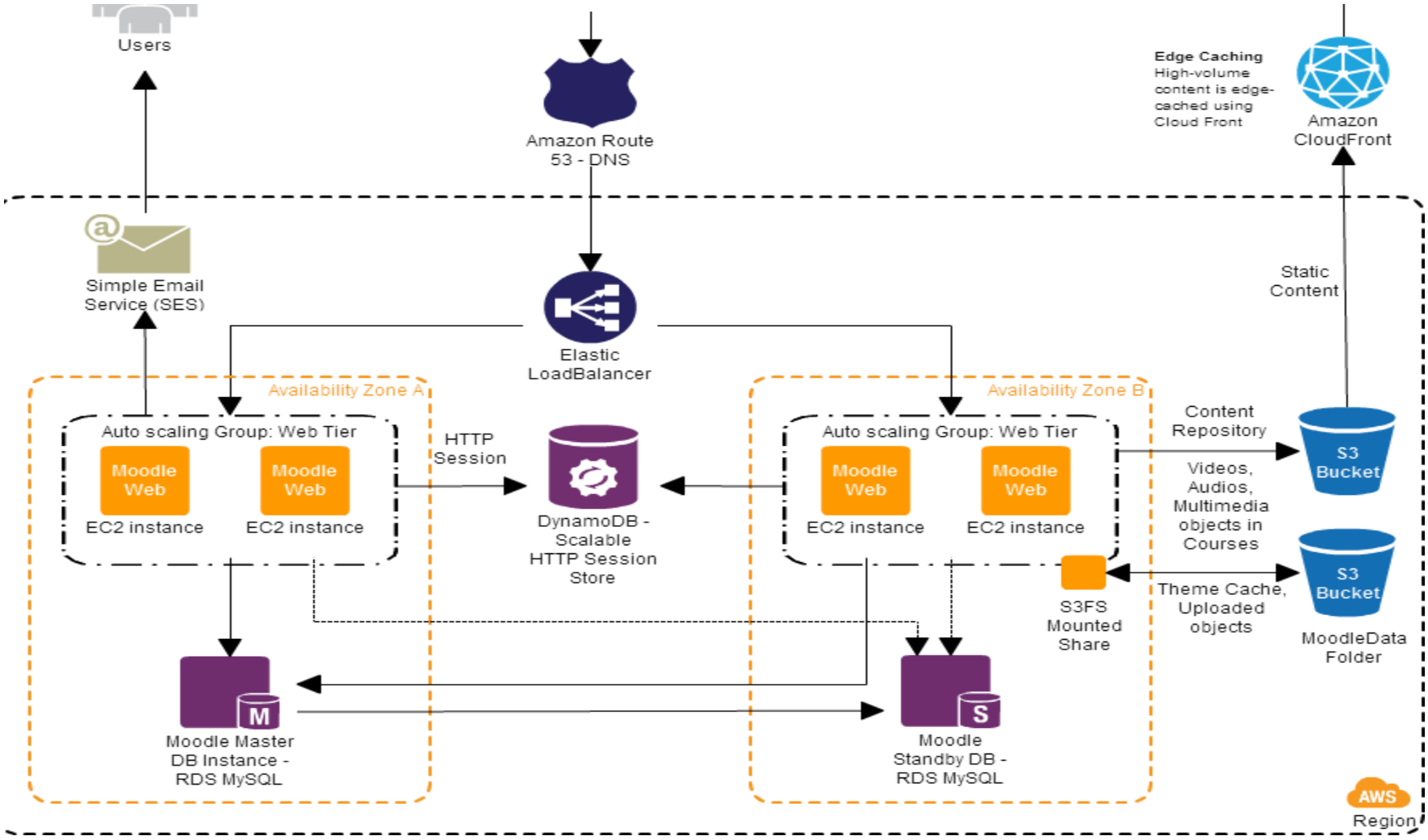


SQS – system kolejek

- Prosta kolejka p2p
- Możliwa persystencja na platformie
- API do różnych języków
- Vendor specifc



Autoscaling IaaS



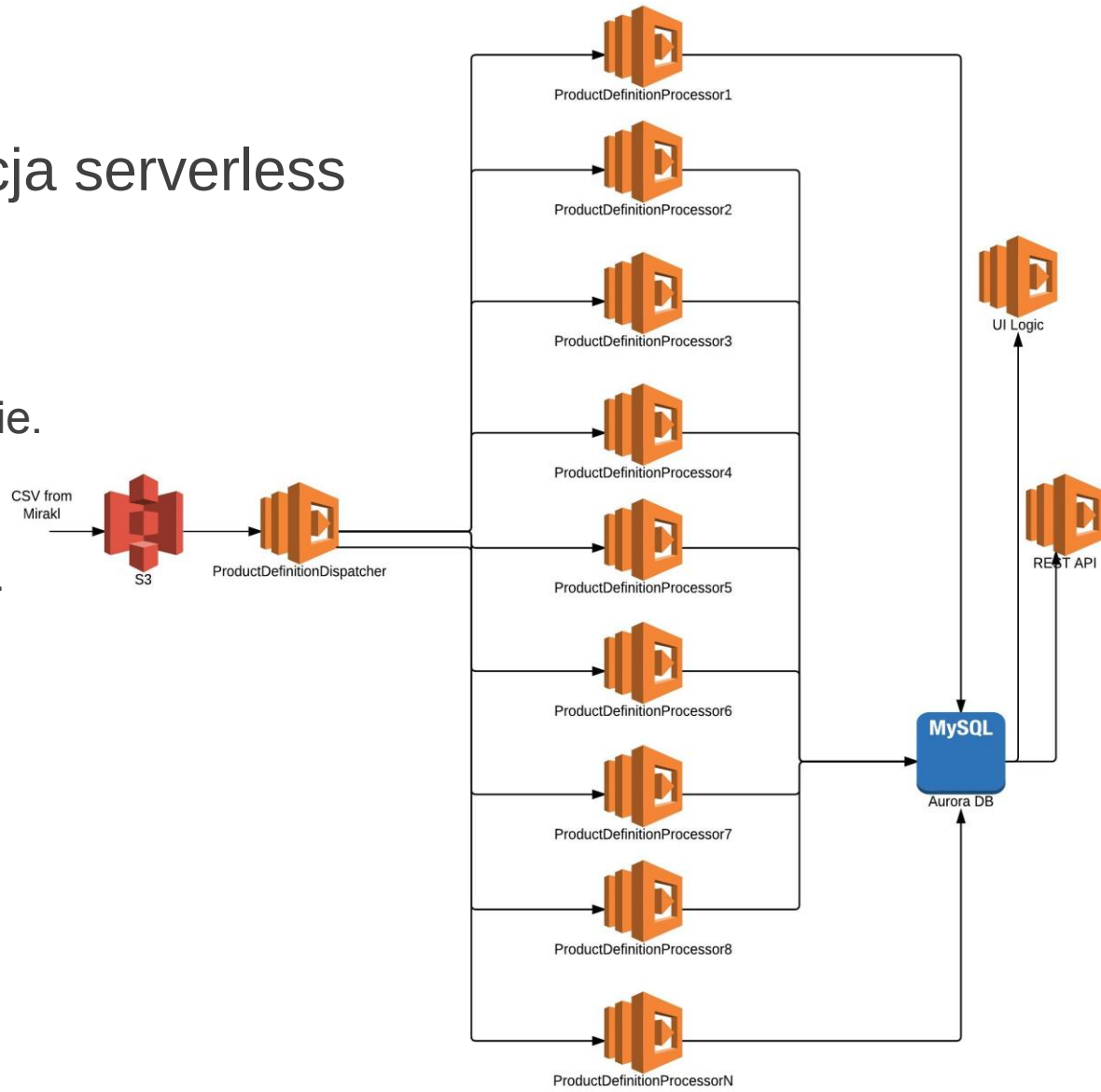
RDS – bazy danych w modelu zarządzanym

- Bazy danych w modelu zarządzanym – powyżej OS
- Oracle, MS SQL Server, Postgress i inne
- Backup, snapshot, HA
- Dynamiczne zwiększanie wydajności
- Vendor specifc

7bulls.com

Lambda – implementacja serverless

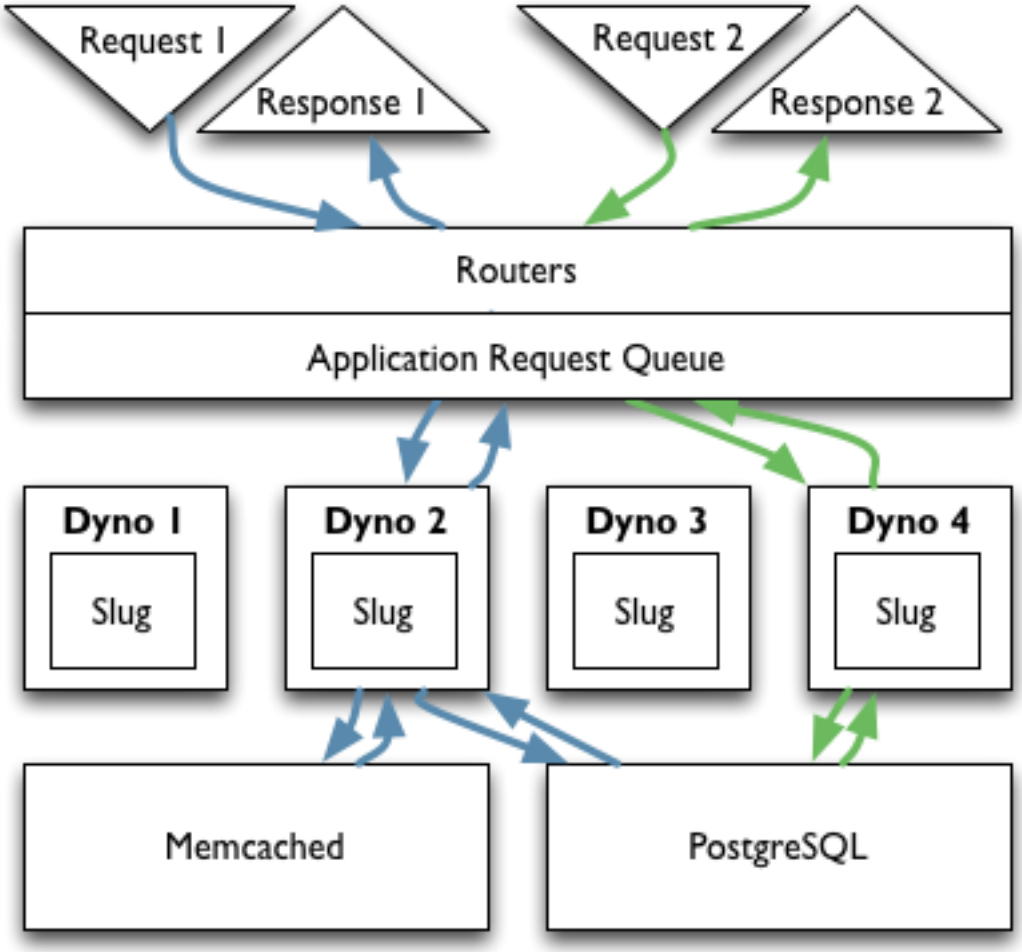
- Komponenty istniejące tylko w czasie wykonania.
- Płatność tylko za realne użycie.
- Różne języki programowania.
- Łatwość skalowania.
- Ograniczony czas życia.
- Vendor specific.



Legenda:
S3 - bucket do którego uploadowane via ftp są csv z definicjami produktów od merchantów
ProductDefinitionDispacher - lambda uruchamiana po upload pliku do S3 która uruchamia przetwarzanie dla zawartości pliku
ProductDefinitionProcesor - lambdy przetwarzające wiersze z pliku CSV, dla każdego wiersza oddzielna lambda przetwarzająca od początku do końca daną definicję produktu
Aurora DB - baza danych z definicjami produktów od merchantów, definicjami referencyjnymi i innymi danymi systemu. Dla definicji produktów zawiera tylko atrybuty deduplikacyjne
UI Logic - lambda zawierająca logikę dla UI
REST API - lambda z interfejsami REST dla Hybris

Dynos – PaaS Heroku

- Dynos – smart containers.
- Najwygodniejsza implementacja PaaS (która de facto jest CaaS).
- Efektywne skalowanie.
- Skuteczne zarządzanie.
- Różne języki programowania.
- Bardzo wydajny development.
- Vendor specific.



Cloud Quality

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Cloud ESB - Mulesoft

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Cloud Computing – jak wykorzystać racjonalnie

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. Cloud Computing – Trendy i Wyzwania

Jak wybrać właściwie

- Elastyczność
- Skalowanie
- Optymalizacja (wydajność i koszty)
- Architektura

?

Elastyczność i skalowalność

- Kontenery
- Orkiestracja
- Optymalizacja deployment
- Skalowanie horyzontalne i wertykalne
- Brak vendor-lock!

W chmurze nie róbmy tak jak dotychczas!

Optymalizacja kosztów

- Płatność tylko za realny czas wykorzystania
- Skalowanie bez limitu i wtedy kiedy potrzeba
- Wybór najlepszego Cloud Providera dla danej aplikacji

To można tylko w chmurze!

Architektura Cloud Computing

- Maksymalne wykorzystanie możliwości Cloud Computing - Microservices
- Świadome zarządzanie architekturą pod kątem możliwości Cloud Computing
- Services load balancing and discovery
- Utylizacja zasobów
- Właściwa orkiestracja

Lepiej zrobić to dobrze od razu!

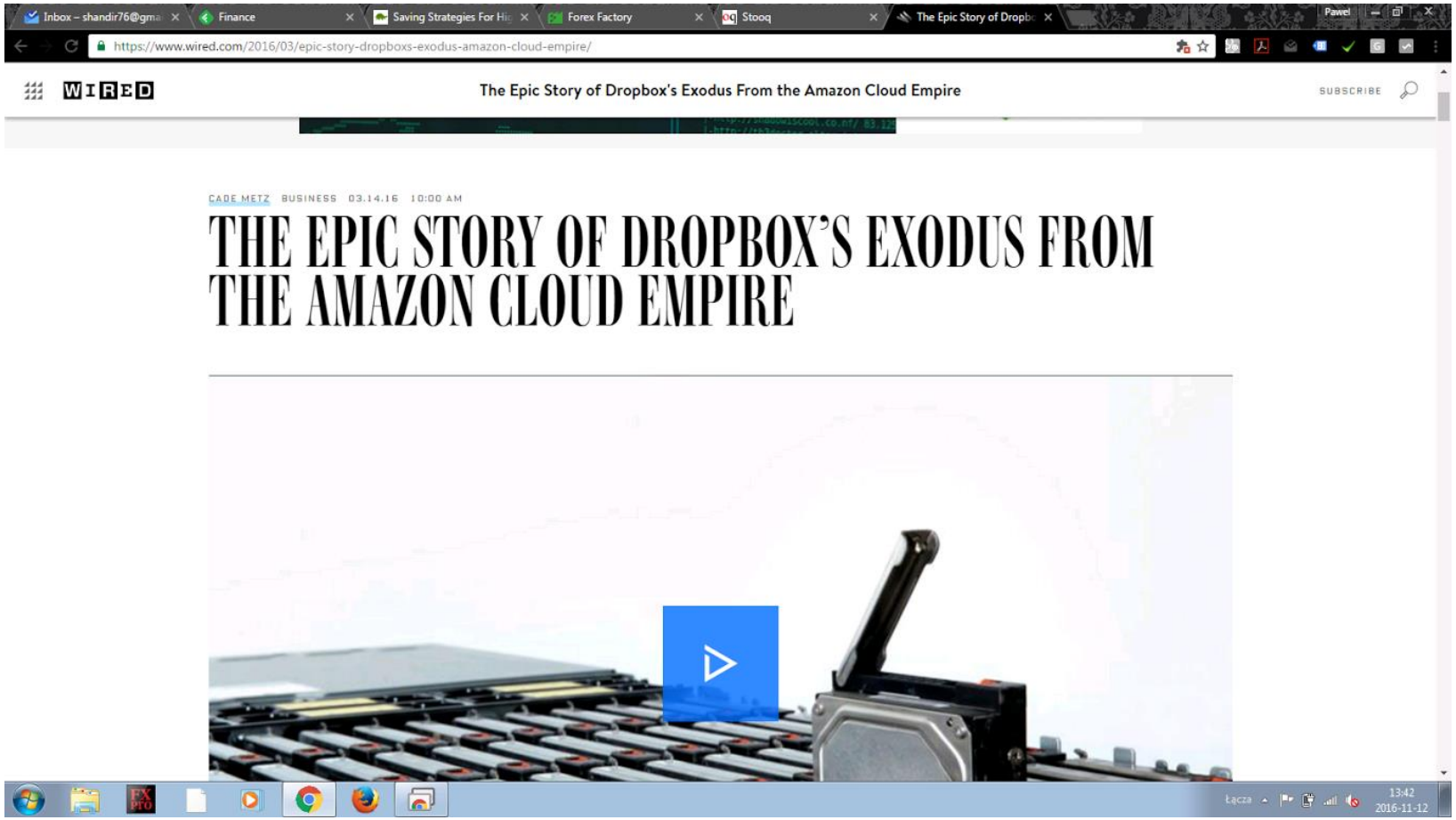
Jaka chmura dla kogo?
Małe organizacje

- Przede wszystkimi minimalizacja kosztów IT – infrastruktury i zatrudnienia ludzi
- Model **Cloud only** wart rozważenia
- SaaS i CaaS – minimalizacja kosztów infrastruktury IT i jej utrzymania
- Serverless – jako wsparcie do **przetwarzania dużych ilości danych** o nieliniowym rozkładzie obciążenia
- Minimum **dwóch różnych Cloud Providers**
- Unikanie vendor lock

Jaka chmura dla kogo?
Duże organizacje

- Kontenery
- Orkiestracja
- **Przejście na microservices** - docelowo
- Model Hybrid cloud
  - wykorzystanie **Cloud do szczytów ruchu**, przetwarzania cykicznego
  - podstawowe przetwarzanie własna infrastruktura, cloud jako backup/HA/disaster recovery
- CaaS/PaaS – jako platforma backupowa, do przejęcia części ruchu
- Serverless – jako wsparcie do **przetwarzania dużych ilości danych** o nieliniowym rozkładzie obciążenia
- **Unikanie vendor lock**

Jaka chmura dla kogo?
Duże organizacje



Cloud Computing
Trendy i Wyzwania

1. Podstawowe pojęcia
2. Co to jest Cloud Computing
3. Rodzaje i modele Cloud Computing
4. Cloud Computing - orkiestracja
5. MELODIC
6. Praktyczne omówienie wybranych rozwiązań
7. Cloud Computing – jak wykorzystać racjonalnie
8. **Cloud Computing – Trendy i Wyzwania**

Trendy Cloud Computing

- Kontenery
- Software defined everything (server, storage, network) – software defined architecture
- Multi-cloud, hybrid-cloud
- Orkiestracja
- Strumienie i przetwarzanie dużej ilości danych
- Lekkie aplikacje (Node.js, Pythoon, Groovy)

Cloud Computing - wyzwania

1. Zarządzanie i planowanie architektury i infrastruktury Cloud
2. Zapewnienie jakości
3. Bezpieczeństwo
4. SLA Ready – Common Reference Model

Cloud Computing - mity

1. Bezpieczeństwo
2. Dane (w szczególności poufne)
3. Vendor lock
4. Rozwiązania hybrydowe
5. Mitologiczny marketing

Cloud Computing - podsumowanie

1. Amazon Webservices
2. Multicloud - Melodic
3. Cloud Quality
4. Cloud Intergration
5. Architektura

Dziękujemy

7bulls.com

Cloud Computing - bezpieczeństwo

Tadeusz Reczyński
Paweł Skrzypek

2017.06.14

Zagrożenia w chmurach - co to znaczy?

Zagrożenia ?
Chmura ?

Spis treści

Bezpieczeństwo - zagrożenia IT

Model STRIDE

- Spoof of identity - fałszowanie tożsamości
- Tampering - zmiana oprogramowania
- (non) Repudiation - niezaprzeczalność operacji
- Information disclosure - ujawnienie danych
- Denial of service - odmowa usług
- Elevation of privilege - zwiększenie poziomu uprawnień

Ciągłość prowadzenia działalności

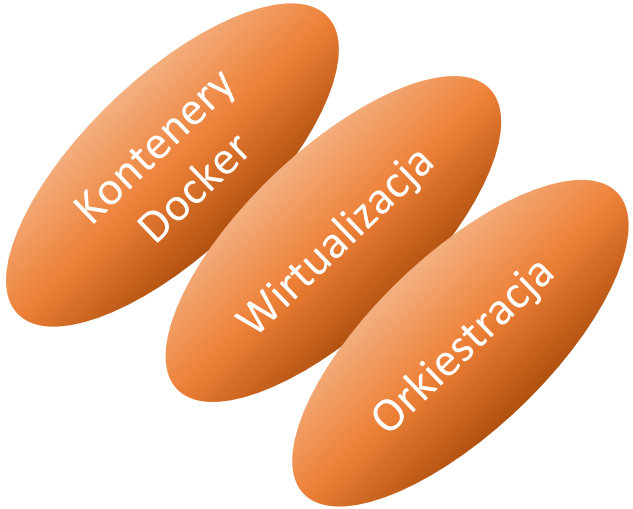
- Bezpieczeństwo fizyczne - zagrożenia ludzkie i naturalne
- Centrum zapasowe
- Możliwość zwiększania zasobów
- Backup i replikacja
- Vendor lock-in
- Koszt prowadzenia działalności IT

Co to jest „Chmura” i jej typy

Chmura – Cloud Computing - wykorzystanie infrastruktury w serwerowni operatora usług chmurowych.

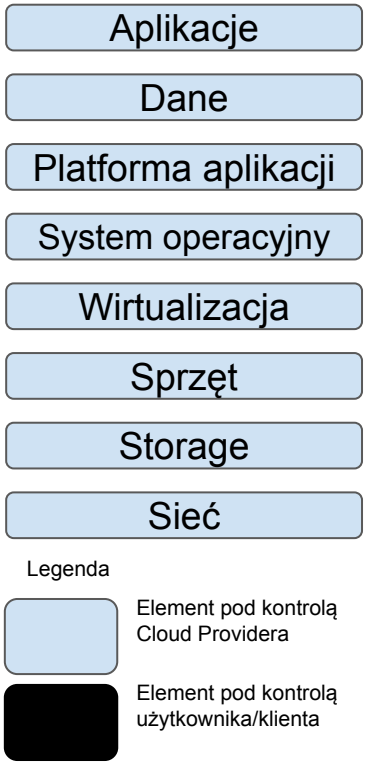
Główne rodzaje to:

- IaaS – Infrastructure as a Service
- CaaS – Container as a Service
- PaaS – Platform as a Service
- PaaS serverless (FaaS)
- SaaS – Software as a Service



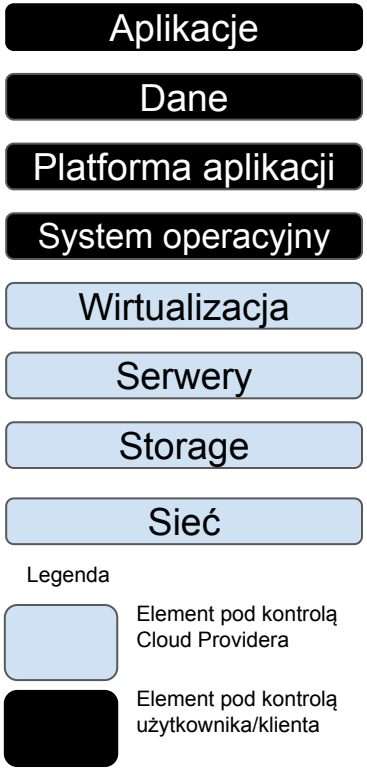
SaaS – co to jest?

Software as a Service – **udostępnienie aplikacji jako całej w chmurze**, dostęp przez Internet.



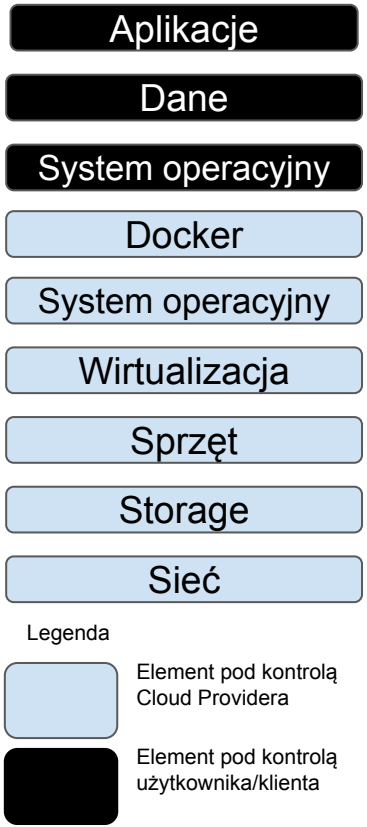
IaaS – Infrastructure as a Service

Udostępnienie **wybranych elementów infrastruktury IT** w modelu cloud z wykorzystaniem zwirtualizowanych zasobów.



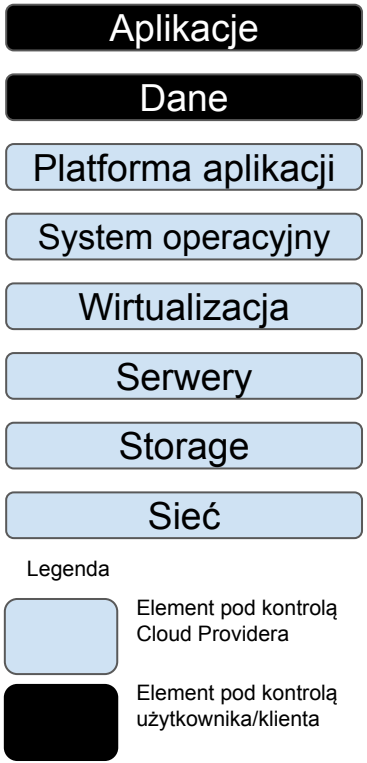
CaaS – Container as a Service

Udostępnienie platformy umożliwiającej **uruchamianie kontenerów** według zadanych parametrów.



PaaS – Platform as a Service

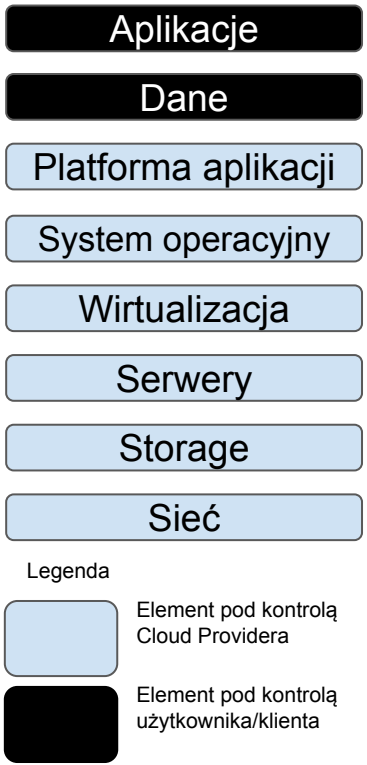
Udostępnienie platformy **umożliwiającej uruchamianie aplikacji** według zadanych parametrów.



7bulls.com

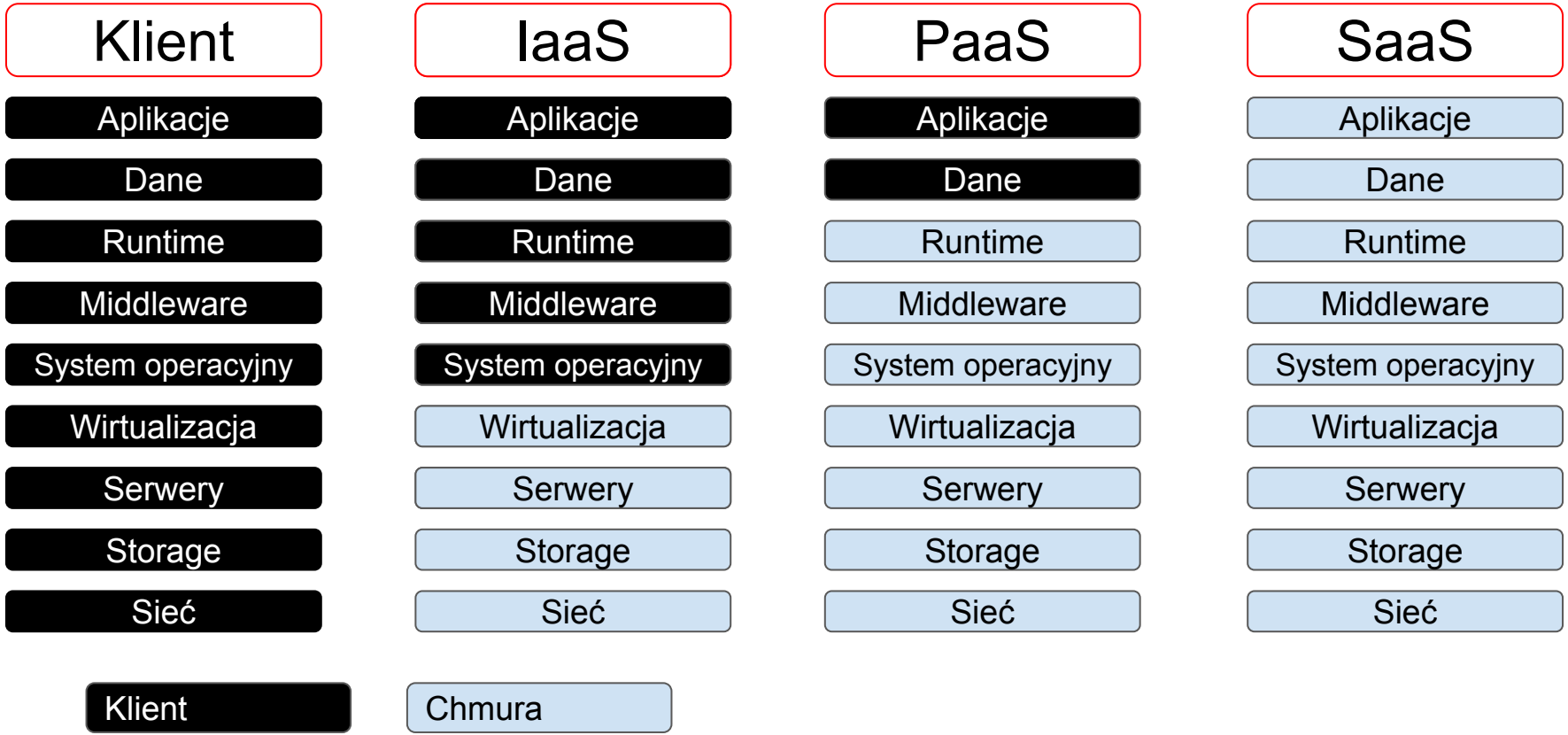
PaaS serverless (FaaS)

Usługa PaaS może być udostępniana również w wersji serverless, czyli takiej w której **opłata jest ponoszona za liczbę wywołań** danej aplikacji lub jej funkcji.



7bulls.com

Modele chmury obliczeniowej



Cloud Computing - cechy

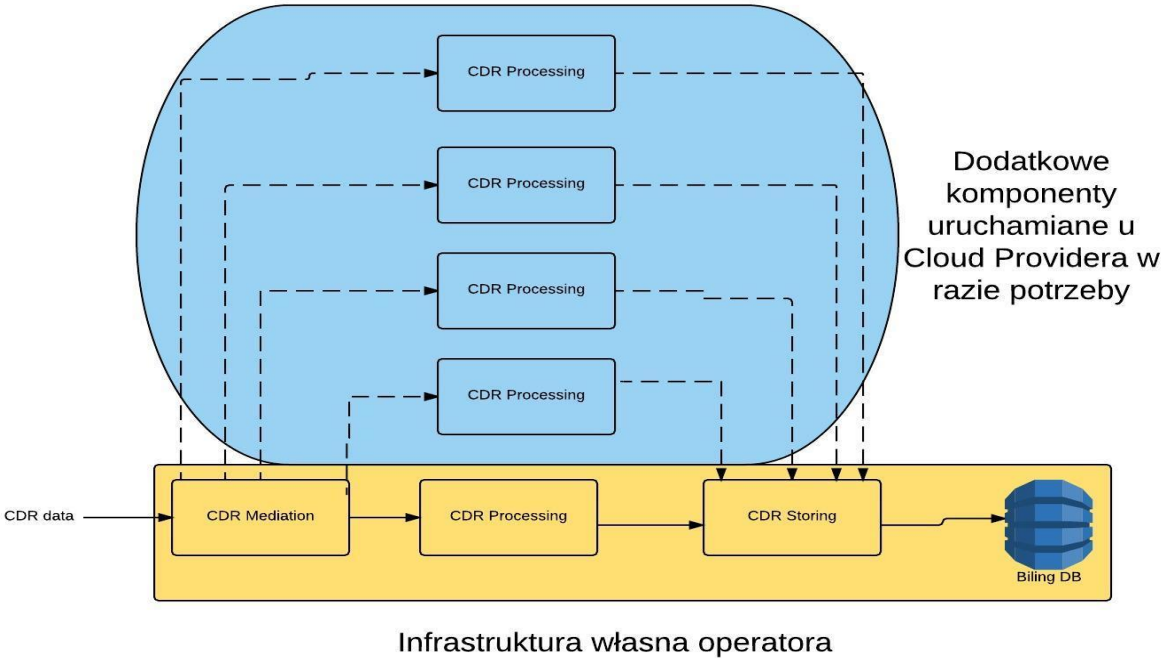
- Płatność tylko za realne wykorzystanie
- Automatyczne skalowanie zasobów bez limitu
- Wybór najlepszego Cloud Providera
- Strumienie i Big Data
- Niezależność od awarii sprzętu

Dywersyfikacja - multicloud

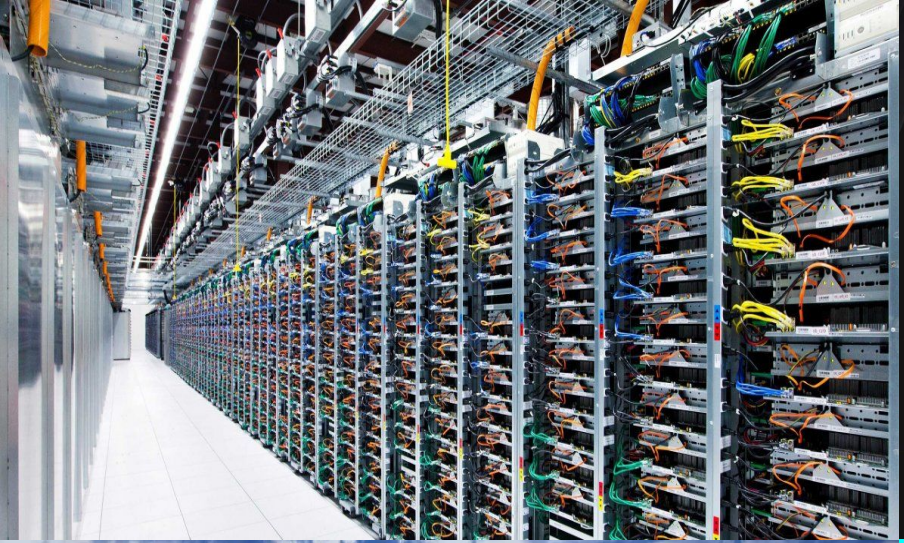
- Brak vendor lock
- Zwiększenie elastyczności wyboru
- Zmniejszenie ryzyka awarii lub utraty danych
- Wybór tego co najlepsze

Elastyczność, zwiększanie wydajności

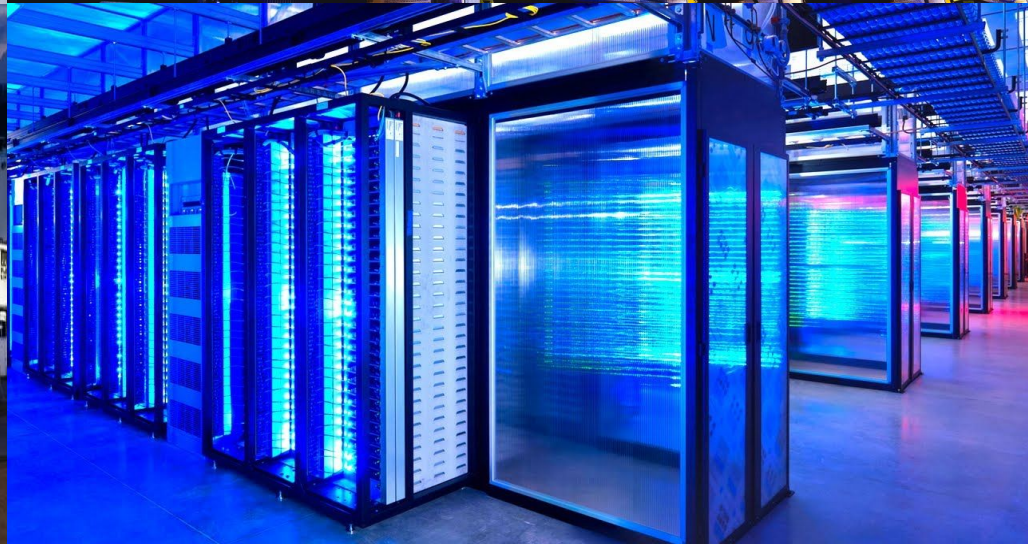
Przykład: System bilingowy operatora telekomunikacyjnego



Przykład:AWS - 42 centra działające




Google | google.com/datacenters



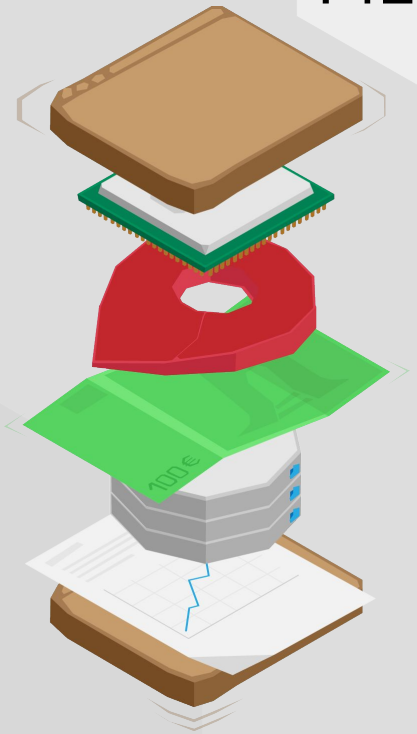
Przykład: AWS Compliance

- ISO 9001 Global Quality Standard,
- ISO 27001 Security Management Standard,
- ISO 27017 Cloud Specific Controls,
- ISO 27018 Personal Data Protection,
- PCI DSS Level 1 Service Provider (The Payment Card Industry Data Security Standard founded by American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa Inc.),
- SOC Compliance (AWS Service Organization Control (SOC) Reports)
- CSA - Cloud Security Alliance Controls,
- C5 [Germany] - Operational Security Attestation,
- Cyber Essentials Plus [UK] - Cyber Threat Protection,
- G-Cloud [UK] - UK Government Standards,
- IT-Grundschutz [Germany] - Baseline Protection Methodology
- ...

https://aws.amazon.com/compliance/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Przykład: Melodic - optymalne użycie multi-cloud

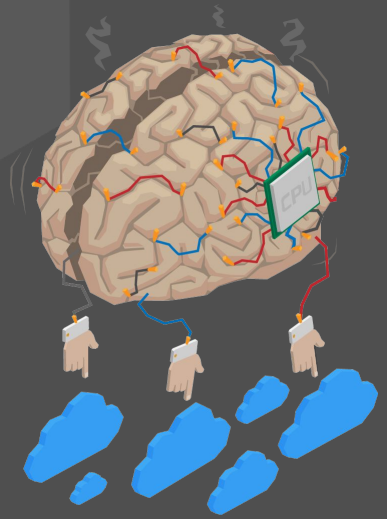


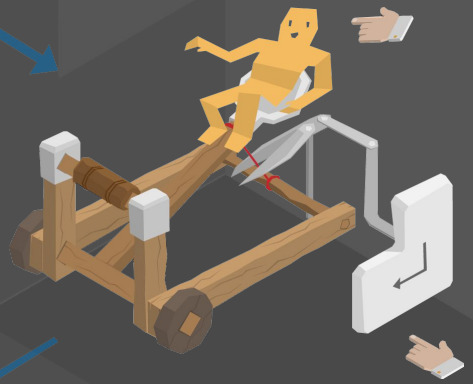
- Open source
- Europejski projekt badawczo-rozwojowy
- Zarządzanie Multicloud/Big Data
- Migracja aplikacji
- Optymalizacja infrastruktury i aplikacji
- Uruchamianie aplikacji u wielu Cloud Providers
- Bezpieczeństwo połączeń aplikacji
- Brak vendor lock

Melodic
Big data cloud

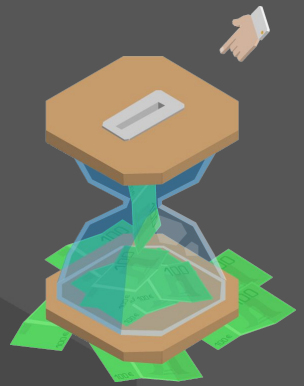
BIG-DATA CLOUD MADE EASY

Melodic wylicza najlepszą opcję multi-cloud dla Twojej aplikacji



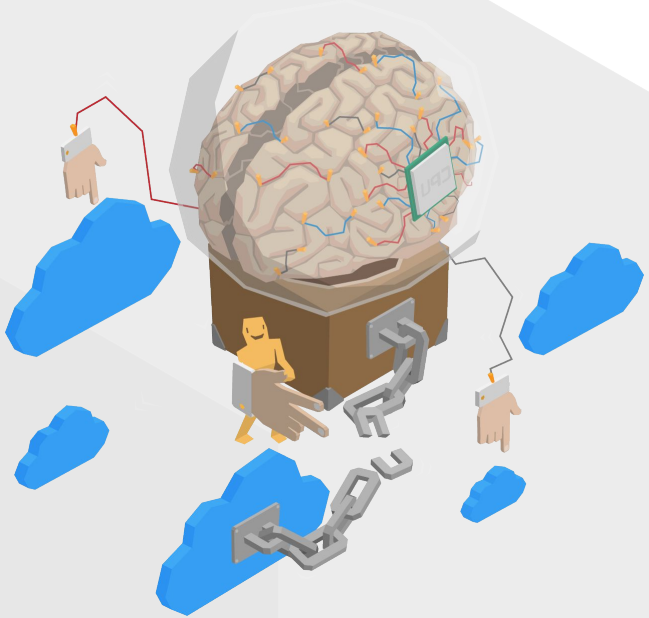


Deployment jest automatyczny

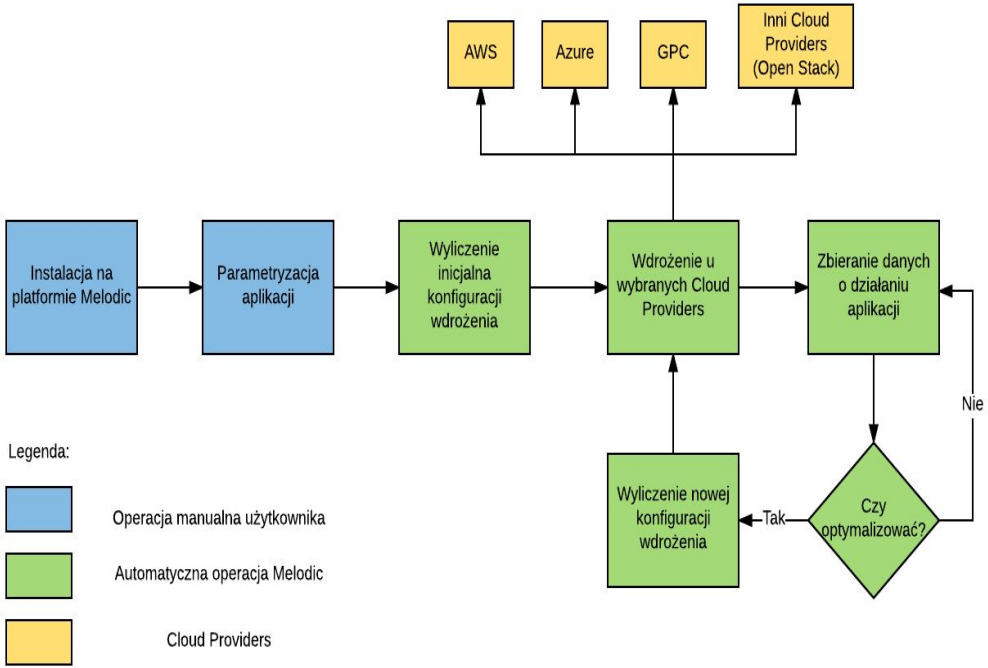


Nie tracisz czasu i pieniędzy

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 731664



Melodic - jak działa



Bezpiecznie w chmurach (sic)

Powszechnie stosowane bezpieczeństwo obwodowe nie jest już wystarczające:

- podatność rozwiązań masowych na włamania z zewnątrz,
- właściciel przechowywanych treści nie ma wpływu na infrastrukturę,
- personel administratora infrastruktury ma dostęp do przechowywanych treści,
- także w przypadku szyfrowania baz danych - klucze przechowywane są na serwerach,
- wysokie standardy bezpieczeństwa, potwierdzane statystycznie niską stopą błędu są, z perspektywy poszkodowanego, bezwartościowe,

SiC™ - Safely in the Cloud

Bezpieczeństwo od podstaw - security by design:

- najwyższy możliwy, dla środowiska rozproszonego, stopień zabezpieczenia poufności, autentyczności oraz integralności przetwarzanej informacji,
- trójwarstwowa architektura zgodna z modelem klient – serwer,
- niezależność poziomu bezpieczeństwa od lokalizacji serwera aplikacji i serwera bazy danych, jak również tytułu prawnego do korzystania z nich,
- niedostępność treści dla personelu administratora infrastruktury,
- hierarchiczne zarządzanie dostępem,
- rozliczalność operacji na zdeponowanych w SIC™ obiektach.

SiC™ - zabezpieczenie informacji

- szyfrowanie algorytmami asymetrycznymi (kryptografia klucza publicznego),
  - szyfrowanie i deszyfrowanie wyłącznie w aplikacji klienckiej na stacji Subskrybenta, wymaga użycia klucza prywatnego użytkownika,
  - zapisywanie informacji w relacyjnej bazie danych w postaci szyfrogramów,
- klucz prywatny przechowywany na karcie kryptograficznej lub (niezalecane) w zaszyfrowanym pliku na stacji osoby uprawnionej,
- komunikacja dwustronna pod osłoną bezpiecznego protokołu TLS w aktualnie zalecanej wersji, obejmująca wymianę danych i informacji uwierzytelniających

SiC™– zarządzanie dostępem

- dostęp do repozytorium mają wyłącznie osoby dysponujące stosownymi, zarejestrowanymi w systemie, kluczami prywatnymi,
- dostawca infrastruktury nie ma dostępu do informacji, nie ma zatem potrzeby zawierania umów o powierzeniu przetwarzania danych osobowych (w trybie art. 31 ust. u.o.d.o.),
- jest możliwość definiowania puli adresów IP, z których można się logować do systemu,
- certyfikaty są rejestrowane przez wskazaną osobę reprezentującą Subskrybenta usługi,

SIC™ – zarządzanie dostępem

- Hierarchiczna struktura uprawnień:
  - dostawca rozwiązania rejestruje w procesie wdrożenia certyfikat nadrzędny, zapewniający ciągłość zarządzania, przypisując go Subskrybentowi,
  - pozostałe certyfikaty są rejestrowane przez wskazaną osobę reprezentującą Subskrybenta,
  - nie ma możliwości modyfikowania własnych uprawnień ani uprawnień osób na poziomach nadrzędnych,
- pełna rozliczalność działań użytkowników:
  - listy kontroli dostępu,
  - pełna historia zmian,
  - dostępność wszystkich poprzednich wersji obiektów.

SiC™ - Funkcjonalność

- zaimplementowana możliwość składania podpisu elektronicznego:
  - niekwalifikowanego,
  - kwalifikowanego przy użyciu klucza na karcie kryptograficznej,
- przygotowywana możliwość korzystania z serwerowego podpisu kwalifikowanego,
- możliwość nieograniczonej rozbudowy struktury repozytorium,
- możliwość dynamicznej rekonfiguracji repozytorium,
- możliwość automatycznego odwzorowania struktury organizacyjnej Subskrybenta, tworzenia dowolnych grup roboczych oraz nadawania uprawnień dostępu zgodnie z aktualnymi potrzebami,
- możliwość udostępniania wydzielonych zasobów zewnętrznym organizacjom partnerskim, z uwzględnieniem ich struktur organizacyjnych.

Bezpieczeństwo - zagrożenia IT

Model STRIDE

Zdarzenie	Jak w chmurze?
Spoof of identity - podszycie się	=
Tampering - zmiana oprogramowania	=
Repudiation - autentyczność operacji	=
Information disclosure - ujawnienie danych	=
Denial of service - odmowa usług	
Elevation of privilege - zwiększenie poziomu dostępu	=

Legenda:  lepiej  gorzej = tak samo

Ciągłość prowadzenia działalności

Zdarzenie	Jak w chmurze?
Bezpieczeństwo fizyczne - zagrożenia ludzkie i naturalne	
Centrum zapasowe	
Możliwość zwiększania zasobów	
Backup i replikacja	
Vendor lock-in	
Koszt prowadzenia działalności IT	

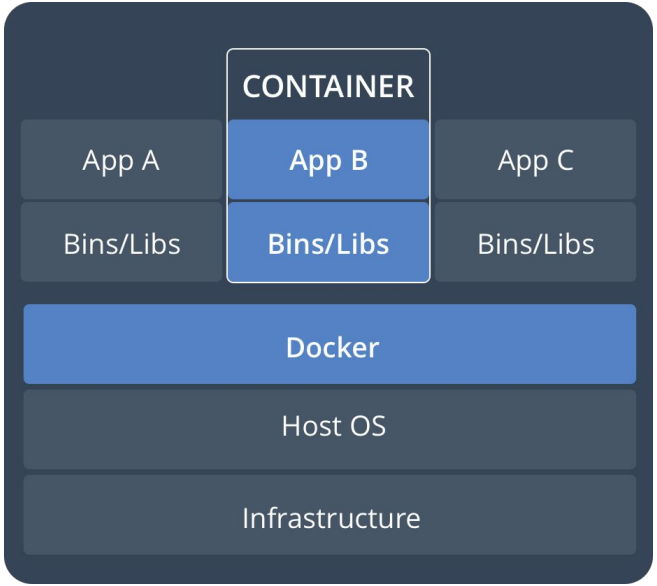
Legenda:  epiej  gorzej  tak samo

Pytania, uwagi, sugestie, kwestie otwarte...

backup

CaaS – Container as a Service

CaaS – Container as a Service – operator usług CaaS udostępnia platformę umożliwiającą **uruchamianie kontenerów** według zadanych parametrów.



Ciągłość prowadzenia działalności

Continuous business operations

- Bezpieczeństwo fizyczne - zagrożenia ludzkie i naturalne 👍
- Centrum zapasowe 👍
- Możliwość zwiększania zasobów 👍
- Backup i replikacja 👍
- Vendor lock-in 👎 (🟰)
- Koszt prowadzenia działalności IT 👍

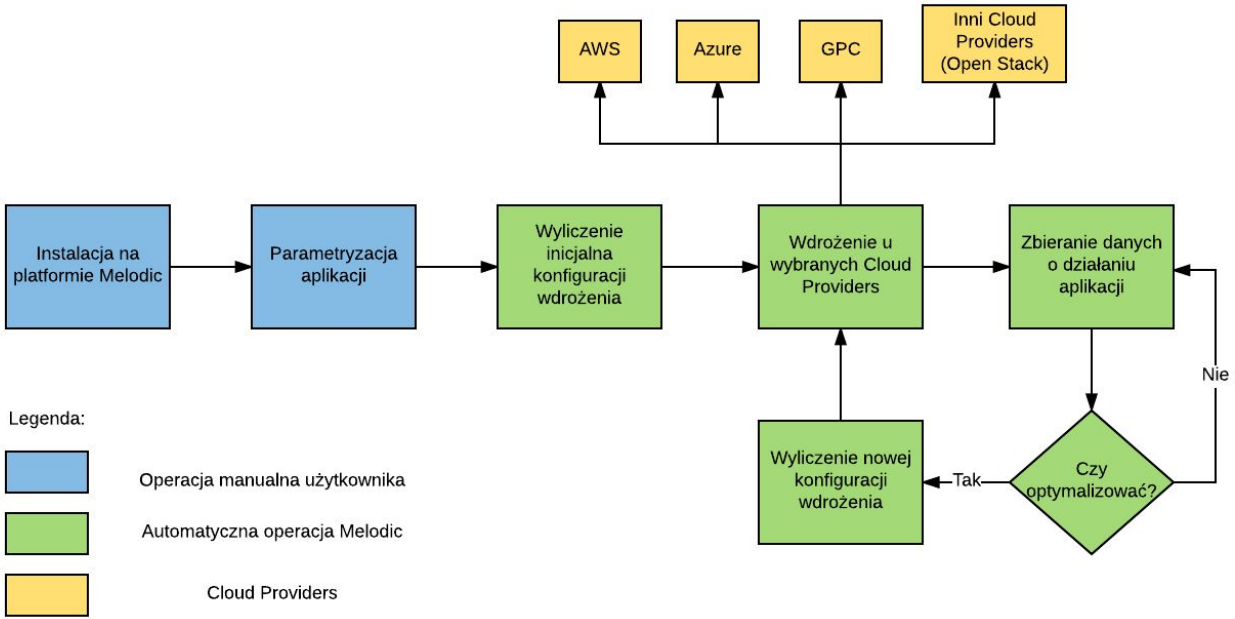
Bezpieczeństwo - zagrożenia IT - Model STRIDE

- Spoof of identity - podszycie się
- Tampering - zmiana oprogramowania
- Repudiation - autentyczność operacji
- Information disclosure - ujawnienie danych
- Denial of service - odmowa usług  w chmurach
- Elevation of privilege - zwiększenie poziomu dostępu

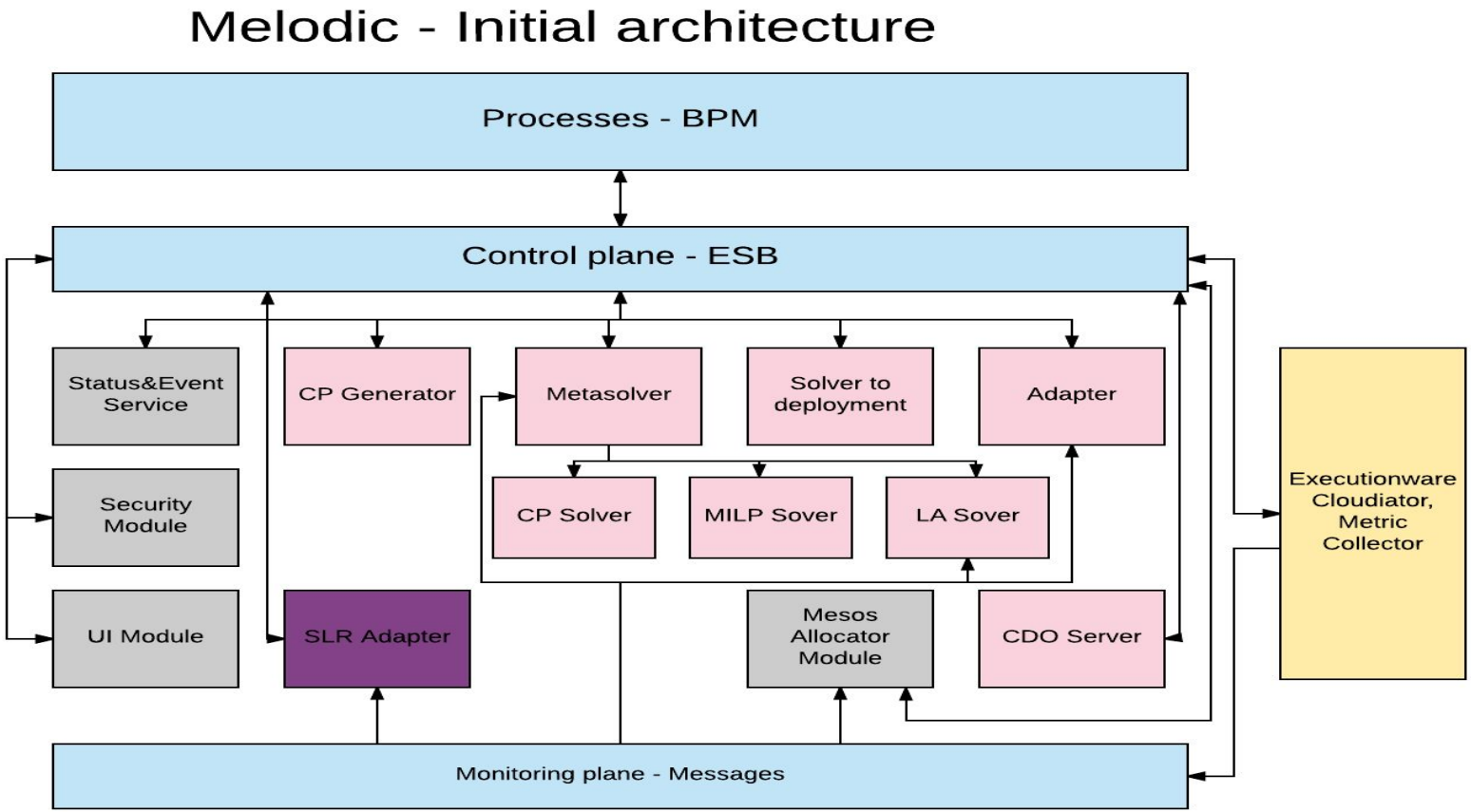
Przykład: Melodic - optymalne użycie multi-cloud

- Platforma zarządzania Multicloud/Big Data
- Migracji aplikacji
- Optymalizacja infrastruktury i aplikacji
- Uruchamianie aplikacji u wielu Cloud Providers
- Security połączeń aplikacji
- Brak vendor lock
- Open source
- Projekt europejski

Melodic - jak działa

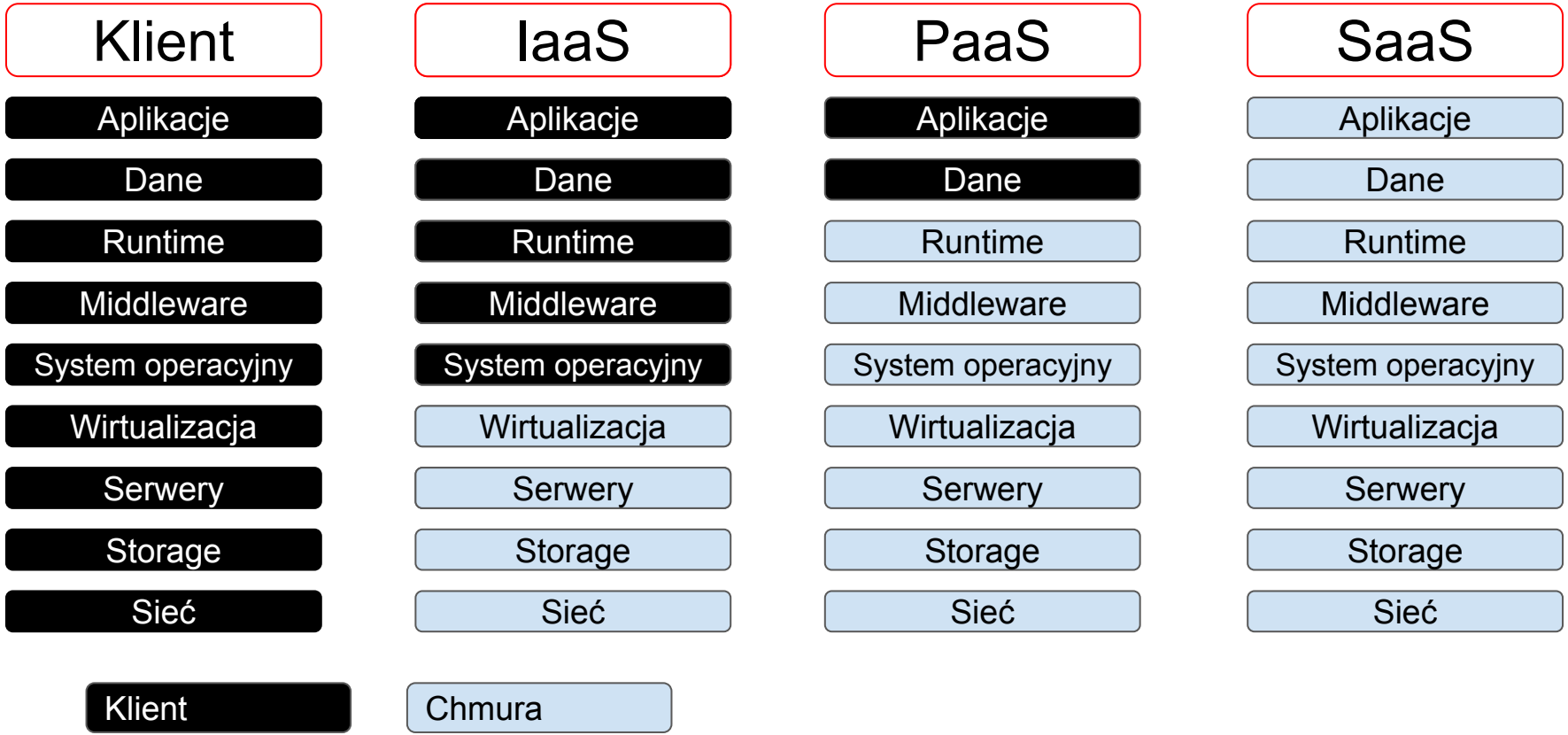


Melodic - jak wygląda



7bulls.com

Modele chmury obliczeniowej



Melodic - projekt

- Projekt jest realizowany w ramach środków przyznanych przez Komisję Europejską
- Projekt rozpoczął się 2016.12.01
- Czas trwania 3 lata
- Pierwsza iteracja po 18 miesiącach
- System będzie dostępny w modelu Open Source
- 7bulls.com odpowiedzialny za jakość, integrację

Dywersyfikacja - multicloud

- Rozwiązanie najlepiej dopasowane do specyficznych potrzeb aplikacji/dnych
- Unikanie barier wydajnościowych
- Ciągłe innowacje w ramach wyspecjalizowanych rozwiązań danego dostawcy

Multicloud, czyli automatyzacja i optymalizacja przetwarzania w chmurze

Podejście agnostyczne względem ofert poszczególnych dostawców usług chmurowych pozwala dynamicznie optymalizować działanie – opartego na tych usługach – środowiska aplikacyjnego. Dotyczy to zarówno wymiaru operacyjnego, jak i kosztowego. Dzisiejsze rozwiązania umożliwiają automatyczne tworzenie i przenoszenie obciążeń pomiędzy różnymi platformami cloud computing. Dzięki temu jedna aplikacja może działać jednocześnie w wielu środowiskach chmurowych.

Model chmury obliczeniowej, czyli możliwość przeniesienia systemów i procesów przetwarzania danych do wydzielonej, współdzielonej lub zewnętrznej infrastruktury, wprowadza zmianę paradygmatu funkcjonowania IT. Za tym jednak idą także nowe wyzwania związane, przykładowo, z koncepcją definiowanej programowo infrastruktury. Dzięki niej zasoby IT – w tym serwery, dyski i sprzęt sieciowy – mogą być traktowane nie jako konkretne, fizyczne urządzenia, ale jako wysoce elastyczne byty definiowane programowo.

Jednocześnie praktyka pokazuje, że poleganie na zasobach chmurowych jednego dostawcy rodzi realne ryzyko biznesowe. Tu z pomocą przychodzi podejście określane mianem multicloud. Zakłada ono wykorzystanie wielu, oferowanych przez różnych dostawców, usług cloud computing w ramach jednej, heterogenicznej i dynamicznie zmiennej architektury. Podejście oparte na multicloud daje m.in. możliwość przenoszenia procesów przetwarzania do tych zasobów i środowisk chmurowych, które w danym momencie lub w danych warunkach okazują się bardziej adekwatne, bezpieczne lub po prostu tańsze. Takie podejście umożliwia wykorzystanie najlepszych cech każdego z rozwiązań chmurowych, po jak najniższych kosztach i jednocześnie zabezpiecza klienta przed zjawiskiem znanym jako „vendor lock-in".

[RÓŻNE MODELE CHMURY I RÓŻNE USŁUGI CLOUD]

Nowy model dostarczania usług IT zrewolucjonizował myślenie o administrowaniu zasobami IT w wielu firmach. Chmura obliczeniowa jest dziś oczywistym wyborem dla wielu organizacji. Wynika to ze wspomnianej elastyczności, możliwości lepszego wykorzystania posiadanych zasobów sprzętowych (chmura prywatna), a także ograniczenia kosztów pozyskania potrzebnych zasobów IT oraz utrzymania infrastruktury (chmura publiczna). Model cloud ułatwia również tworzenie nowych aplikacji i usług, przechowywanie danych, wykonywanie i odzyskiwanie kopii zapasowych oraz przetwarzanie dużych zbiorów informacji. Utrzymanie tych samych usług w ramach klasycznej architektury IT – w sytuacji, gdy istnieją tańsze, bezpieczniejsze i bardziej elastyczne rozwiązania – często staje się wręcz nieopłacalne.

Warto pamiętać, że na rynku istnieją różne rodzaje rozwiązań chmurowych. Im lepsza znajomość ich specyfiki, możliwości oraz ograniczeń, tym łatwiejsze będzie osiąganie oczekiwanych po modelu chmury celów biznesowych. Większość dostępnych na rynku usług typu cloud należy do jednej z trzech ogólnych kategorii: infrastruktura jako usługa (IaaS), platforma jako usługa (PaaS) lub oprogramowanie jako usługa (SaaS). Infrastruktura chmurowa zaś może być wykorzystywana w formie współdzielonej (publicznej), lokalnej (prywatnej) oraz łączącej oba te modele (hybrydowej).

Model IaaS (Infrastructure as a Service) to najbardziej podstawowa kategoria usług chmury obliczeniowej, gdzie infrastruktura IT – serwery, maszyny wirtualne, zasoby dyskowe, sieci i systemy operacyjne – zostaje wynajęta od dostawcy chmury w modelu płatności zgodnie z rzeczywistym użyciem.

Z kolei model PaaS (Platform as a Service) zaprojektowano tak, aby ułatwić deweloperom szybkie tworzenie aplikacji bez przejmowania się konfigurowaniem niezbędnej podstawowej infrastruktury serwerów, pamięci masowych, sieci i baz danych oraz zarządzaniem takim środowiskiem.

Odmianą modelu PaaS są usługi określane jako FaaS (Function as a Service). Kluczowym czynnikiem odróżniającym te modele jest fakt, że w modelu FaaS komponenty tworzone są tylko na czas wykonania zadania, a po jego realizacji są usuwane. Podejście takie pozwala znacząco ograniczyć koszty rozwiązania, które nie rosną, gdy obciążenie systemu jest niskie. Zwiększa też skalowalność i umożliwia tworzenie komponentów na żądanie, w liczbie odpowiedniej do wykonania zadania. Model FaaS jest najnowszym modelem rozwiązań chmurowych, obecnie oferowany komercyjnie przez niewielką jeszcze liczbę dostawców. Ma on jednak pewne ograniczenia, np. maksymalny czas życia komponentu.

Natomiast usługi typu SaaS (Software as a Service) sprowadzić można do dostarczenia w formie usługi jedynie określonych funkcjonalności oprogramowania. Klient płaci za każdorazowe korzystanie z usługi, a dostęp do niej uzyskuje na żądanie. Należy zauważyć, że SaaS to najszybciej rozwijający się segment technologii, który może funkcjonować na bazie PaaS i IaaS.

Ciekawym rozwiązaniem rozwijanym pod kątem wsparcia języka CAMEL jest platforma Melodic. Jest to środowisko umożliwiające optymalizację i wdrażanie aplikacji opisanych w języku CAMEL w modelu multicloud. Platforma Melodic jest projektem europejskim, finansowanym w ramach programu Horizon 2020, i rozwijanym w modelu open source. **Jej funkcjonalność pozwala na uruchamianie aplikacji modelowanych przy użyciu języka CAMEL w sposób w pełni agnostyczny względem dostawcy usług chmurowych, a także wybranie najbardziej optymalnego modelu wdrożenia, w zależności od charakterystyki aplikacji.**

[PIĘĆ FILARÓW ARCHITEKTURY CLOUD]

Pomimo że ze środowisk chmurowych korzysta coraz większa liczba przedsiębiorstw, a wśród nich spora część wybiera rozwiązania multicloud, to w trakcie ich wdrażania i utrzymania mogą pojawić się pewne trudności. Dotyczą one najczęściej zarządzania jednocześnie wieloma usługami i licencjami, potrzeby budowania i zarządzania kolejnymi środowiskami aplikacyjnymi, a także niejednolitymi zasadami bezpieczeństwa. W zapewnieniu właściwego i bezpiecznego wdrożenia, a później administrowania usługami w chmurach obliczeniowych, pomagają sprawdzone metodologie. Jedną z nich jest wypracowana przez Amazon filozofia pięciu filarów dobrze zaprojektowanej architektury cloud. Są to:

1. **Bezpieczeństwo** – autentykacja i autoryzacja użytkowników i aplikacji, szyfrowanie danych, certyfikacja i inne kwestie związane z bezpieczeństwem.
2. **Niezawodność** – dostępność i stabilność systemów, redundancja, w tym geograficzna, odporność na awarie.
3. **Wydajność** – efektywność i skalowalność użycia zasobów, w szczególności wykorzystanie mechanizmu automatycznego skalowania (autoscaling).
4. **Optymalizacja kosztów** – podejście do projektowania architektury chmurowej, tak aby minimalizować całkowite koszty, m.in. za sprawą dynamicznego dostosowywania wielkości środowisk (skalowania w zależności od potrzeb) i optymalizacji procesów backupu.
5. **Utrzymanie operacyjne** – wiele zadań związanych z procedurami operacyjnymi utrzymania i rozwoju systemów, w szczególności

Główni dostawcy usług chmurowych zwykle udostępniają własne, specyficzne interfejsy API. Ich unikalność powoduje jednak, że przenoszenie dużych systemów do platform chmurowych innych dostawców staje się utrudnione. Rozwiązaniem tego problemu jest standard TOSCA (Topology and Orchiestration Specification for Cloud Application) **i stworzony na jej podstawie język CAMEL** (Cloud Application Modeling and Execution Language), pozwalający opisać aplikację, jej wymagania i infrastrukturę w sposób niezależny od konkretnego dostawcy.

automatyzacja zadań Continuous Integration oraz Continuous Delivery.

Równolegle z rozwojem modelu cloud computing daje się zaobserwować kilka innych znaczących trendów, w tym: upowszechnienie koncepcji infrastruktury definiowanej programowo, dążenie do skrócenia średniego czasu dostarczania oprogramowania, a także zautomatyzowanie procesów testowania i wdrażania oprogramowania.

Typowe rozwiązanie Continuous Integration/ Continuous Delivery (CI/CD) realizuje następujące kroki:

1. **Budowanie oprogramowania** – z określonej gałęzi z systemu kontroli wersji budowane jest oprogramowanie z wykorzystaniem wybranego narzędzia do budowania (najczęściej Maven lub Gradle).
2. **Weryfikacja rozwiązania** – pod kątem analizy statycznej jakości kodu przez oprogramowanie SonarQube lub analogiczne.
3. **Umieszczenie zbudowanego oprogramowania w repozytorium**.
4. **Wykonanie testów jednostkowych** – na zbudowanym oprogramowaniu wykonywane są testy weryfikujące działanie metod i kluczowych elementów oprogramowania.
5. **Wdrożenie na środowisko testowe** – automatyczne wdrożenie oprogramowania na wybrane środowisko testowe, w szczególności wraz ze stworzeniem infrastruktury (maszyny wirtualne, dyski itp.)
6. **Wykonanie automatycznych testów akceptacyjnych** – funkcjonalnych, wydajnościowych, bezpieczeństwa i innych.
7. **Wsparcie testów manualnych** – manualne wykonanie przypadków testowych przez testerów.
8. **Wdrożenie na wybrane środowisko docelowe** – automatyczne wdrożenie na środowisko produkcyjne, po akceptacji procesu testowego.

Wdrożenie w pełni zautomatyzowanego rozwiązania klasy CI/CD znacząco przyspiesza proces dostarczania oprogramowania, a przede wszystkim pozwala na jego powtarzalność. Dostarczana wersja zawsze jest budowana i wdrażana w taki sam sposób oraz w pełni automatycznie, co minimalizuje ryzyko pomyłek. Najbardziej popularne rozwiązania klasy CI/CD to rozwiązania open source Jenkins lub Bamboo, dostarczane przez firmę Atlassian.

Jednocześnie rozwiązania klasy Continuous Integration, których zadaniem jest budowa i testowanie przy użyciu testów jednostkowych, są wykorzystywane od pewnego czasu także w środowiskach typu on-premise, czy w odniesieniu do infrastruktury Bare Metal. Tymczasem rozwój rozwiązań Continuous Deployment jest możliwy tylko na bazie modelu chmury. W tradycyjnych środowiskach IT zwykle nie ma bowiem możliwości tworzenia infrastruktury w sposób programowy, co jest niezbędne w procesie Continuous Deployment.

Istotnym elementem, na który należy zwrócić uwagę w kontekście planowania migracji do środowiska cloud computing, jest zapewnienie jakości usług IT. Zastosowanie mechanizmów programowego definiowania infrastruktury wymaga bowiem ciągłego i wielowymiarowego testowania systemów aplikacyjnych. Dotyczy to w dodatku aspektów wykraczających poza typowe rozwiązania do zarządzania jakością aplikacji. Konieczne jest wprowadzenie testów bezpieczeństwa, najlepiej wraz z testami penetracyjnymi, w szczególności w kontekście połączeń między komponentami aplikacyjnymi oraz samymi aplikacjami. Ze względu na elastyczność, skalowalność i redundancję rozwiązań cloud testy takie powinny zakładać m.in. obligatoryjną weryfikację, czy jakiś element nie jest wąskim gardłem w danym rozwiązaniu.

[MULTICLOUD SPOSOBEM NA OGRANICZENIE RYZYKA]

Wraz z coraz większym wykorzystaniem rozwiązań chmurowych znaczenia zaczęła nabierać możliwość przenoszenia obciążeń pomiędzy środowiskami różnych dostawców, a także tworzenia aplikacji w taki sposób, aby możliwe było jej automatyczne wdrożenie w różnych środowiskach. Takie podejście przynosi znaczące zyski organizacji korzystającej z rozwiązań chmurowych. Pozwala bowiem wybrać najlepszego usługodawcę pod kątem danej aplikacji, a nawet – praktycznie na bieżąco – optymalizować koszty, zwiększyć bezpieczeństwo i stabilność rozwiązania poprzez dywersyfikację kontrahentów.

Warto jednak pamiętać, że czołowi dostawcy usług chmurowych zwykle udostępniają własne, specyficzne interfejsy pozwalające sterować dostarczaną infrastrukturą z poziomu aplikacji. Ich unikalność powoduje jednak, że przenoszenie dużych systemów do platform chmurowych innych dostawców staje się utrudnione.

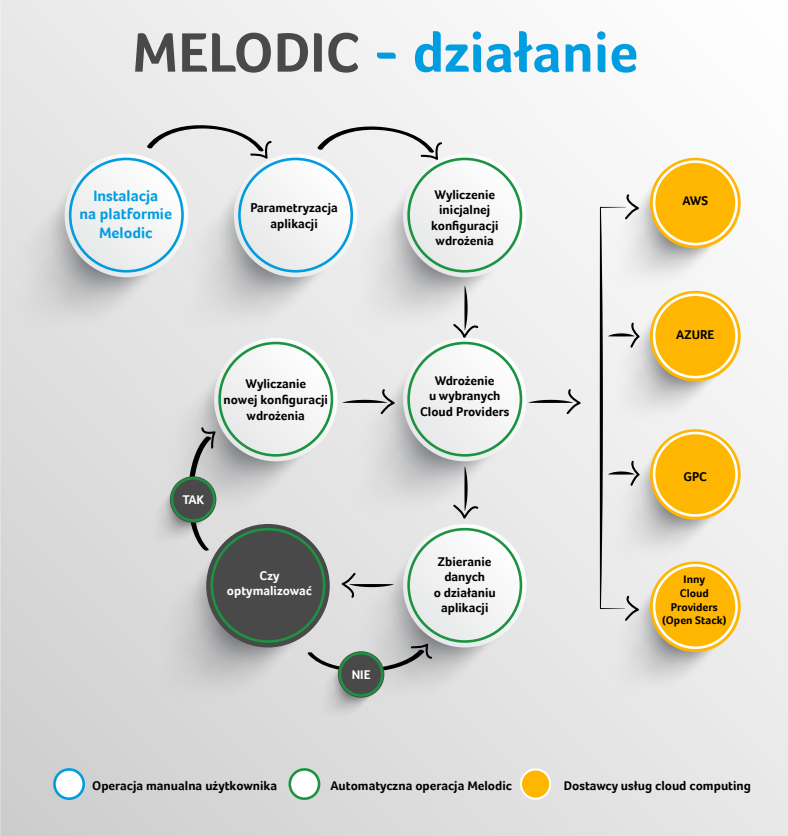
Rozwiązaniem tego problemu jest standard TOSCA (Topology and Orchestration Specification for Cloud Application) i stworzony na jego podstawie język CAMEL (Cloud Application Modeling and Execution Language), pozwalający opisać aplikację, jej wymagania i infrastrukturę w sposób niezależny od konkretnego dostawcy. Dzięki temu możliwe jest zarówno wdrażanie aplikacji w różnych środowiskach, jak i optymalizacja działania aplikacji, w zależności od charakterystyki jej wykonania i wymagań określonych przez użytkownika.

[CAMEL W PRAKTYCE]

Ciekawym rozwiązaniem rozwijanym pod kątem wsparcia języka CAMEL jest platforma Melodic. Jest to środowisko umożliwiające optymalizację i wdrażanie aplikacji opisanych w języku CAMEL w modelu multicloud. Platforma Melodic jest projektem europejskim, finansowanym w ramach programu Horizon 2020, i rozwijanym w modelu open source. Jej funkcjonalność pozwala na uruchamianie aplikacji modelowanych przy użyciu języka CAMEL w sposób w pełni niezależny względem dostawcy usług chmurowych, jak również wybranie najbardziej optymalnego modelu wdrożenia, w zależności od charakterystyki aplikacji. Dodatkowym elementem platformy jest zdolność do optymalizacji rozwiązań big data i obsługi lokalności danych (data locality awareness).

Proces modelowania aplikacji w języku CAMEL w ramach platformy Melodic w pierwszej kolejności obejmuje określenie jej komponentów, połączeń między nimi, a także wymagań odnośnie do wydajności i zasobów oraz sposobu wdrażania aplikacji. W dalszym kroku następuje automatyczne optymalizowanie konfiguracji wdrożenia – platforma „decyduje", jakiej i gdzie umieszczonej infrastruktury należy użyć. Wstępna optymalizacja jest dokonywana na podstawie parametrów określonych przez użytkownika.

Na podstawie konkretnej, optymalnej konfiguracji automatycznie tworzona jest ściśle zdefiniowana infrastruktura u wybranych dostawców usług chmurowych (maszyny wirtualne o ustalonych parametrach) i wdrażana jest aplikacja, wraz z ustawieniami połączeń między komponentami. Po wdrożeniu aplikacja jest monitorowana – zbierane są m.in. metryki określające charakterystykę jej działania, które jednocześnie stanowią podstawę do automatycznej optymalizacji wdrożenia aplikacji uruchomionej w ramach platformy Melodic.



[AUTOMATYZACJA TO PRZYSZŁOŚĆ CHMURY]

Rozwiązania chmurowe stopniowo stają się powszechne wśród poszczególnych firm i organizacji, a także dostawców rozwiązań IT, również w Polsce. Pozwalają bowiem na znaczące obniżenie nakładów finansowych przeznaczanych na rozwiązania IT, zarówno na poziomie sprzętowym (brak konieczności posiadania serwerowni i sprzętu), jak i kompetencyjnym (istotna część zadań wykonywana jest przez dostawcę usług chmurowych). Właściwe wykorzystanie tego potencjału wymaga jednak wiedzy o możliwościach i ryzykach wynikających z zastosowania zasobów chmurowych na dużą skalę oraz umiejętności dostosowania zasobów i aplikacji do potrzeb danej organizacji.

Podejście wdrożone w platformie Melodic to przyszłość rozwiązań cloud computing. W szczególności użycie zaawansowanych algorytmów machine learning pozwala coraz lepiej optymalizować wdrażanie aplikacji, co dla dużych systemów może mieć bardzo istotne znaczenie, zarówno co do wydajności, jak i kosztów.

Paweł Skrzypek,
Melodic & AWS Cloud Architect w firmie 7bulls.com;
Karolina Watras,
Senior Technical Writer w firmie 7bulls.com

Multicloud – dywersyfikacja, optymalizacja i bezpieczeństwo w chmurze

Abstrakt
Agnostyczne podejście względem ofert poszczególnych dostawców usług chmurowych skłania do rozważań, jak dynamicznie optymalizować korzystanie z opartego na tych usługach środowiska IT. Dotyczy to zarówno wymiaru operacyjnego, jak i kosztowego. Współczesne technologie umożliwiają przecież automatyczne tworzenie i przenoszenie obciążeń pomiędzy różnymi platformami cloud computing. Dzięki temu jedna aplikacja może działać jednocześnie w wielu środowiskach chmurowych, a uzyskanie płynności i niezawodności tego procesu łagodzi, postrzegane jako poważne ryzyko, relacje prowadzące do uzależnienia odbiorcy usługi od jej dostawcy (tzw. zjawisko *vendor lock-in)*.
Inny, niezależny nurt poszukiwań zmierza ku rozwiązaniom wychodzącym naprzeciw obawom utraty kontroli nad ważnymi informacjami przeniesionymi do zasobów wyniesionych.

MULTICLOUD – WSZYSTKO W JEDNYM

Idea przenoszenia systemów i procesów przetwarzania danych do wydzielonej, współdzielonej lub zewnętrznej infrastruktury, przy zapewnieniu nieprzerwanego dostępu do niej przez Internet, nazwana *chmurą obliczeniową*, inspiruje rozważenie potrzeby zmiany paradygmatu funkcjonowania IT. Efekt skali, nie tylko kosztowy, ale także techniczny i technologiczny, będący zapłonnikiem myślenia kategoriami *chmury*, jest jednak moderowany świadomością konsekwencji braku bezpośredniej kontroli nad zasobami zewnętrznymi, relatywnie dużej niezależności dostawcy usług chmurowych od zamawiającego. To ograniczenie blokuje niejedną inicjatywę, zwłaszcza w kontekście powszechnej opinii o trudnej odwracalności procesu migracji do *chmury*.
Z kolei nieustanny rozwój idei *chmury* stymuluje pojawianie się kolejnych pomysłów udoskonalających sposoby jej wykorzystywania, a zarazem redukujących ograniczenia. Jednym z nich jest koncepcja definiowanej programowo infrastruktury. Wizja, w ramach której zasoby IT - w tym serwery, dyski, sprzęt sieciowy, itp. – mogą być traktowane nie jako fizyczne urządzenia, ale jako wysoce elastyczne byty opisywane i konfigurowane za pomocą inteligentnych algorytmów. Pochodną takiej wizji jest podejście określane mianem *multicloud*, zakładające wykorzystanie wielu, oferowanych przez różnych dostawców, usług *cloud computing* w ramach jednej, heterogenicznej i dynamicznie zmiennej architektury. Takie podejście daje m.in. możliwość sprawnego, docelowo automatycznego, przenoszenia procesów przetwarzania do tych zasobów i środowisk chmurowych, które w danym momencie lub w danych warunkach okazują się bardziej adekwatne, bezpieczniejsze lub po prostu tańsze. *Multicloud* umożliwia optymalizację - wykorzystanie najlepszych cech każdego z dostępnych rozwiązań chmurowych, po jak najniższych kosztach. Zabezpiecza zarazem użytkownika przed konsekwencjami zbyt silnego powiązania z jednym dostawcą, relacją znaną jako „*vendor lock-in*” [i].

DYWERSYFIKACJA – RÓŻNE MODELE I RÓŻNE USŁUGI *CHMURY*

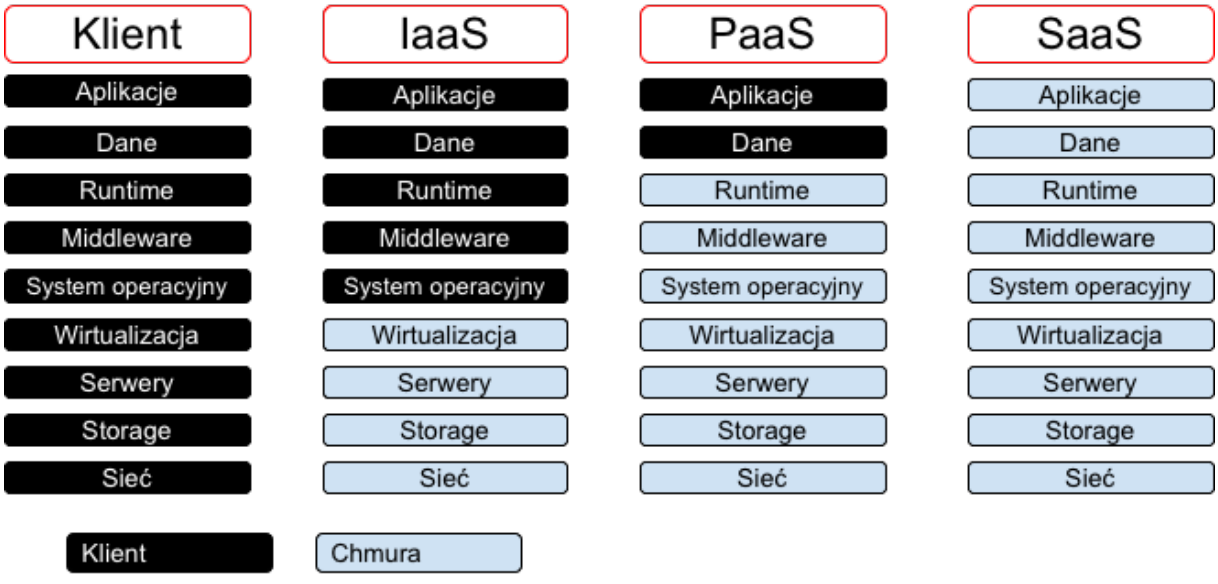
Nowy model dostarczania usług IT zrewolucjonizował myślenie o administrowaniu zasobami IT, a chmura obliczeniowa wydaje się być oczywistym wyborem. Wynika to ze wspomnianej elastyczności, możliwości lepszego wykorzystania posiadanych zasobów sprzętowych (chmura prywatna), a także ograniczenia kosztów pozyskania potrzebnych zasobów IT oraz utrzymania infrastruktury (chmura publiczna). Model *cloud* ułatwia również tworzenie nowych aplikacji i usług, przechowywanie danych, wykonywanie i odzyskiwanie kopii zapasowych oraz przetwarzanie dużych zbiorów informacji. Utrzymanie tych samych usług w ramach klasycznej architektury IT – w sytuacji, gdy istnieją tańsze, bezpieczniejsze i bardziej elastyczne rozwiązania – staje się nieopłacalne.
Na rynku istnieją różne rodzaje rozwiązań chmurowych. Im lepsza znajomość ich specyfiki, możliwości oraz ograniczeń, tym łatwiejsze będzie osiąganie oczekiwanych celów biznesowych. Większość dostępnych na rynku usług typu *cloud* należy do jednej z trzech ogólnych kategorii:

- infrastruktura jako usługa (IaaS),
- platforma jako usługa (PaaS) lub
- oprogramowanie jako usługa (SaaS).

Infrastruktura chmurowa zaś może być wykorzystywana w formie:

- współdzielonej (publicznej),
- lokalnej (prywatnej) oraz
- mieszanej (hybrydowej).

Kryterium określającym model korzystania z usług chmurowych jest podział zakresu administrowania poszczególnymi elementami zasobów IT (odpowiedzialności) pomiędzy dostawcą a użytkownikiem. Poglądowo przedstawiono to na diagramie poniżej:



Model IaaS (*Infrastructure as a Service*) to najbardziej podstawowa kategoria usług chmury obliczeniowej, gdzie infrastruktura IT – serwery, maszyny wirtualne, zasoby dyskowe, sieci i systemy operacyjne – zostaje wynajęta od dostawcy chmury w modelu płatności zgodnie z rzeczywistym użyciem.

Model PaaS (*Platform as a Service*) zaprojektowano tak, aby ułatwić deweloperom szybkie tworzenie aplikacji bez przejmowania się konfigurowaniem niezbędnej podstawowej infrastruktury serwerów, pamięci masowych, sieci i baz danych oraz zarządzaniem takim środowiskiem.
Odmianą modelu PaaS są usługi określane jako FaaS (*Function as a Service*). Kluczowym czynnikiem odróżniającym te modele jest fakt, że w modelu FaaS komponenty tworzone są tylko na czas wykonania zadania, a po jego realizacji są usuwane. Podejście takie pozwala znacząco ograniczyć koszty rozwiązania, które nie rosną, gdy obciążenie systemu jest niskie. Zwiększa też skalowalność i umożliwia tworzenie komponentów na żądanie, w liczbie odpowiedniej do wykonania zadania. Model FaaS jest najnowszym modelem rozwiązań chmurowych, obecnie oferowany komercyjnie przez niewielką jeszcze liczbę dostawców. Ma on jednak pewne ograniczenia, np. maksymalny czas życia komponentu.

Usługi typu SaaS (*Software as a Service*) sprowadzić można do dostarczenia w formie usługi jedynie określonych funkcjonalności oprogramowania. Klient płaci za każdorazowe korzystanie z usługi, a dostęp do niej uzyskuje na żądanie. Należy zauważyć, że SaaS to najszybciej rozwijający się segment technologii, który może funkcjonować zarówno na bazie PaaS jak i IaaS.

OPTYMALIZACJA – PIĘĆ FILARÓW ARCHITEKTURY CLOUD

Pomimo że ze środowisk chmurowych korzysta coraz większa liczba przedsiębiorstw, a wśród nich spora część wybiera rozwiązania *multicloud*, to w trakcie ich wdrażania i utrzymania mogą pojawić się pewne trudności, dotyczące najczęściej: zarządzania jednocześnie wieloma usługami i licencjami, potrzeby budowania i zarządzania kolejnymi środowiskami aplikacyjnymi, niejednolitych zasadami bezpieczeństwa i konieczności efektywnego nimi zarządzania. W zapewnieniu właściwego i bezpiecznego wdrożenia, a później administrowania usługami w chmurach obliczeniowych, pomagają sprawdzone metodologie. Jedną z nich jest wypracowana przez Amazon filozofia pięciu filarów dobrze zaprojektowanej architektury *cloud*. Są to:

1. **Bezpieczeństwo** – autentykacja i autoryzacja użytkowników i aplikacji, szyfrowanie danych, certyfikacja i inne kwestie związane z bezpieczeństwem.
2. **Niezawodność** – dostępność i stabilność systemów, redundancja, w tym geograficzna, odporność na awarie.
3. **Wydajność** – efektywność i skalowalność użycia zasobów, w szczególności wykorzystanie mechanizmu automatycznego skalowania (*autoscaling*).
4. **Optymalizacja kosztów** – podejście do projektowania architektury chmurowej, tak aby minimalizować całkowite koszty, m.in. za sprawą dynamicznego dostosowywania wielkości wykorzystywanych środowisk (skalowanie w zależności od potrzeb) i optymalizacji procesów backupu.
5. **Utrzymanie operacyjne** – wiele zadań związanych z procedurami operacyjnymi utrzymania i rozwoju systemów, w szczególności automatyzacja zadań *Continuous Integration* oraz *Continuous Delivery*.

Równolegle z rozwojem modelu *cloud computing* daje się zaobserwować kilka innych wartościowych trendów, w tym: upowszechnienie koncepcji infrastruktury definiowanej programowo, dążenie do skrócenia średniego czasu dostarczania oprogramowania, a także zautomatyzowanie procesów testowania i wdrażania oprogramowania.

BEZPIECZEŃSTWO – MULTICLOUD SPOSOBEM NA OGRANICZENIE RYZYKA

Wraz z coraz większym wykorzystaniem rozwiązań chmurowych znaczenia zaczęła nabierać możliwość przenoszenia obciążeń pomiędzy środowiskami różnych dostawców, a także tworzenia aplikacji w taki sposób, aby możliwe było jej automatyczne wdrożenie w różnych środowiskach. Takie podejście przynosi znaczące zyski organizacji korzystającej z rozwiązań chmurowych. Pozwala bowiem wybrać najlepszego usługodawcę pod kątem danej aplikacji, a nawet – praktycznie na bieżąco – optymalizować koszty, zwiększyć bezpieczeństwo i stabilność rozwiązania poprzez dywersyfikację kontrahentów.
Czołowi dostawcy usług chmurowych zwykle udostępniają własne, specyficzne interfejsy pozwalające sterować dostarczaną infrastrukturą z poziomu aplikacji. Ich unikalność powoduje jednak, że przenoszenie dużych systemów do platform chmurowych innych dostawców staje się utrudnione.
Z tego powodu zaproponowano standard TOSCA (*Topology and Orchestration Specification for Cloud Application*)[ii] i stworzony na jego podstawie język CAMEL (*Cloud Application Modeling and Execution Language*)[iii], pozwalający opisać aplikację, jej wymagania i infrastrukturę w sposób niezależny od konkretnego dostawcy. Dzięki temu możliwe jest zarówno wdrażanie aplikacji w różnych środowiskach, jak i optymalizacja działania aplikacji, w zależności od charakterystyki jej wykonania i wymagań określonych przez użytkownika.

CAMEL W PRAKTYCE

Ciekawym rozwiązaniem rozwijanym pod kątem wsparcia języka CAMEL jest platforma Melodic (*Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing*).[iv]

Jest to środowisko umożliwiające optymalizację i wdrażanie aplikacji opisanych w języku CAMEL w modelu *multicloud*. Platforma Melodic jest projektem europejskim, finansowanym w ramach programu Horizon 2020, i rozwijanym w modelu open-source. Jej funkcjonalność pozwala na uruchamianie aplikacji modelowanych przy użyciu języka CAMEL w sposób w pełni niezależny względem dostawcy usług chmurowych, jak również wybranie najbardziej optymalnego modelu wdrożenia, w zależności od charakterystyki aplikacji. Dodatkowym elementem platformy jest zdolność do optymalizacji rozwiązań big data i obsługi lokalności danych (*data locality awareness*).
Proces modelowania aplikacji w języku CAMEL w ramach platformy Melodic w pierwszej kolejności obejmuje określenie jej komponentów, połączeń między nimi, a także wymagań odnośnie do wydajności i zasobów oraz sposobu wdrażania aplikacji. W dalszym kroku następuje automatyczne optymalizowanie konfiguracji wdrożenia – platforma „decyduje", jakiej i gdzie umieszczonej infrastruktury należy użyć. Wstępna optymalizacja jest dokonywana na podstawie parametrów określonych przez użytkownika.
Na podstawie konkretnej, optymalnej konfiguracji automatycznie tworzona jest ściśle zdefiniowana infrastruktura u wybranych dostawców usług chmurowych (maszyny wirtualne o ustalonych parametrach) i wdrażana jest aplikacja, wraz z ustawieniami połączeń między komponentami. Po wdrożeniu aplikacja jest monitorowana – zbierane są m.in. metryki

określające charakterystykę jej działania, które jednocześnie stanowią podstawę do automatycznej optymalizacji wdrożenia aplikacji uruchomionej w ramach platformy Melodic.

AUTOMATYZACJA TO PRZYSZŁOŚĆ CHMURY

Rozwiązania chmurowe stopniowo stają się powszechne również w Polsce. Pozwalają bowiem na znaczące obniżenie nakładów finansowych przeznaczanych na rozwiązania IT, zarówno na poziomie sprzętowym (brak konieczności posiadania serwerowni i sprzętu), jak i kompetencyjnym (istotna część zadań wykonywana jest przez dostawcę usług chmurowych). Właściwe wykorzystanie tego potencjału wymaga jednak wiedzy o możliwościach i ryzykach wynikających z zastosowania zasobów chmurowych na dużą skalę oraz umiejętności dostosowania zasobów i aplikacji do potrzeb danej organizacji.
Podejście wdrożone w platformie Melodic to przyszłość rozwiązań cloud computing.
W szczególności użycie zaawansowanych algorytmów machine learning pozwala optymalizować wdrażanie aplikacji, co dla dużych systemów może mieć bardzo istotne znaczenie, zarówno co do wydajności, jak i kosztów.

MULTICLOUD A POUFNOŚĆ DANYCH

W ślad za ideą wynoszenia zasobów i wykonywanych na nich operacji do środowisk zewnętrznych, silniej wyartykułowano konieczność zwiększenia poziomu ochrony danych, uznawanych za wrażliwe czy szczególnie wartościowe dla ich właściciela bądź użytkownika. W kontekście rozporządzenia o ochronie danych osobowych (RODO), szczególnie zwrócono uwagę na zagrożenie upubliczniania poufnych informacji (powierzanych przecież w ramach *cloud computing* opiece obcych zespołów,) oraz prawne oraz ekonomiczne następstwa naruszenia zasad bezpieczeństwa.
Trudnym i złożonym zagadnieniem jest zapewnienie odpowiedniego poziomu poufności danych subskrybenta po stronie dostawców chmur obliczeniowych w sytuacji, gdy są tam one przechowywane i przetwarzane w postaci jawnej. Nie sposób wtedy skutecznie rozdzielić uprawnień administratorów technicznych odpowiadających za poprawność działania usługi i jej interakcję z teleinformatyczną infrastrukturą dostawcy od uprawnień administratorów biznesowych (zarządzających z ramienia subskrybenta usługą i dostępem do niej) oraz użytkowników, którzy jako jedyni powinni mieć dostęp do treści przetwarzanej informacji. Niełatwo ustalić odpowiedzialność, w przypadku ujawnienia informacji chronionej. Związane z tym ryzyko może, w przypadku przetwarzania informacji o poufności krytycznej z punktu widzenia żywotnych interesów usługobiorcy, osiągnąć poziom nieakceptowalny. Przechowywanie na serwerach usługi niezaszyfrowanej informacji oznacza również znaczne ryzyko utraty przez nią waloru autentyczności i integralności: mniej lub bardziej swobodny niskopoziomowy[v] dostęp do danych, częstokroć z pominięciem logowania takich operacji lub ich logowaniu w dziennikach niedostępnych dla subskrybenta, pozwala – jeśli aplikacje usługi nie wykorzystują intensywnie technik podpisu elektronicznego – na niewykrywalną lub trudno i poniewczasie wykrywalną nieautoryzowaną zmianę informacji lub wprowadzanie informacji fałszywej. Trudno też w takiej sytuacji polegać na wbudowanych w aplikację usługową mechanizmach zapewniania rozliczalności i niezaprzeczalności różnych operacji.

MINIMALNE WYMAGANIA BEZPIECZEŃSTWA

Z wymienionych powyżej względów za absolutne minimum wymagań należy uznać przechowywanie przez usługodawcę powierzonej mu informacji wyłącznie w postaci zaszyfrowanej odpowiednio mocnymi standardowymi algorytmami, przy czym wymaganie to musi być spełnione zarówno w przypadku baz danych, jak i przechowywanych poza strukturami baz oddzielnych plików (np. skanów dokumentów, dzienników operacji itp.) oraz kopii zapasowych wszystkich tych danych. Istnieje wiele technik oraz wariantów implementacji powyższego wymagania i nie miejsce tu na ich analizę i porównywanie.

Jako przykładowe rozwiązania po stronie dostawcy usług można zalecić stosowanie przezeń techniki określanej jako *Transparent Data Encryption* (dostępnej m.in. w bazach danych firmy Oracle i Microsoft SQL Server) oraz szyfrowania całych nośników – dysków, macierzy dyskowych itp. W każdym z tych przypadków musi być zagwarantowane przez dostawcę również szyfrowanie kopii zapasowych (baz i nośników), co może wymagać po jego stronie oddzielnych rozwiązań.

W przypadku przetwarzania informacji o skrajnie wysokiej wrażliwości[vi] (w tym poufności) może się okazać, że odpowiedni poziom jej bezpieczeństwa mogą zapewnić jedynie rozwiązania, które można określić terminem *client-side encryption*. Szyfrowanie i odszyfrowywanie danych jest w ich przypadku realizowane po stronie usługobiorcy przy użyciu kluczy dostępnych jedynie dla zalogowanego użytkownika, mającego przypisany odpowiedni poziom autoryzacji. Rozwiązania takie – choć godne szczególnego polecenia – są aktualnie bardzo mało rozpowszechnione, między innymi z powodu istotnych trudności, wiążących się z zapewnieniem odpowiedniej do praktycznych zastosowań efektywności operacji wyszukiwania w zaszyfrowanych bazach danych. Jako przykłady takiego podejścia można wskazać rozwiązania firmy Thales e-Security (Vormetric Data Security Platform) oraz oferowane m.in. przez firmę 7bulls.com krajowe rozwiązanie o handlowej nazwie SiC (Safely in the Cloud).

Nie należy przy tym sądzić, że zastosowanie rozwiązania klasy *client-side encryption* uwalnia całkowicie subskrybenta od ryzyka ujawnienia chronionej informacji niepowołanym podmiotom lub jej nieautoryzowanej modyfikacji – jest ono jednak wtedy praktycznie całkowicie przeniesione do własnego środowiska subskrybenta, nad którego bezpieczeństwem panuje lub panować powinien.

MULTICLOUD A TRWAŁY NOŚNIK

Zasygnalizowane powyżej rozwiązania klasy *client-side encryption* bazują zazwyczaj na koncepcji tzw. pęku kluczy – każda informacja jest zaszyfrowana zestawem (pękiem) kluczy kryptograficznych, będących w dyspozycji autoryzowanych jej użytkowników. Usunięcie z takiego pęku klucza jakiegoś użytkownika oznacza utratę przezeń dostępu do odpowiedniej porcji informacji.

Przechowywane w repozytorium banku i udostępniane klientom przeznaczone dla nich dokumenty (w tym spersonalizowane) mogą stać się nieusuwalnymi przez bank przed upływem wynikającego z przepisów i umów terminu również wtedy, gdy zostaną zaszyfrowane kluczem znanym jedynie klientowi i pozbawione cech pozwalających przyporządkować je do konkretnego klienta – np. przez nadanie im losowej nazwy niezapamiętywanej w kartotece danego klienta.

Depozytariusz tak opracowanych dokumentów będzie pozbawiony możliwości selektywnego ich usuwania na podstawie kryteriów innych niż wymagany czas retencji. Tym samym skuteczne usunięcie lub podmienienie dokumentów pojedynczego klienta wymagałoby usunięcia całej zawartości takiego repozytorium, na co z oczywistych względów żadna instytucja nie może sobie pozwolić.

Kompleksowa aranżacja metodyki opracowywania i przechowywania dokumentów i danych w połączeniu z umiejętnym wykorzystaniem dostępnych metod zarządzania kluczami użytymi do szyfrowania informacji wrażliwych, rodzi możliwość relatywnie taniego i prostego w implementacji rozwiązania problemu tzw. trwałego nośnika.

---

[i] https://www.techopedia.com/definition/26802/vendor-lock-in

[ii] http://searchcloudcomputing.techtarget.com/definition/TOSCA-Topology-and-Orchestration-Specification-for-Cloud-Applications

[iii] http://camel-dsl.org/

[iv] http://melodic.cloud/

[v] tj. bez pośrednictwa warstwy aplikacji, z wykorzystaniem narzędzi bezpośredniego dostępu do systemu plików, operacji w języku SQL itp.

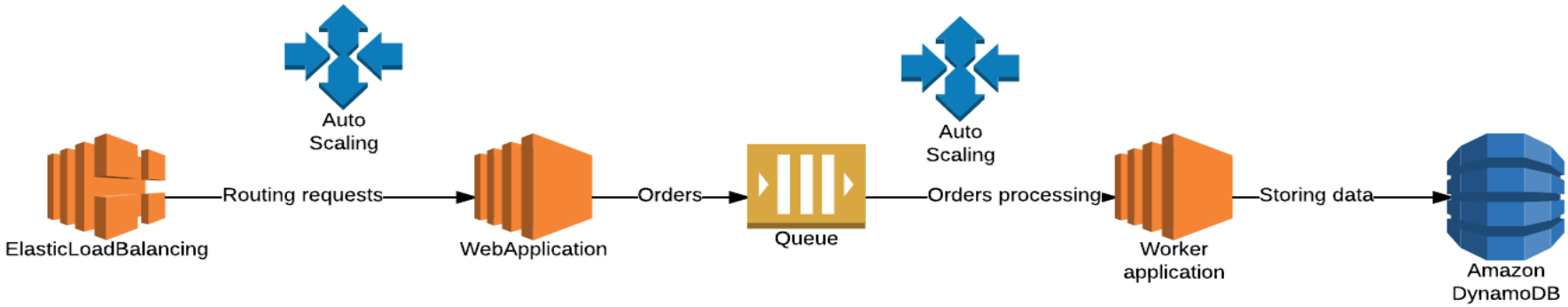
[vi] W sensie nadanym temu terminowi przez odpowiednią normę z serii ISO/IEC 27000.

Cloud Computing
From AWS to MELODIC

Katarzyna Materka

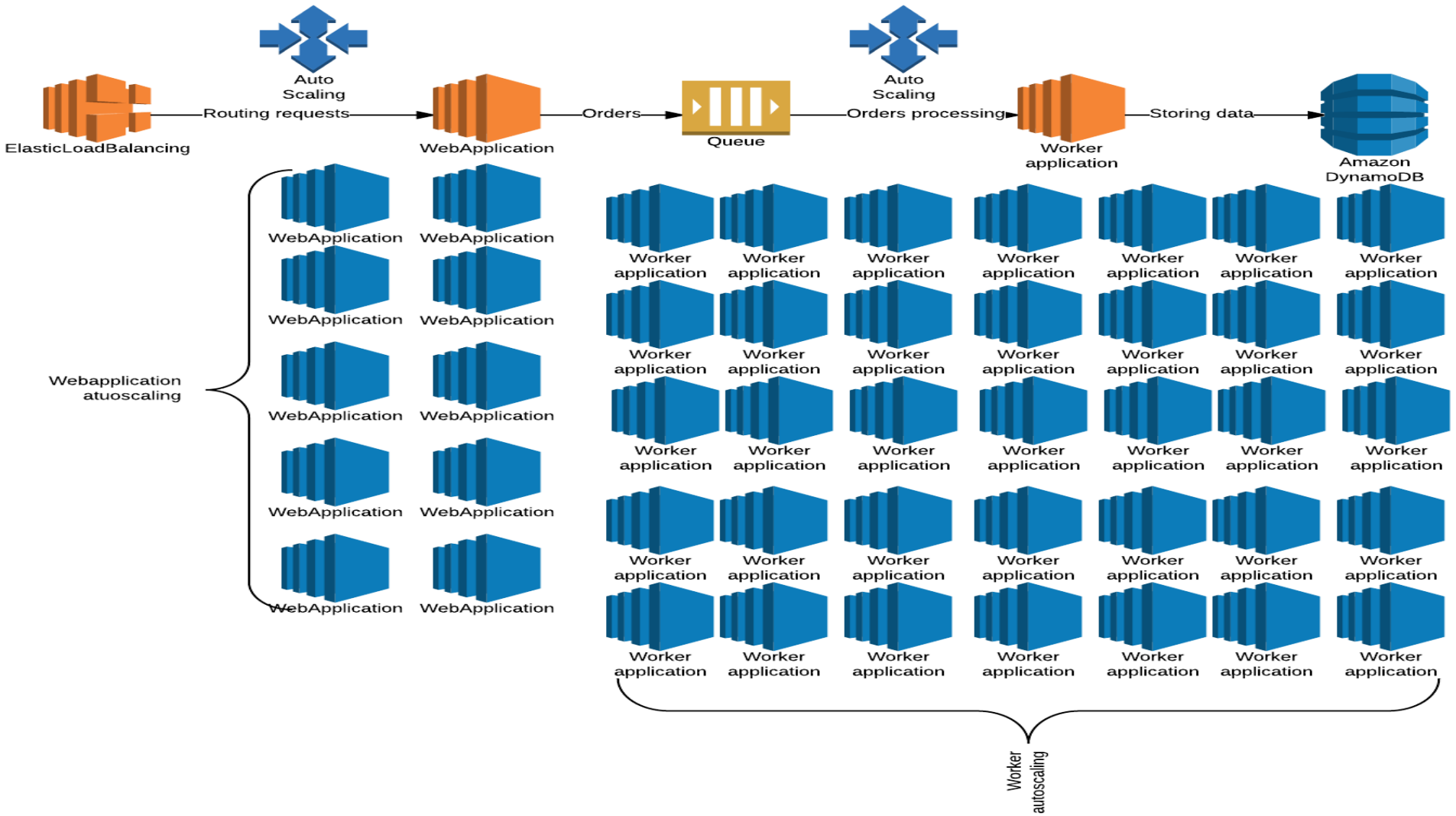
Paweł Skrzypek

Architecture of image processing system



Processing around 5000 requests per hour by one worker instance

Architecture of image processing system



Processing 175,000 requests per hour after autoscaling, in total 140 core, 560GB RAM. Configuration time: 5-10 minutes.

Everybody's doing cloud –
but what is it, how to do it and,
most of all, what for?

Cloud Computing? Cloud? What and how?

- SaaS
- IaaS
- PaaS
- CaaS
- FaaS
- StaaS
- And others

- Virtual machines
- VMWare
- Docker
- Kubernetes
- OpenStack
- OpenShift
- And others

- AWS
- Azure
- Google Cloud
- Heroku
- Hetzner
- Aruba
- And others

- EC2
- Dynos
- SmartCont
- EBS
- Lambda
- VPS
- Bare metal
- And others

What will I gain?

1. **Understanding of the cloud**
2. Knowledge on how to rationally **seize opportunities**
3. Debunking cloud myths

?

Cloud Computing - myths

1. Security
2. Data (esp. confidential)
3. Vendor lock
4. Hybrid solutions
5. Mythological marketing

Basic concepts – before we get to the cloud

1. **Basic concepts**
2. What is Cloud Computing
3. Different types and models Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use it rationally
8. Cloud Computing – trends and challenges

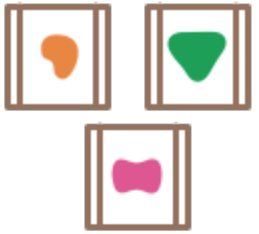
Before we get to the cloud

1. Microservices vs SOA architecture
2. Virtualization
3. Containers
4. Platform for applications
5. Services orchestration
6. DevOps

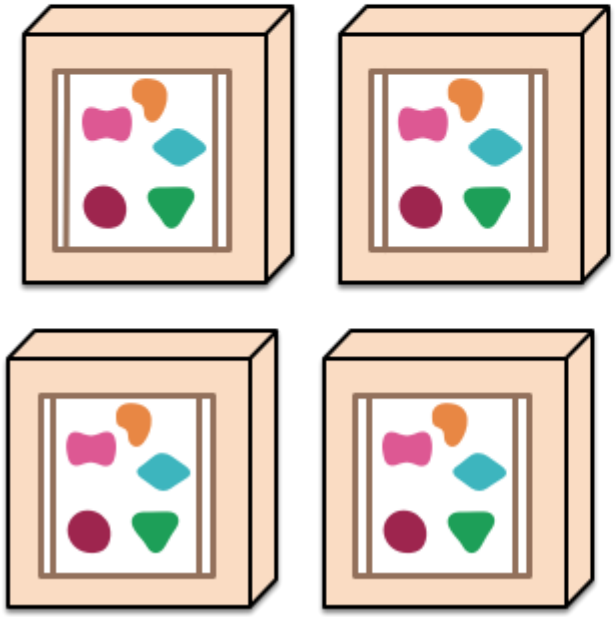
Microservices architecture

A monolithic application puts all its functionality into a single process...

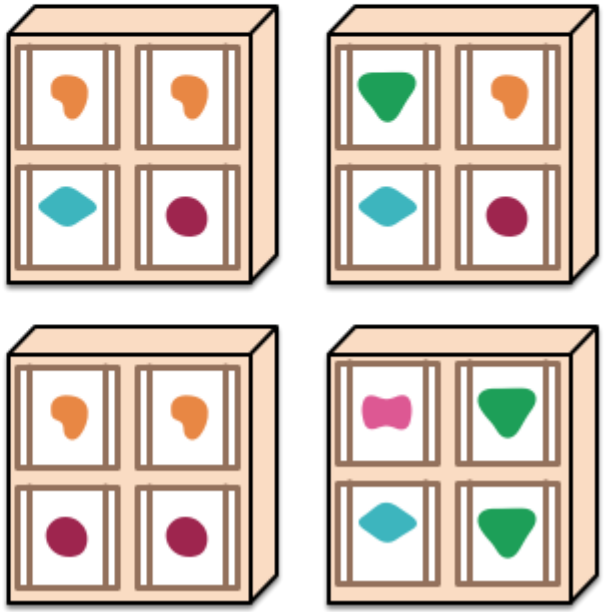
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers



... and scales by distributing these services across servers, replicating as needed.



Microservices are not SOA

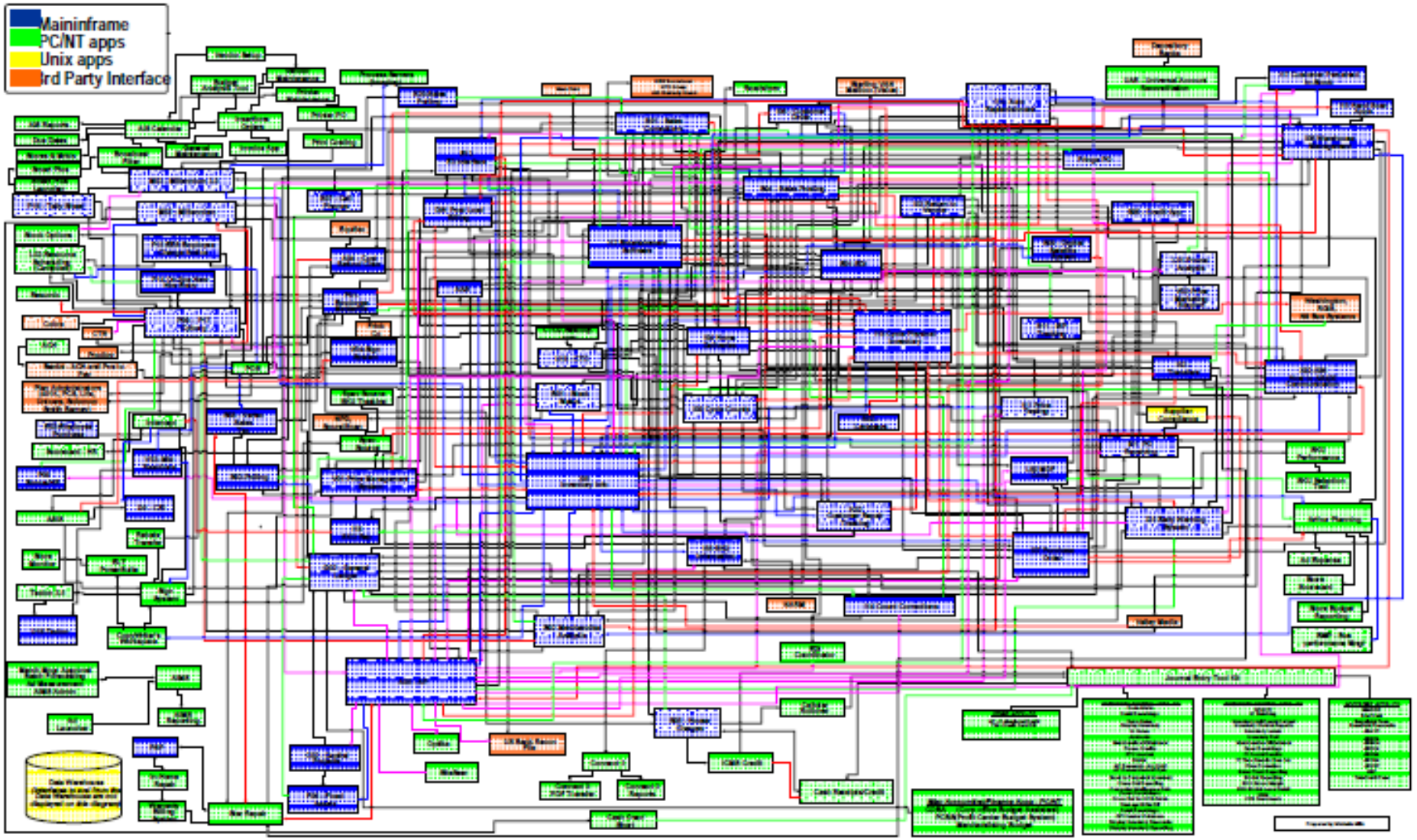
Microservices

- One service = one component
- Independent components
- Replicable and distributable components
- Small-sized components

SOA

- Many services from one monolithic system
- Dependent services – one system
- Irreplicable and non-distributable
- Monolithic system behind the services (usually)

Microservices? Disadvantages



Hardware virtualization

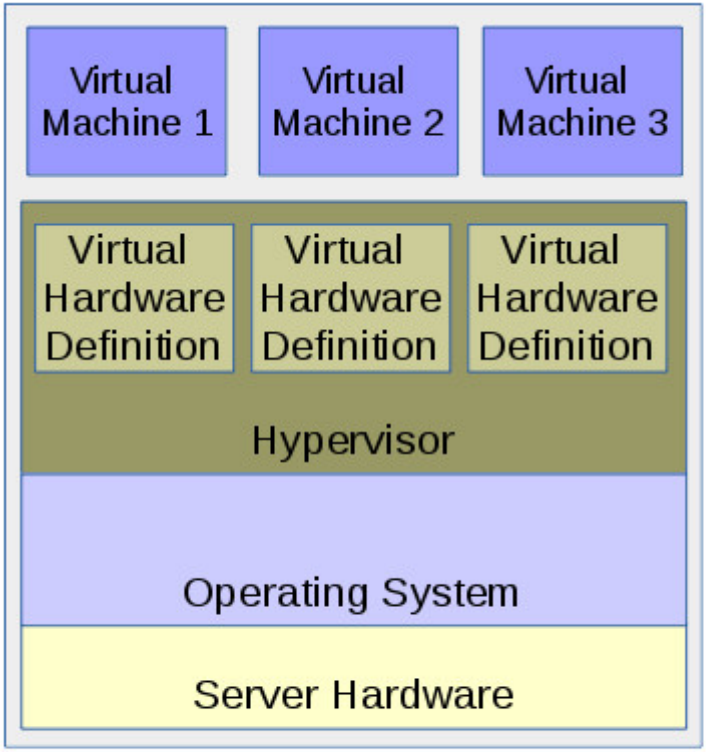
Running **many virtual systems** on one hardware server, with allocation of processing units, memory and other resources.

Advantages:

- **full control** and mapping.

Disadvantages:

- Difficult to manage, transfer and modify - **big images.**



Hardware virtualization

Containers (OS virtualization)
Application delivery

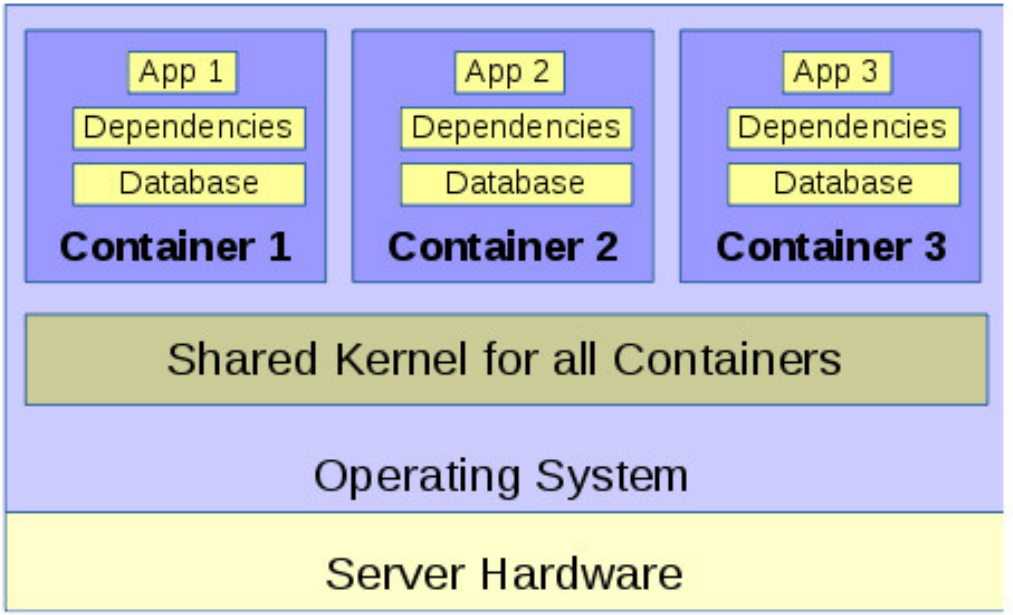
Running **many container systems** on one server, with possibility of allocating processing units, memory and other resources.

Advantages:

- **Easy to use,** relatively small size.

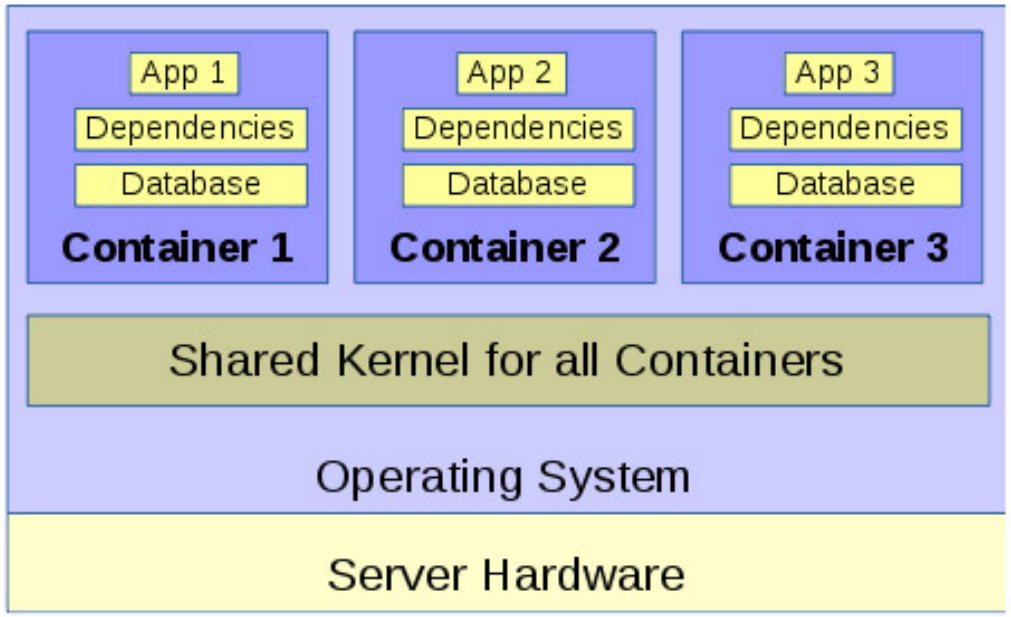
Disadvantages:

- **No full control** (shared kernel).

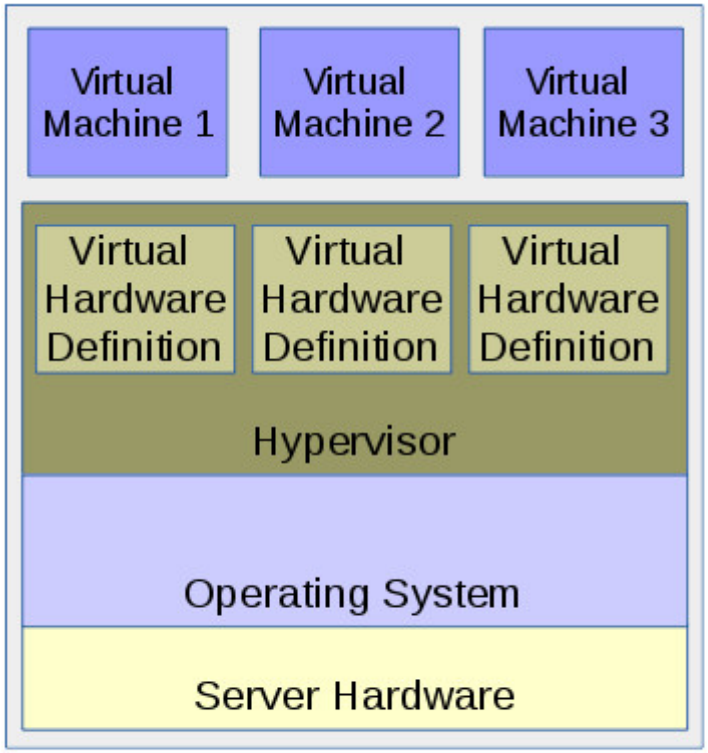


OS level virtualization

Containers and full virtualization



OS level virtualization



Hardware virtualization

Application platform

- Platform enabling running many application components and other system elements **without VM/container overhead**
- Application servers – **and why they no longer exist**
- POJO and Spring as well as other languages
- What next?

Orchestration of system components

Management of deployment of particular components in an ordered way:

- Application
- Infrastructure

Monitoring and optimization of:

- Security
- Logging and debugging

Particularly important in clouds!

DevOps

Practice that aims at unifying software development (Dev) and operation (Ops) as well as quality assurance in order to ensure **most efficient delivery of new software releases.**

- One of the most corrupted approaches in the IT world
- **One team**
- Proper understanding of goals

SQL vs NoSQL databases

- **Growing importance of NoSQL**, perfect for processing large sets of distributed data
- **NoSQL is natural for Cloud Computing**
- Necessity of using relational databases, transactions
- **Hybrid solutions** in regard to data storage

Cloud Computing – types and models

1. Basic concepts
2. What is Cloud Computing
3. **Types and models of Cloud Computing**
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use it rationally
8. Cloud Computing – trends and challenges

What is cloud and what are its types

Cloud Computing is a **possibility of using infrastructure** (servers, disk drives, networks etc.) located in the server room of cloud service provider.

Its main types are:

- **IaaS** – Infrastructure as a Service
- **CaaS** – Container as a Service
- **PaaS** – Platform as a Service
- **PaaS serverless** (FaaS)
- **SaaS** – Software as a Service

IaaS – Infrastructure as a Service

Sharing **of selected IT infrastructure elements** in a cloud model using virtualized resources.

Typical components provided in this model:

- **Virtual servers** (VM/VPS)
- Mass memory (disks)
- Networking elements (load balancing, routing)

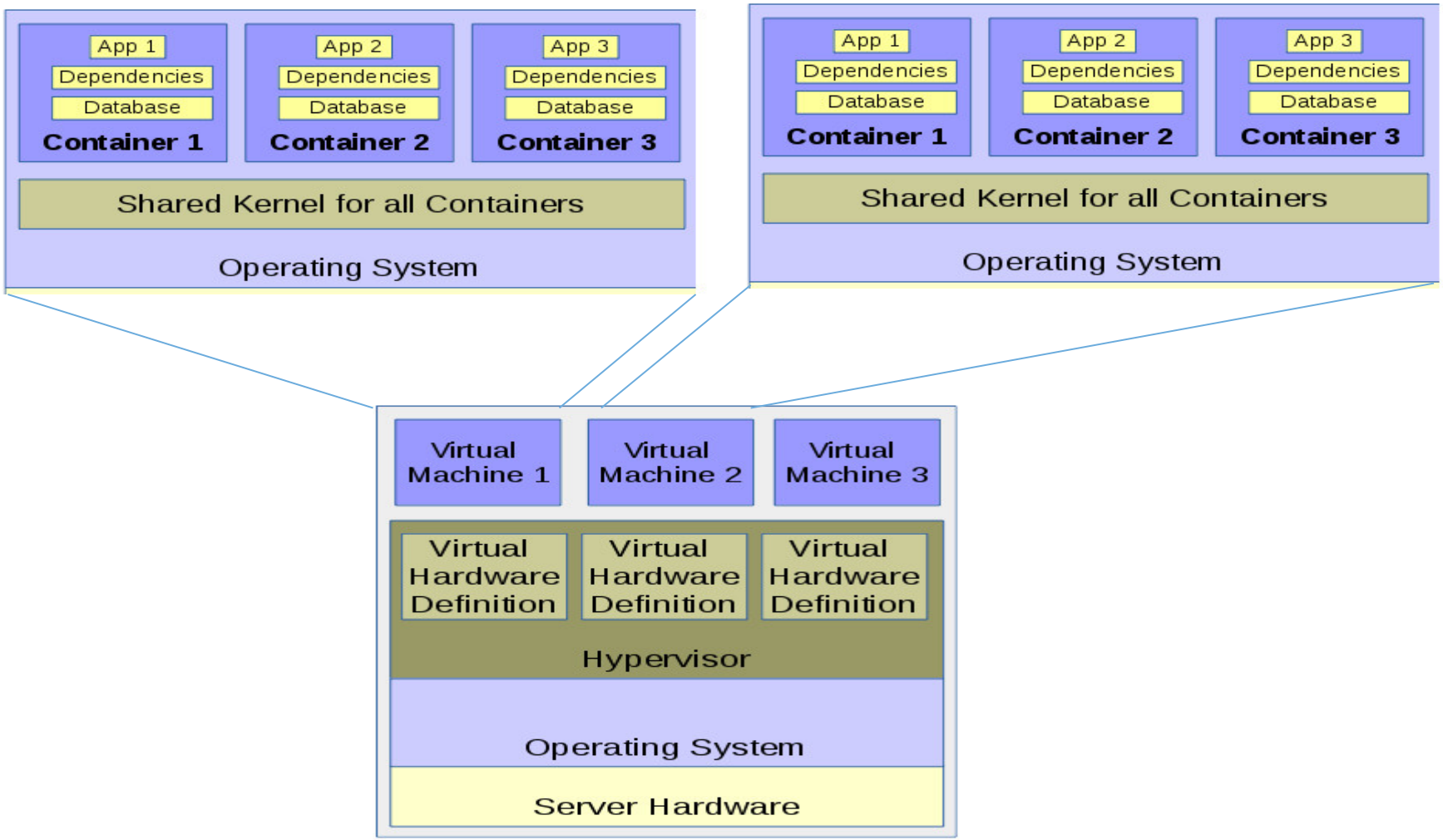
CaaS – Container as a Service

CaaS – Container as a Service – CaaS services provider offers a platform that enables **running containers** according to given parameters.

We do not buy particular services such as servers or disks, but the service of a platform for containers and the elements of infrastructure are operated by the platform provider.

Everybody loves DOCKER!

CaaS – disadvantages



Hardware virtualization

PaaS – Platform as a Service

PaaS – Platform as a Service – PaaS services provider offers a platform that enables **running applications** according to given parameters.

Thanks to that we do not have to buy particular services such as servers or disks as in IaaS model, but we buy **a service of platform for application** and the elements of infrastructure are operated by the platform provider.

PaaS serverless (FaaS)

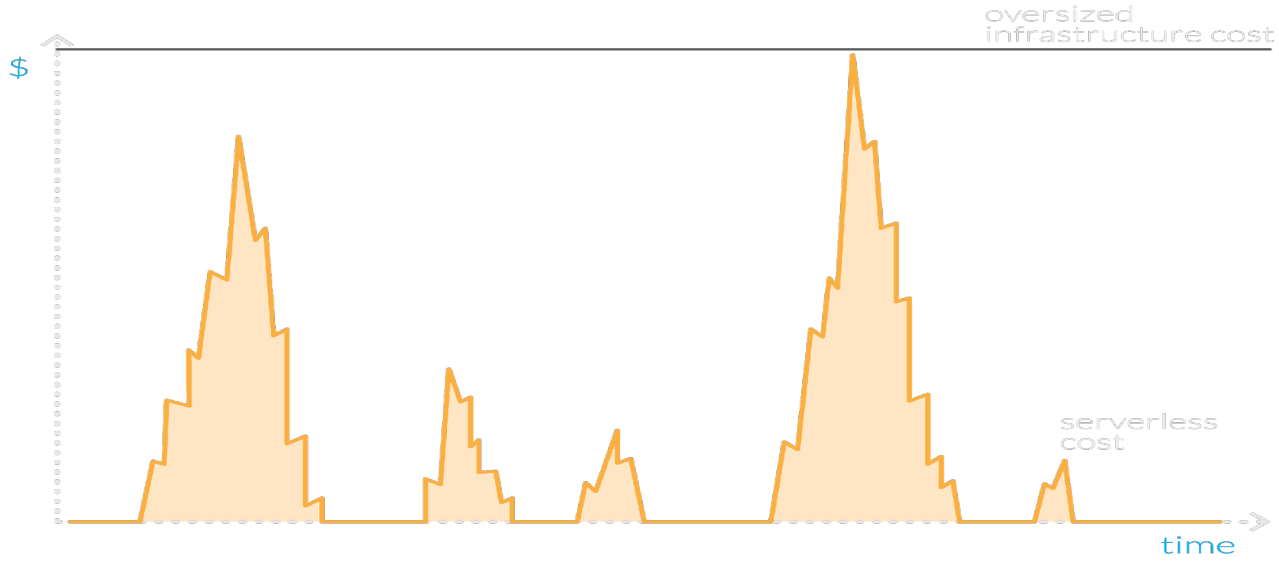
PaaS service can be used in a serverless version, meaning that the **payment is made only for the number of times the application is run or its functions are called.**

Thanks to that we can acquire **highest cost efficiency**, because we pay for the actual usage of system - we do not pay if we do not use the system.

PaaS serverless - advantages

Same as PaaS plus:

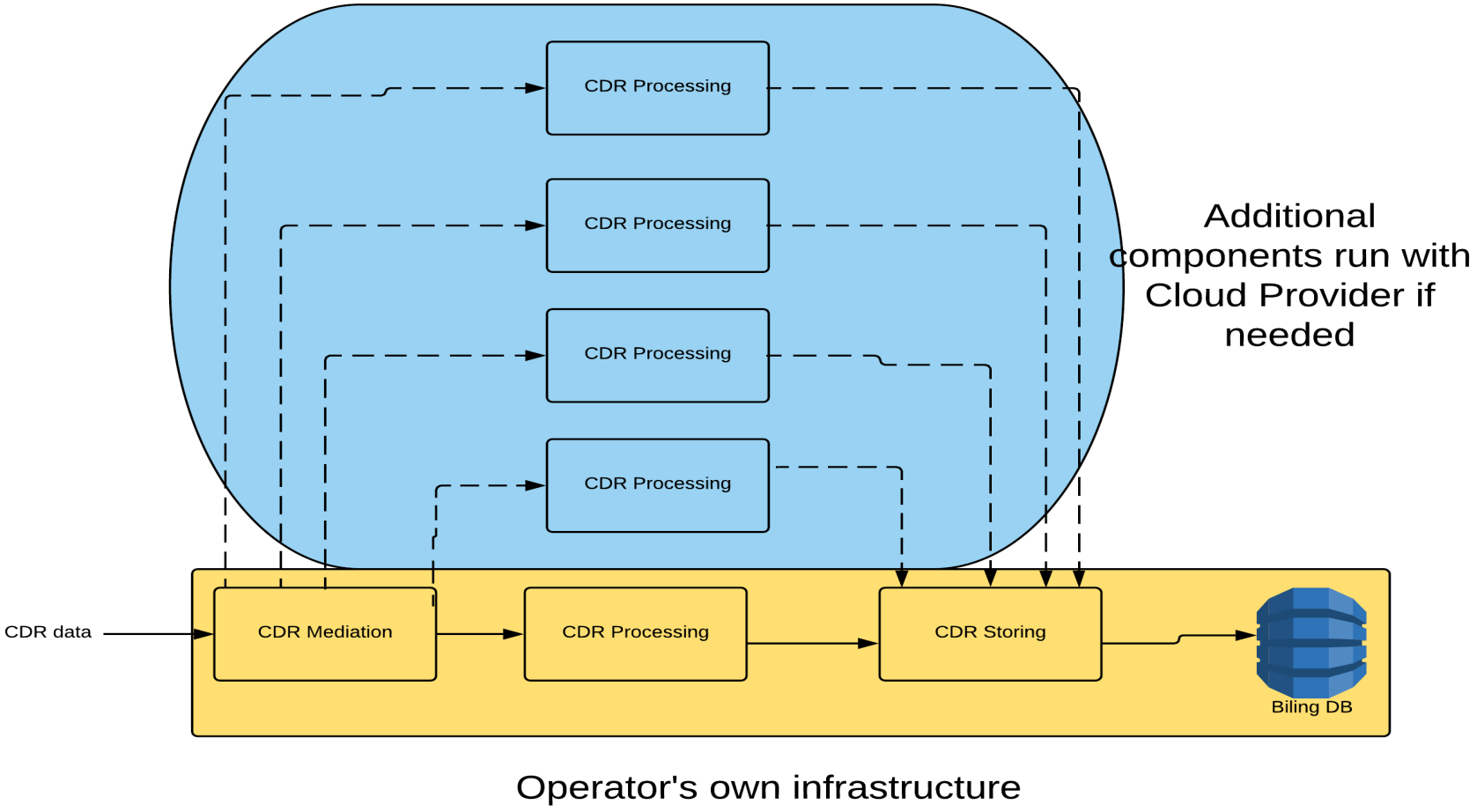
- **Pay-as-you-go pricing** - we do not pay if we do not use the system.
- **Easier scaling** to multiple instances – system construction forces adjustment to scaling.



PaaS serverless - disadvantages

- Platform limitations.
- **Difficult development** and deployment.
- **Difficult management** and monitoring.
- New service on the market – half-baked, **only one commercial implementation.**

Serverless – Biling use case



SaaS – what is it?

Software as a Service – **sharing the app as a whole in cloud,** through an Internet access.

- It is not **Cloud Computing**, more of a payment model.
- Proper understanding of justification of its use.

Cloud Computing – orchestration – how it works

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. **Cloud Computing - orchestration**
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use rationally
8. Cloud Computing – trends and challenges

Orchestration in cloud - solutions

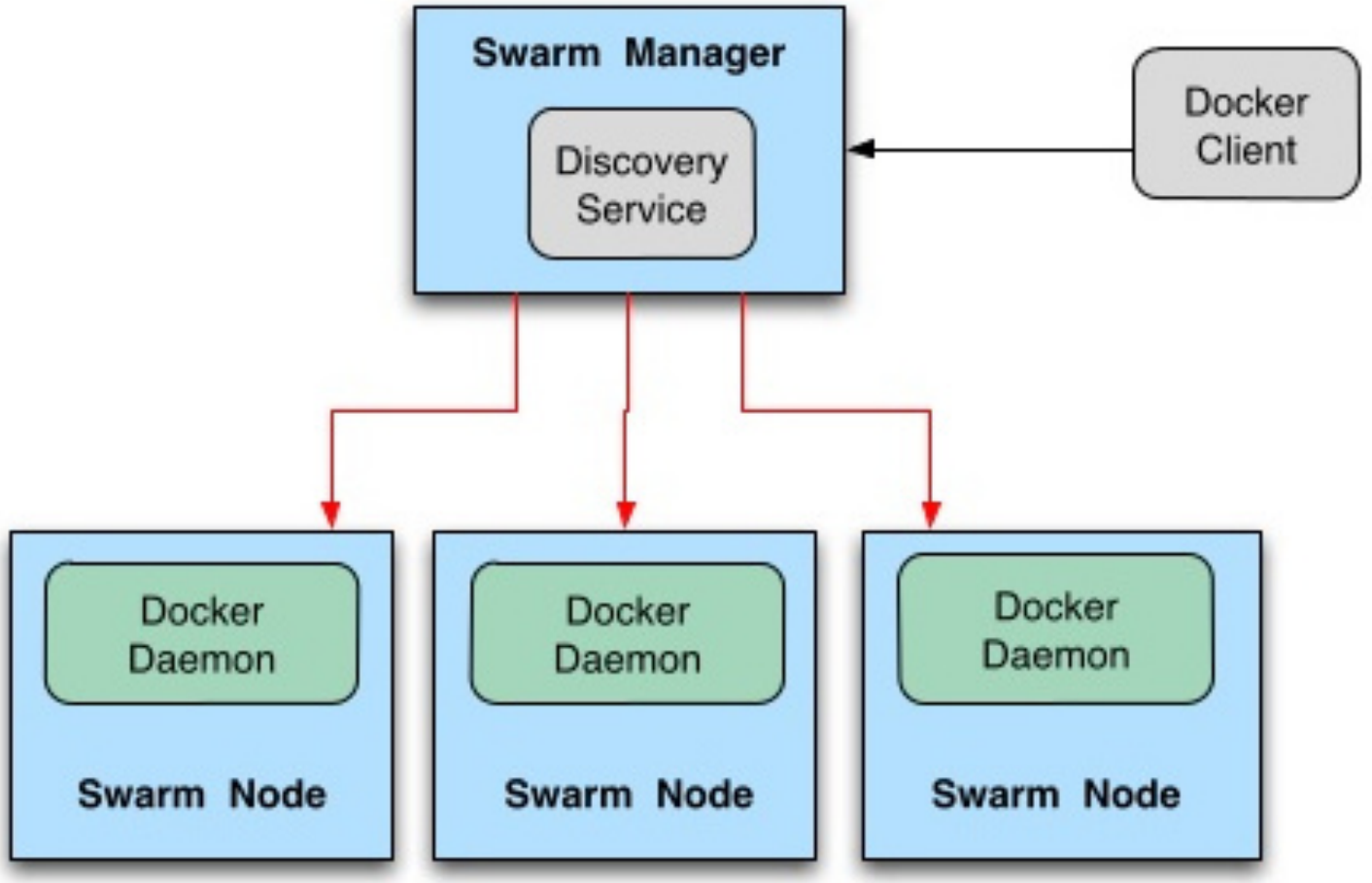
- Open Stack
- Kubernetes
- Docker Swarm
- TerraForm
- Tosca/Cloudify
- Heat

Docker Swarm – container cluster

Docker's native orchestration service system

- Managing a pool of hosts with containers using one controller.
- Managing the rules of distributing the containers in hosts.

Swarm Architecture



Docker Swarm – advantages and disadvantages

Advantages

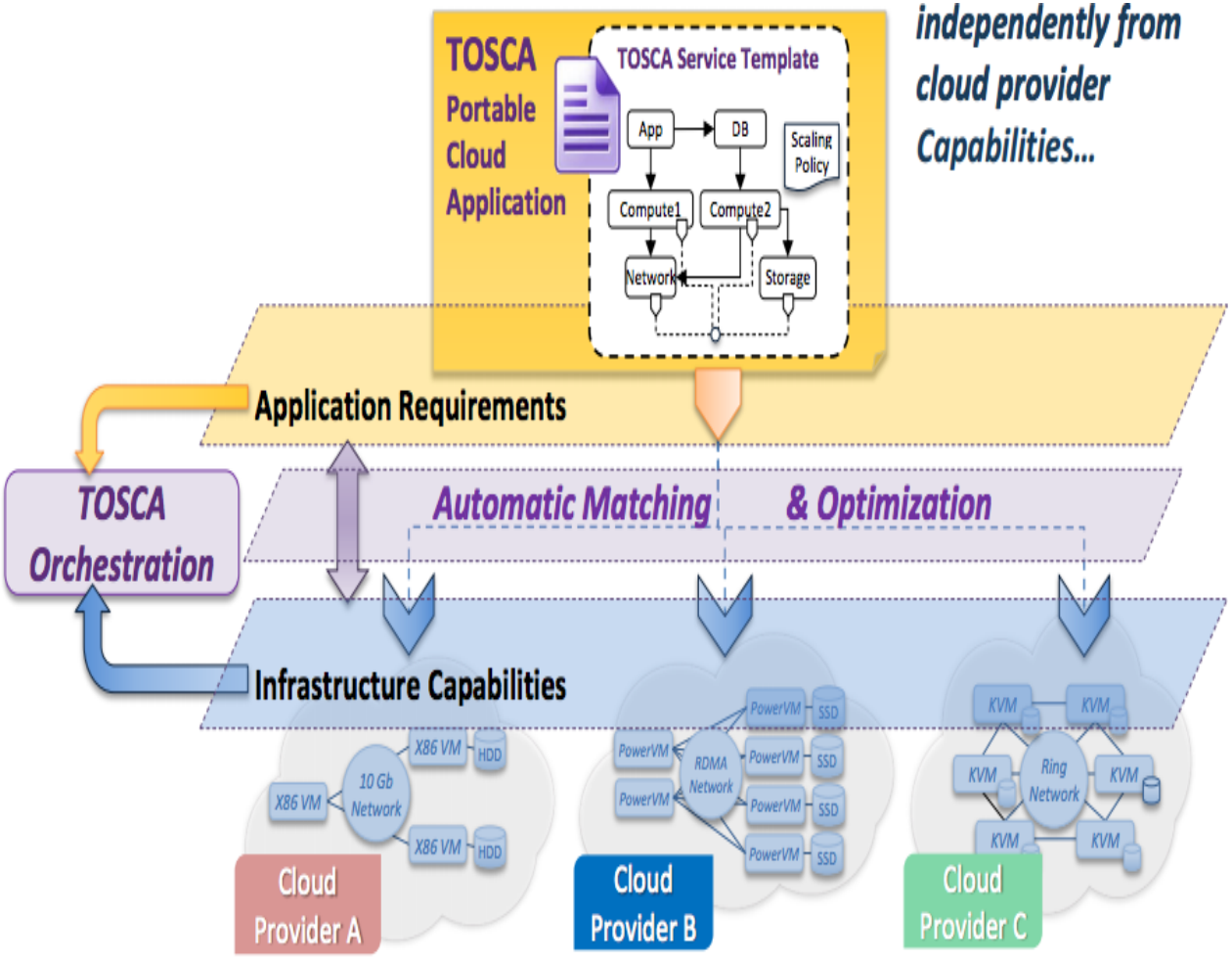
- Easy to use with Docker,
- possibility of flexible management of container location.

Disadvantages

- First version,
- manual configuration of multiple instances,
- no deployment process,
- lack of monitoring, automatic scaling etc.
- does not support multiple cloud providers.

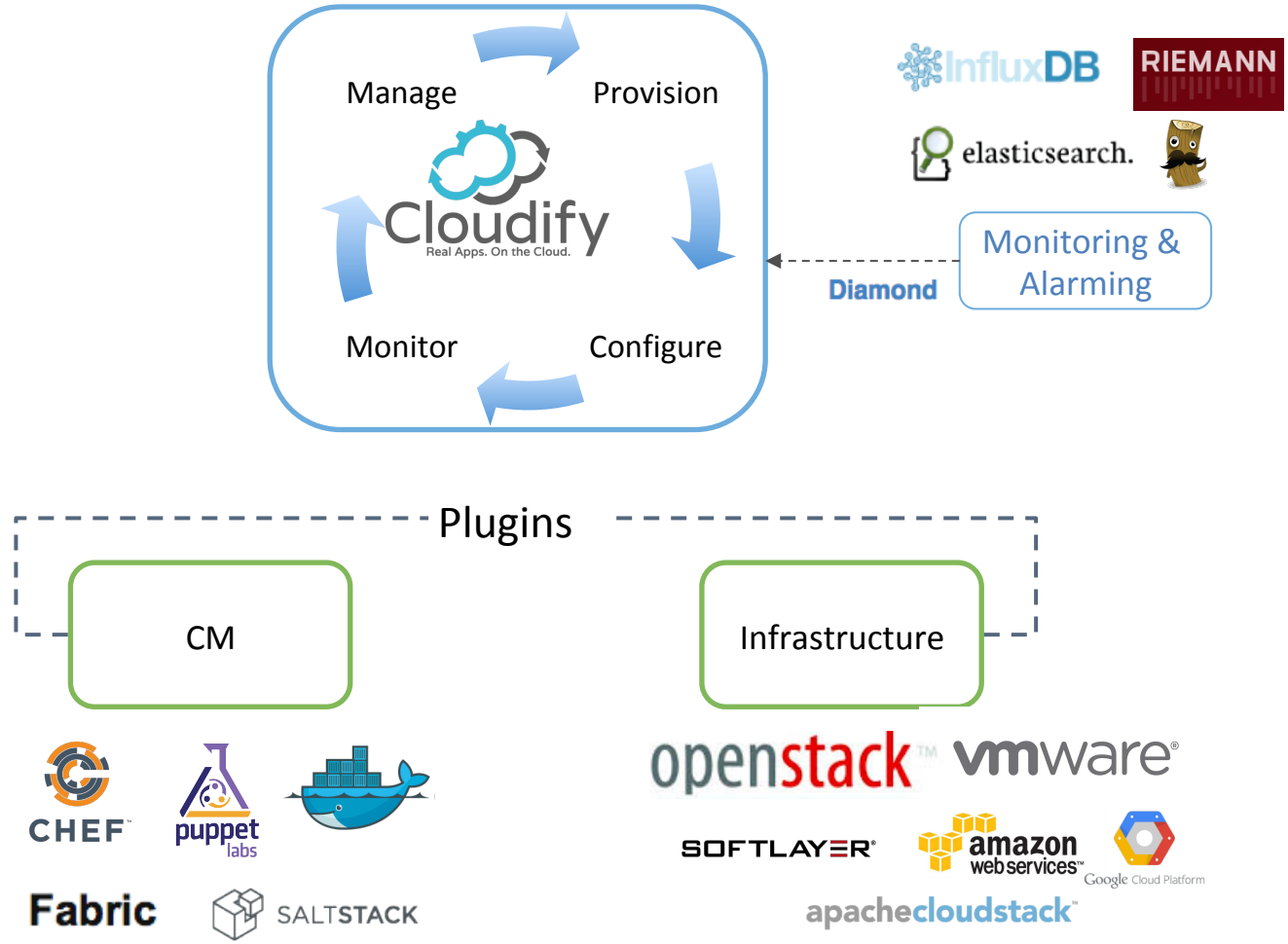
TOSCA

Topology and Orchestration Specification for Cloud Applications



Comprehensive definition of cloud architecture for applications and components with relations, dependencies, requirements and possibilities through DSL

Cloudify – open source version of TOSCA



Summary of Cloudify/TOSCA

Advantages

- Agnostic in terms of infrastructure and frameworks
- Full architecture lifecycle
- Supports infrastructure and applications
- Full scope of orchestration:
  - Monitoring
  - Deployment process
  - Deployment policy
  - Logging

Disadvantages

- Still under construction
- Lack of dynamic scaling management and HA
- Limited set of tools

MELODIC

Multicloud **E**xecution-ware for **L**arge-scale **O**ptimized **D**ata **I**ntensive **C**omputing

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. **MELODIC**
6. Practical overview of selected solutions
7. Cloud Computing – how to use rationally
8. Cloud Computing – trends and challenges

MELODIC

1. One universal platform for installation, running and management of applications in cloud environment.

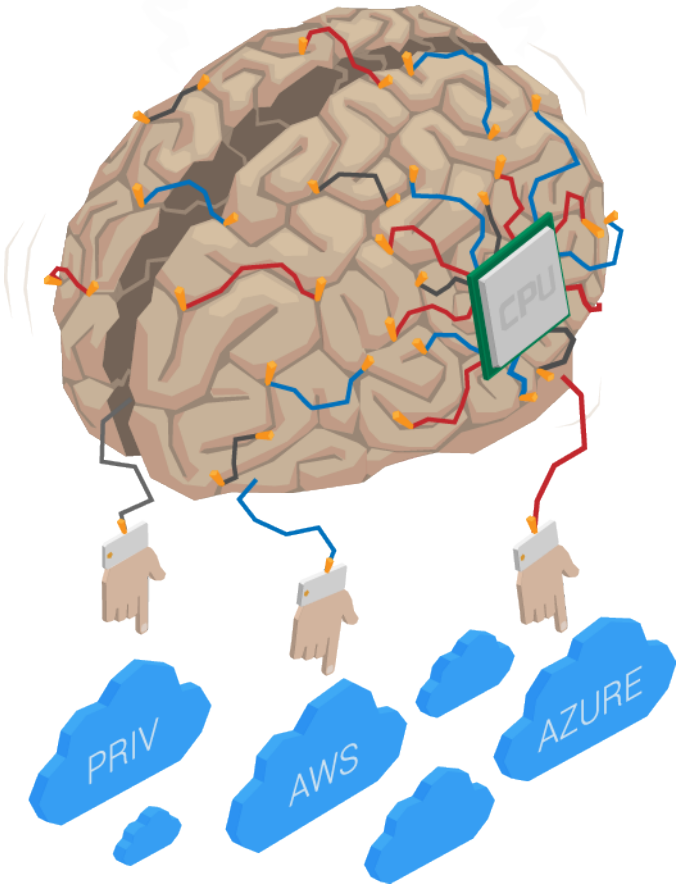
2. Possibility of application migration between cloud providers.

3. Possibility of running application most optimally basing on its characteristics.

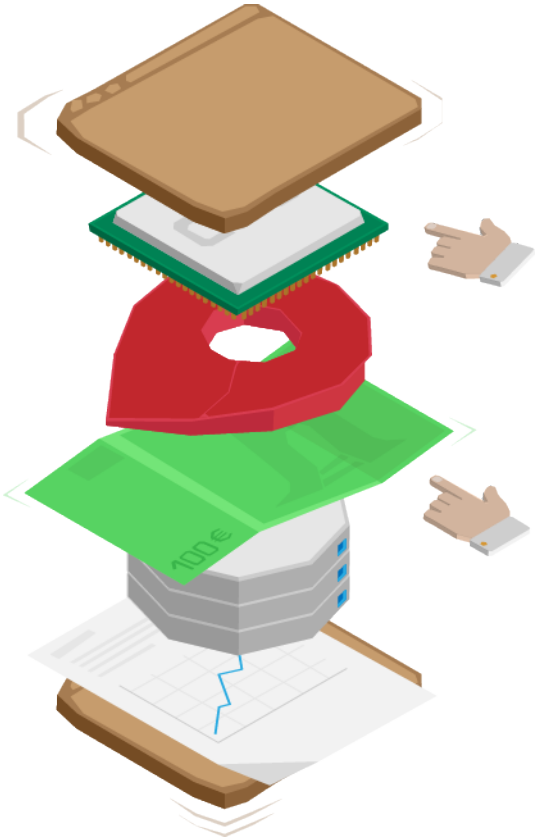
4. Security - running application in clouds of more than one provider.

5. No vendor – lock.

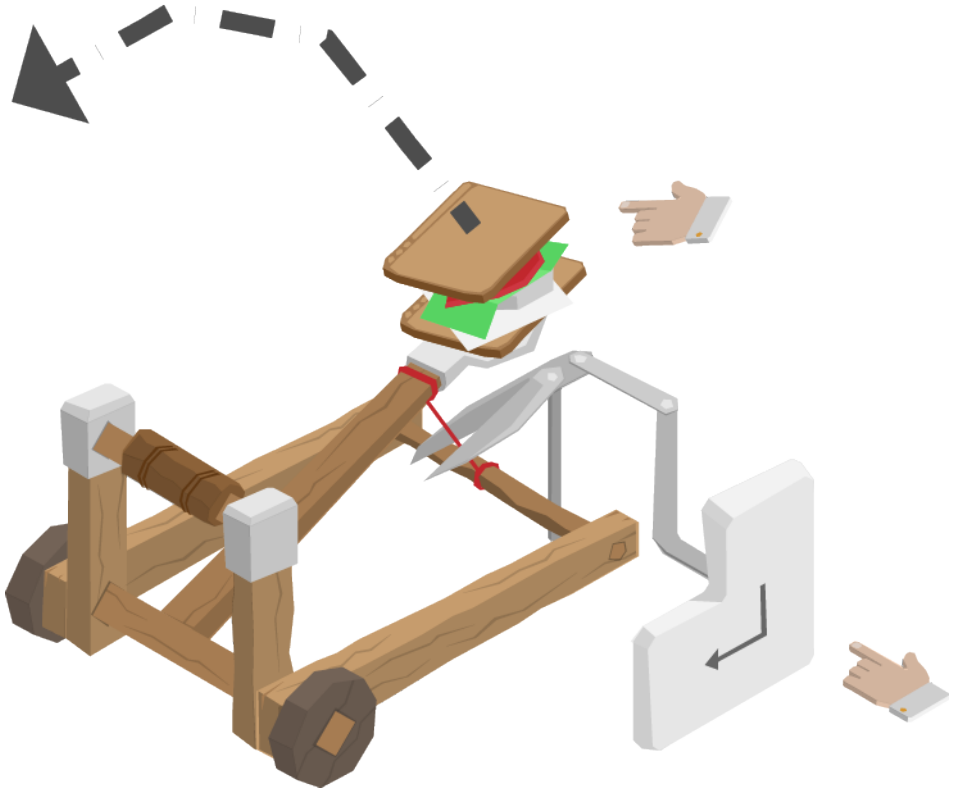
MELODIC - optimization



MELODIC – application and infrastructure



MELODIC – deployment



New application

Legacy application

Speculative profiler

Community expertise

CAMEL

- Provisioning and deployment requirements
- Scalability rules
- Service-level objectives
- Provider models

Design time optimisation loop

PaaSage Platform

Speculative profiler

Intelligent reasoner

Extra functional adaption

Metadata

Execution optimisation loop

Metadata collection

Execution control

Execution monitoring

Platform specific mapping

Execution environments

MELODIC – how to use it?

1. Installation of application on Melodic platform.
2. Choice of provider, whose cloud will be used to install the app.
3. Running and operation of application.
4. Analysis of application's operation .
5. Application's cost and efficiency optimization basing on its characteristics.
6. Migration to the most optimum cloud provider.

MELODIC – what can Asseco gain?

1. Unified platform for application deployment – uniform deployment instructions
2. Possibility of using multicloud and hybrid cloud – unique offer on the market
3. Optimization (incl. cost) of infrastructure and applications – especially important in cloud environments

MELODIC – status and 7bulls.com's role

1. The project was launched on 1/12/2016, duration: 3 years, first iteration after 12 months.
2. Whole system will be available as an open-source software.
3. **7bulls.com is responsible for Quality Assurance and Integration.**
4. We are looking for:
   - Cloud Services providers
   - Clients/users – early adopters

Practical overview of selected solutions

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. **Practical overview of selected solutions**
7. Cloud Computing – how to use it rationally
8. Cloud Computing – trends and challenges

Amazon Web Services

Currently the most complete cloud platform offering services of different types

Advantages:

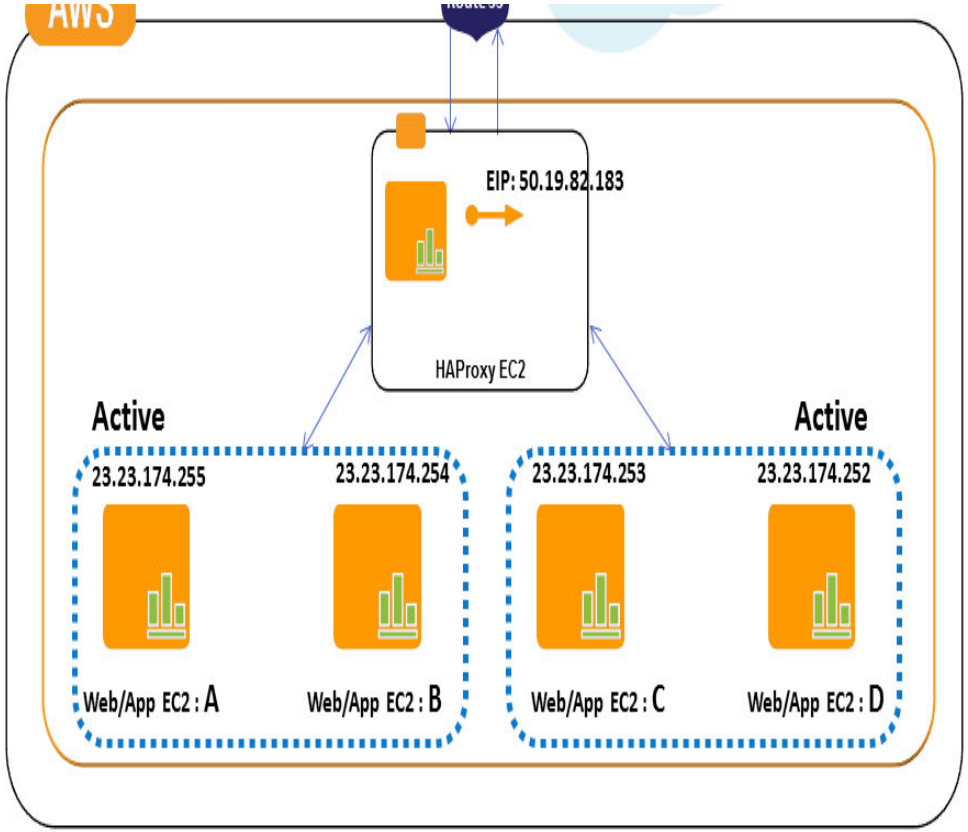
- wide choice of components,
- the only serverless platform,
- solutions for monitoring, logging, deployment,
- integrated access rights system,
- different types of autoscaling,
- hourly payment, Reserved Instances
- multiple datacenters.

Disadvantages:

- some solutions are non-optimal,
- solutions specific to AWS - strong vendor lock,
- prices.

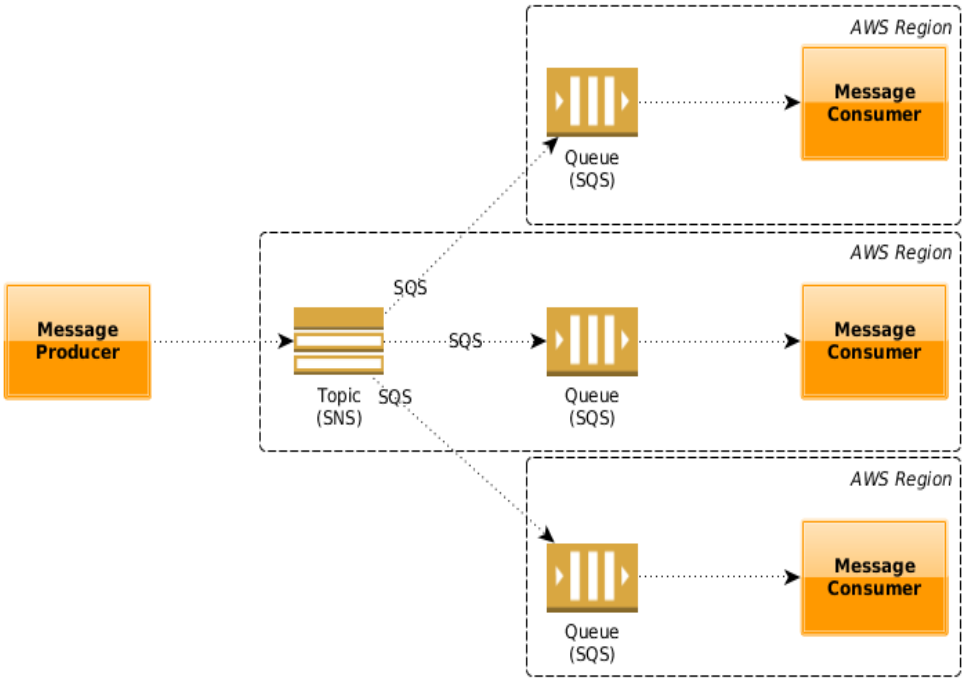
EC2 - virtual server

- IaaS on KVM.
- Creation of images and backup.
- Multiple types of machines available, including GPU.
- Autoscaling.
- Pay-per-minute pricing and Reserved Instances.
- Quick reconfiguration.

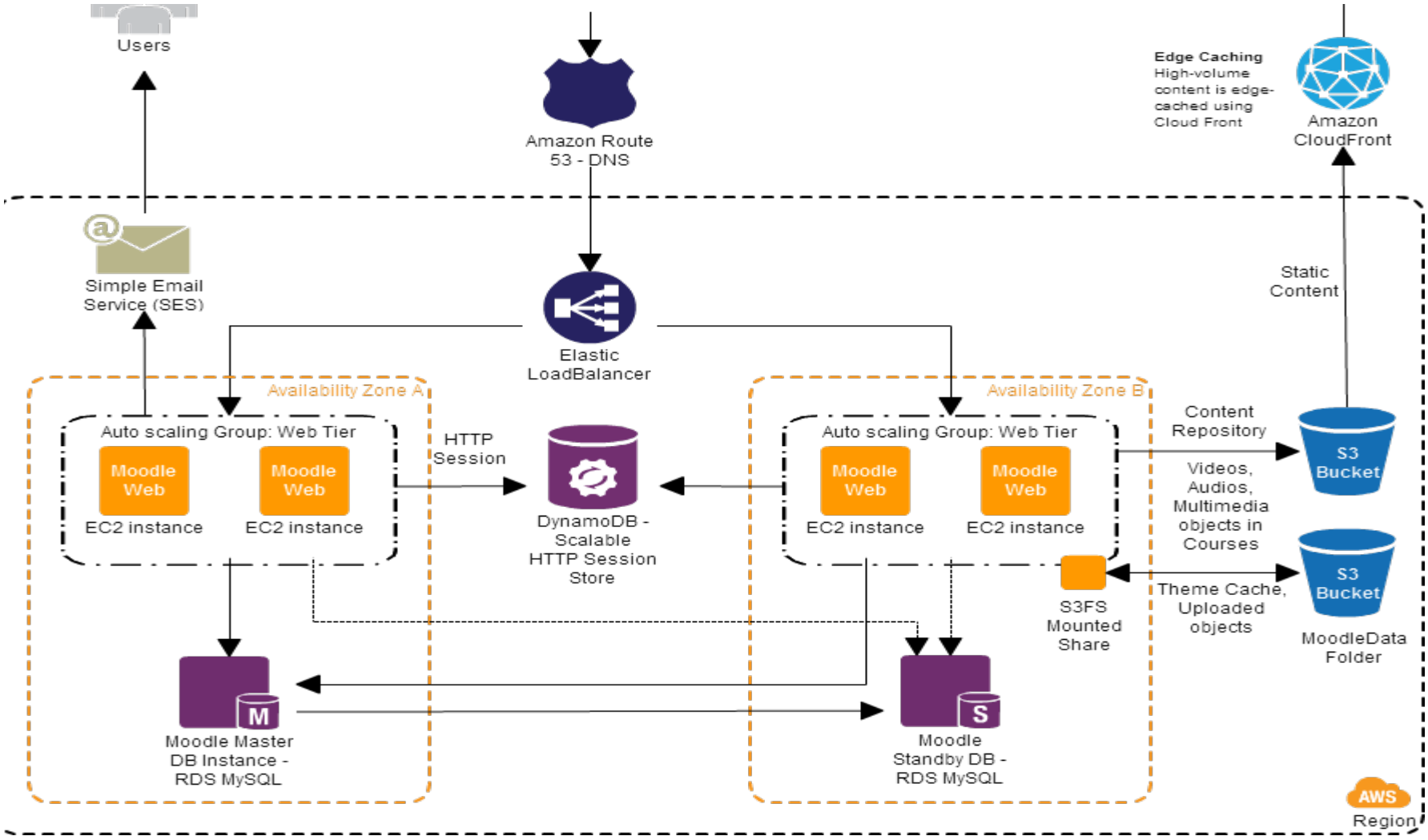


SQS – queue system

- Simple p2p queue
- Possible persistance on platform
- API for different languages
- Vendor specific



Autoscaling IaaS



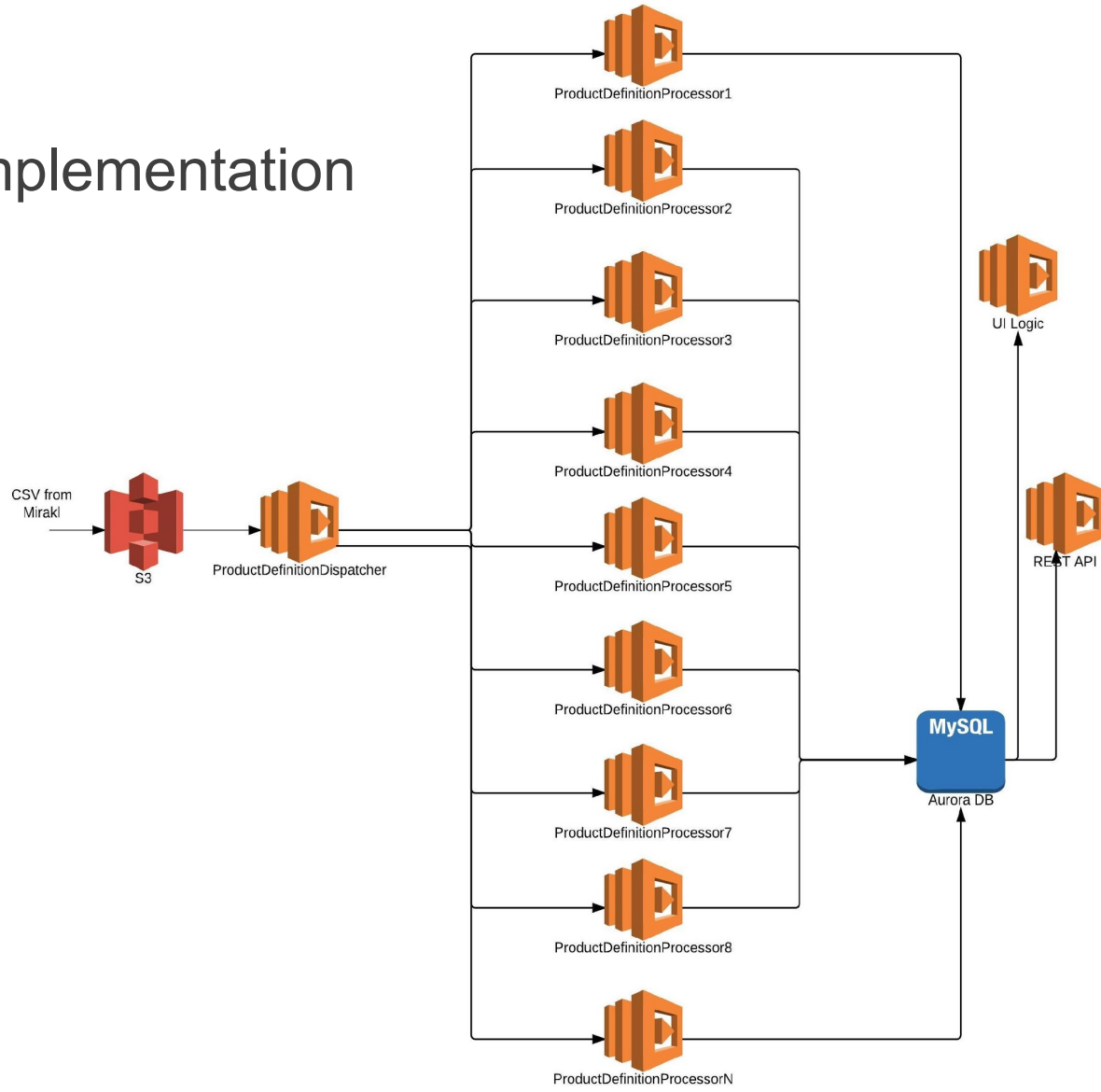
RDS – databases in managed system

- Databases in managed system – above OS
- Oracle, MS SQL Server, Postgress and others
- Backup, snapshot, HA
- Increase of dynamic performance
- Vendor specific

7bulls.com

Lambda – serverless implementation

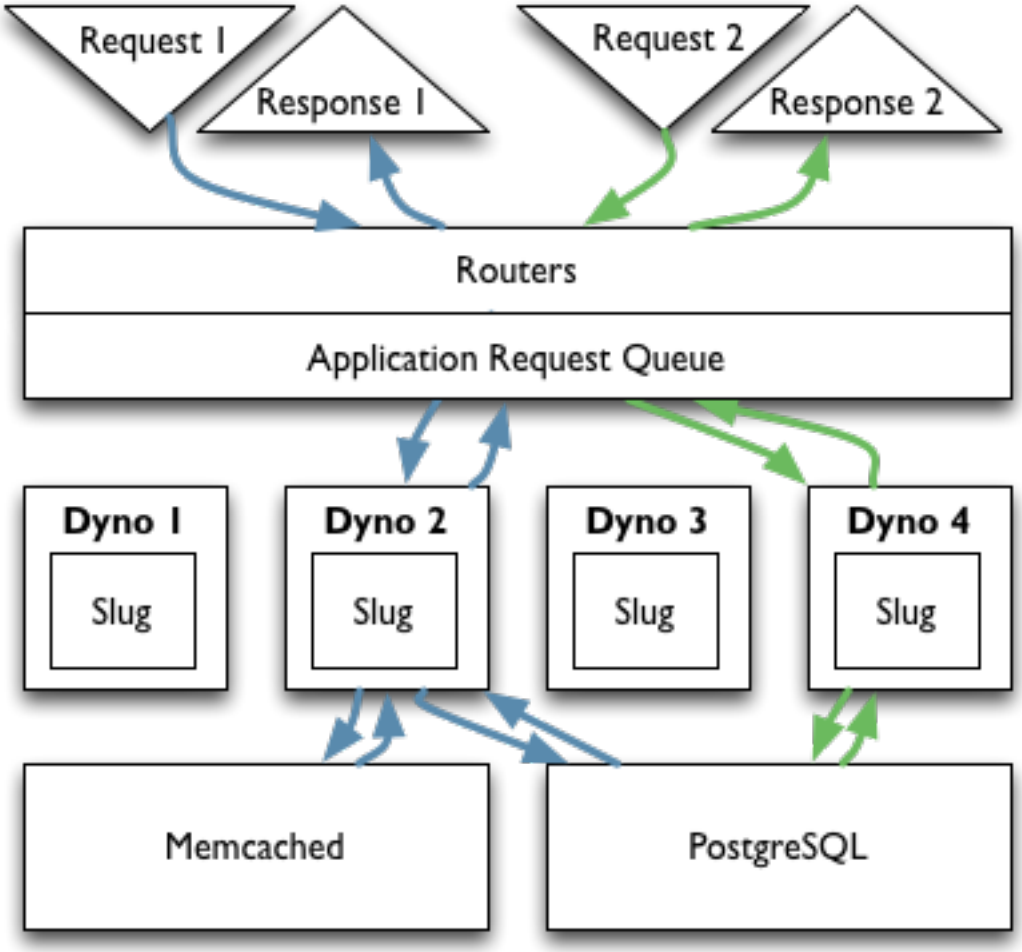
- Components exist only at the time of execution.
- Payments only for actual use.
- Different programming languages.
- Easily scalable.
- Limited lifetime.
- Vendor specific.



Legenda:
S3 - bucket do którego uploadowane via ftp są csv z definicjami produktów od merchantów
ProductDefinitionDispacher - lambda uruchamiana po upload pliku do S3 która uruchamia przetwarzanie dla zawartości pliku
ProductDefinitionProcesor - lambdy przetwarzające wiersze z pliku CSV, dla każdego wiersza oddzielna lambda przetwarzająca od początku do końca daną definicję produktu
Aurora DB - baza danych z definicjami produktów od merchantów, definicjami referencyjnymi i innymi danymi systemu. Dla definicji produktów zawiera tylko atrybuty deduplikacyjne
UI Logic - lambda zawierająca logikę dla UI
REST API - lambda z interfejsami REST dla Hybris

Dynos – PaaS Heroku

- Dynos – smart containers.
- Most convenient implementation of PaaS (that is in fact CaaS).
- Effective scaling.
- Effective management.
- Different programming languages.
- Very efficient development.
- Vendor specific.



Cloud Quality

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use it rationally
8. Cloud Computing – trends and challenges

Cloud ESB - Mulesoft

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use it rationally
8. Cloud Computing – trends and challenges

Cloud Computing – how to use it rationally

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. **Cloud Computing – how to use it rationally**
8. Cloud Computing – trends and challenges

How to choose right?

- Flexibility
- Scalability
- Optimization (efficiency and costs)
- Architecture

?

Flexibility and scalability

- Containers
- Orchestration
- Optimization of deployment
- Horizontal and vertical scaling
- No vendor-lock!

In the cloud we do not do things as usual!

Cost optimization

- Pay-as-you-go pricing.
- Unlimited scaling (whenever needed)
- Selection of the best cloud provider for given application

It can be only done in the cloud!

Cloud computing architecture

- Making full use of cloud computing possibilities - Microservices
- Conscious architecture management in terms of cloud computing possibilities
- Services load balancing and discovery
- Resources utilization
- Proper orchestration

It’s better to do it properly from the start!

Who needs different types of cloud? Small businesses

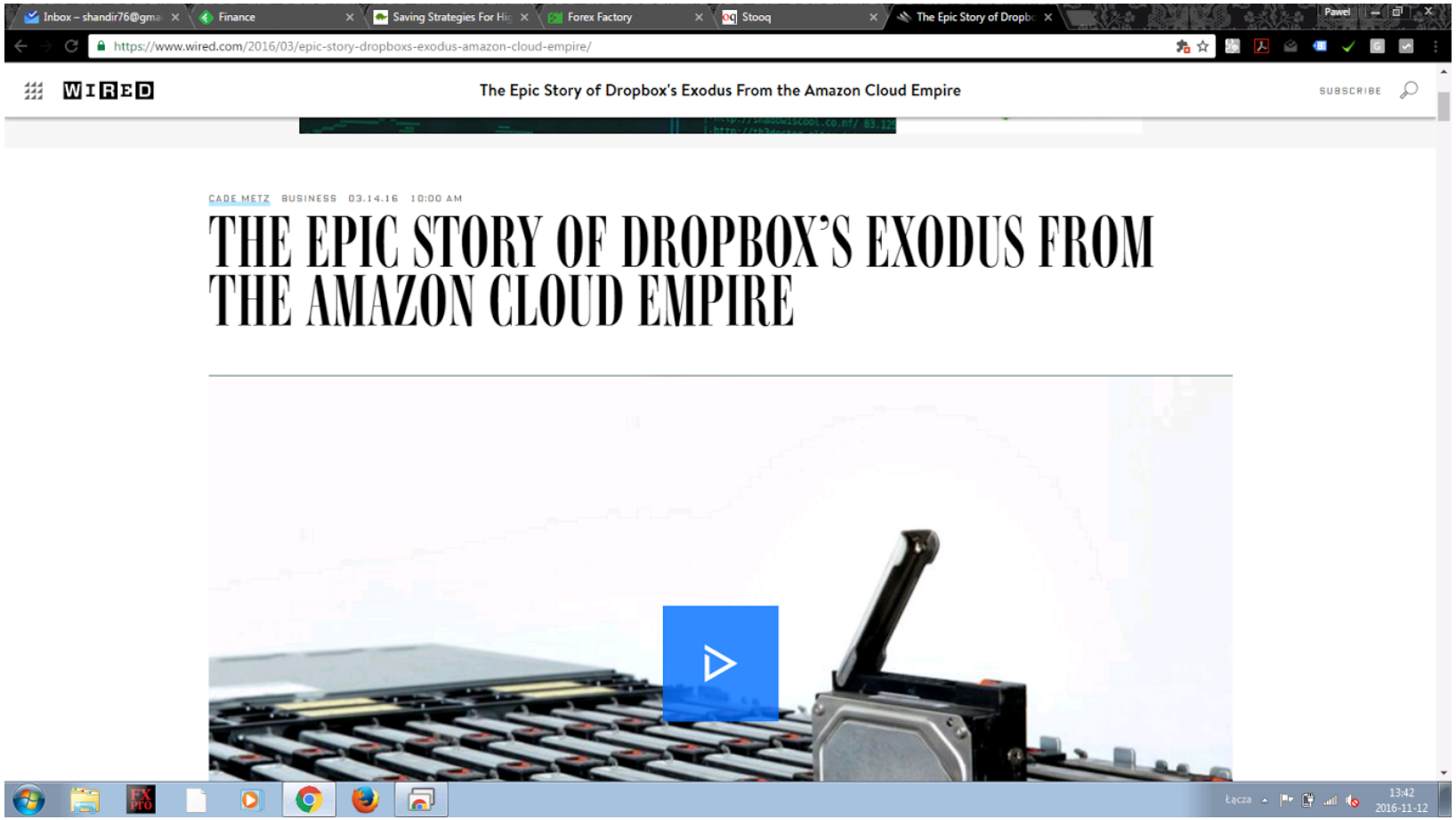
- First of all: reducing expenditures on IT – infrastructure and human resources
- **Cloud only** model is worth considering
- SaaS and CaaS – reducing costs of IT infrastructure and its maintenance
- Serverless – as a support in **processing big sets of data** of nonlinear workload distribution
- Minimum **two different cloud providers**
- Avoiding vendor lock

Who needs different types of cloud? Large businesses

- Containers
- Orchestration
- **Migration to microservices** - ultimately
- **Hybrid cloud model**
  - using **cloud to handle peaks of traffic**, periodic processing
  - basic processing, own infrastructure, cloud as a backup/HA/disaster recovery
- CaaS/PaaS – as a backup platform, to balance the load
- Serverless – as a support for **processing big sets of data** of nonlinear workload distribution
- **Avoiding vendor lock**

Who needs different types of cloud?

Large businesses



Cloud Computing
Trends and challenges

1. Basic concepts
2. What is Cloud Computing
3. Types and models of Cloud Computing
4. Cloud Computing - orchestration
5. MELODIC
6. Practical overview of selected solutions
7. Cloud Computing – how to use it rationally
8. **Cloud Computing – trends and challenges**

Cloud computing trends

- Containers
- Software defined everything (server, storage, network) – software defined architecture
- Multicloud, hybrid cloud
- Orchestration
- Streams and processing large sets of data
- Light apps (Node.js, Python, Groovy)

Cloud Computing - challenges

1. Cloud architecture and infrastructure management and planning
2. Quality assurance
3. Security
4. SLA Ready – Common Reference Model

Cloud Computing - myths

1. Security
2. Data (especially confidential data)
3. Vendor lock
4. Hybrid solutions
5. Mythological marketing

Cloud Computing - summary

1. Amazon Webservices
2. Multicloud - Melodic
3. Cloud Quality
4. Cloud Integration
5. Architecture

Thank you

7bulls.com

Cloud Computing - Security

Paweł Skrzypek
Tadeusz Reczynski
2017.06.14

Threats in the cloud- what does it mean?

Threats
Cloud

Table of contents

Security - IT threats - STRIDE model

- Spoof of identity
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Continuous business operation

- Physical security - human and natural threats
- Recovery center
- Possibility of increase of resources
- Backup and replication
- Vendor lock-in
- Costs of running IT business

What is cloud and what are its types

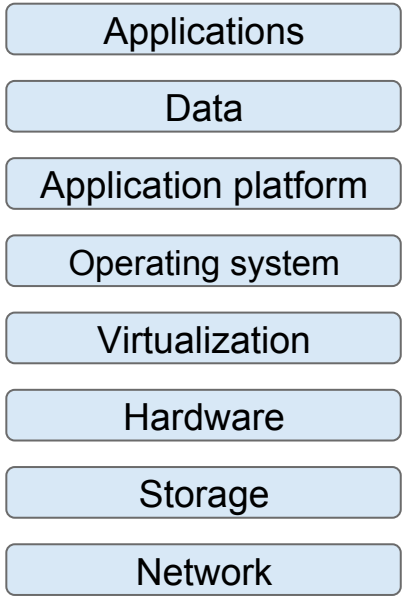
Cloud Computing is a **possibility of using infrastructure** (servers, disk drives, networks etc.) located in the server room of cloud service provider.

Its main types are:

- **IaaS** – Infrastructure as a Service
- **CaaS** – Container as a Service
- **PaaS** – Platform as a Service
- **PaaS serverless** (FaaS)
- **SaaS** – Software as a Service

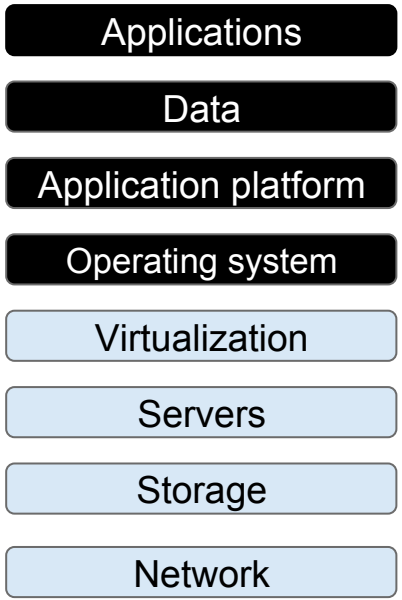
SaaS – what is it?

Software as a Service – **sharing the app as a whole in cloud,** through an Internet access.



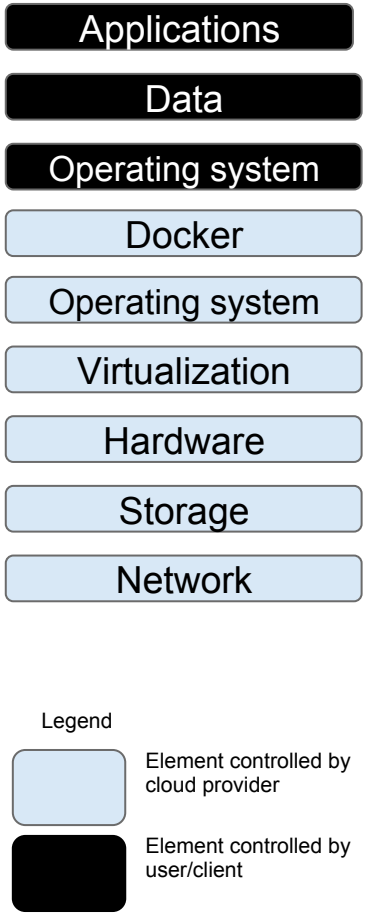
IaaS – Infrastructure as a Service

Sharing **of selected IT infrastructure elements** in a cloud model using virtualized resources.



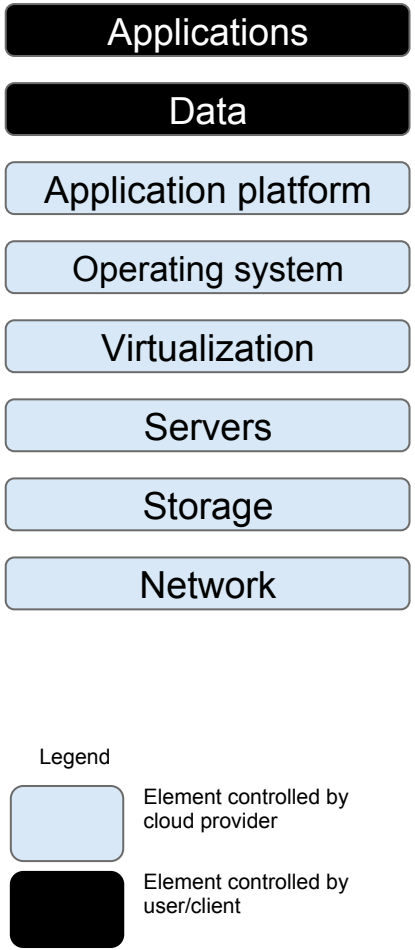
CaaS – Container as a Service

CaaS – Container as a Service – CaaS services provider offers a platform that enables **running containers** according to given parameters.



PaaS – Platform as a Service

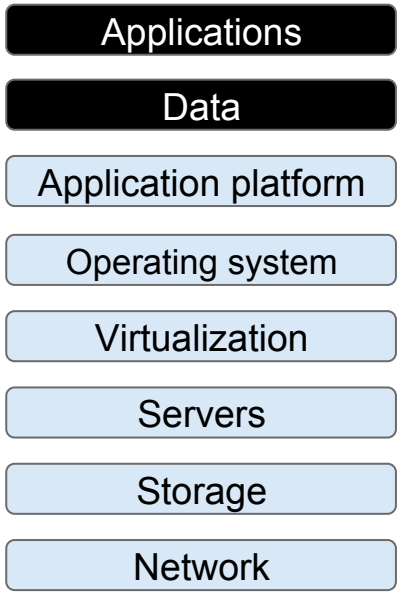
PaaS – Platform as a Service
– PaaS services provider
offers a platform that enables
running applications
according to given
parameters.

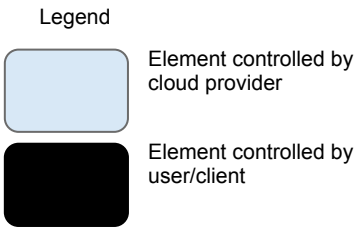


7bulls.com

PaaS serverless (FaaS)

PaaS service can be used in a serverless version, meaning that the **payment is made only for the number of times the application is run or its functions are called.**

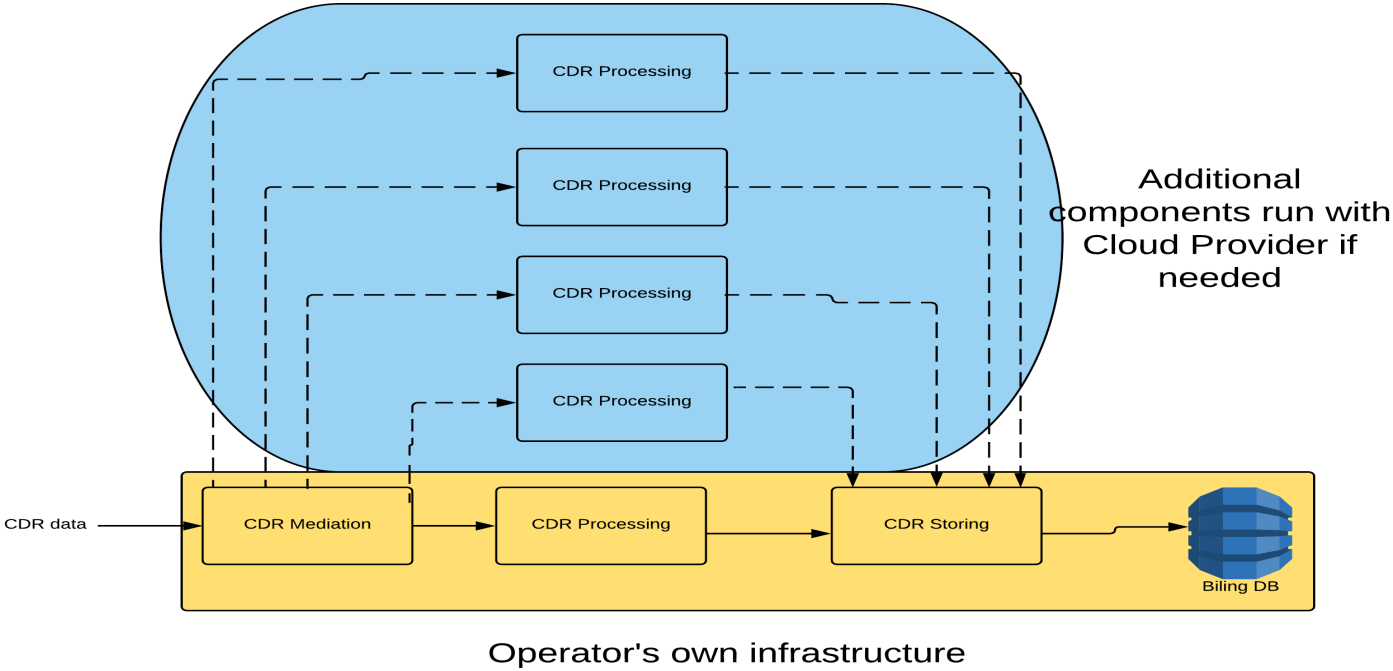




Cloud Computing - features

- Pay-as-you-go pricing
- Unlimited automatic resource scaling
- Selection of the best cloud provider
- Streams and Big Data
- Independence from equipment malfunction

Flexibility, increase of performance

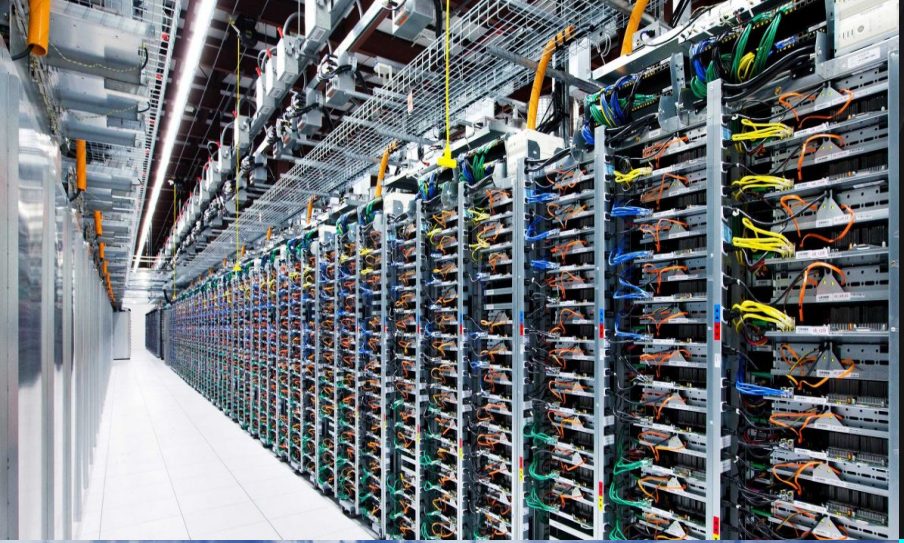


7bulls.com

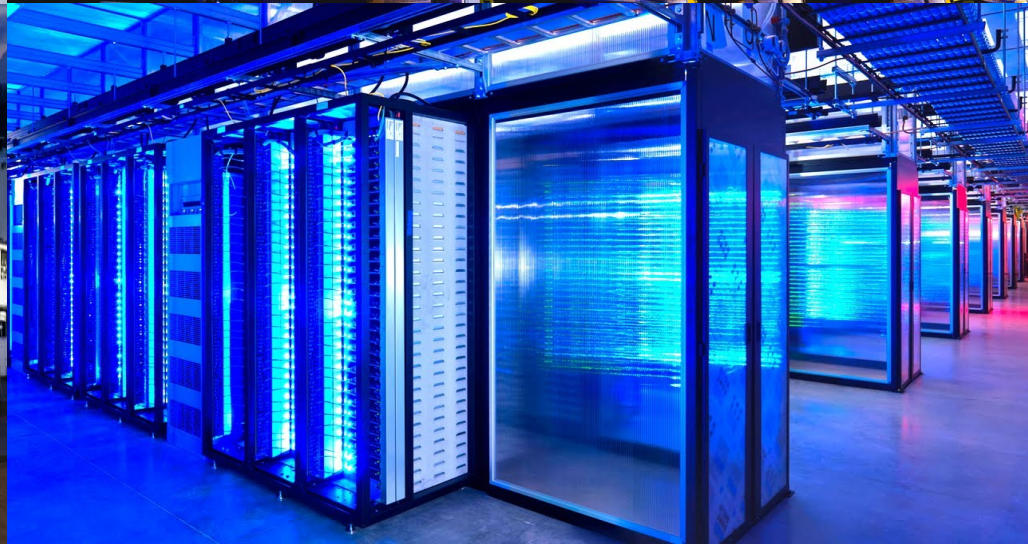
Diversification - multicloud

- No vendor lock
- Increasing flexibility of choice
- Reducing risk of malfunction or loss of data
- Choosing what is best

AWS - 42 data centers and 3 under construction




Google | google.com/datacenters



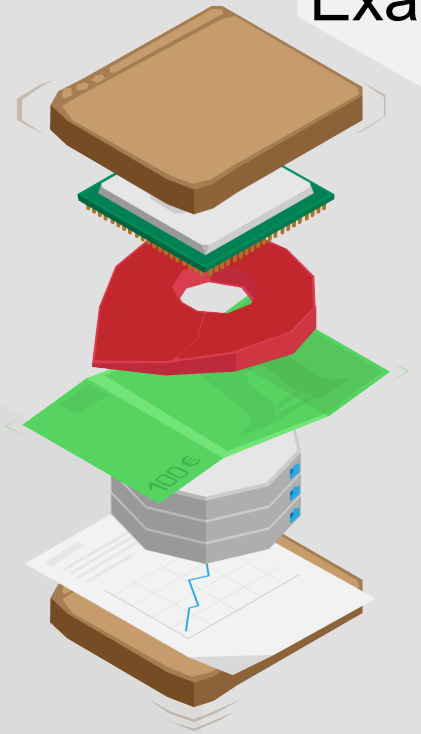
Example: AWS Compliance

- ISO 9001 Global Quality Standard,
- ISO 27001 Security Management Standard,
- ISO 27017 Cloud Specific Controls,
- ISO 27018 Personal Data Protection,
- PCI DSS Level 1 Service Provider (The Payment Card Industry Data Security Standard founded by American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa Inc.),
- SOC Compliance (AWS Service Organization Control (SOC) Reports)
- CSA - Cloud Security Alliance Controls,
- C5 [Germany] - Operational Security Attestation,
- Cyber Essentials Plus [UK] - Cyber Threat Protection,
- G-Cloud [UK] - UK Government Standards,
- IT-Grundschutz [Germany] - Baseline Protection Methodology
- ...

https://aws.amazon.com/compliance/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Example: Melodic - optimum multicloud use

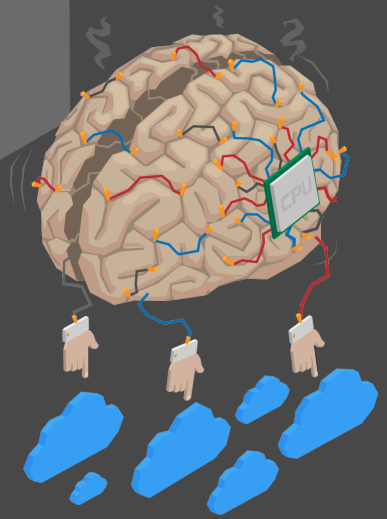


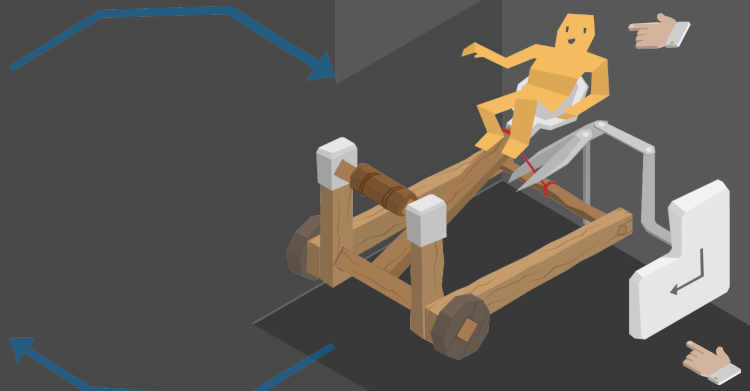
- Managing Multicloud/Big Data
- Application migration
- Optimization of infrastructure and application
- Running application on multiple cloud platforms of multiple providers
- Security of application connections
- No vendor lock
- Open source
- European research and development project

Melodic
Big data cloud

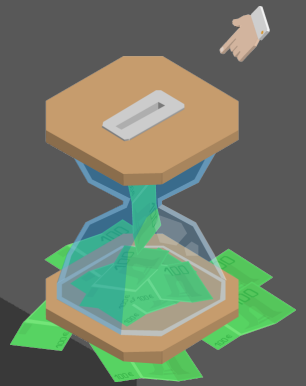
BIG-DATA CLOUD MADE EASY

Melodic calculates best multicloud option for your app



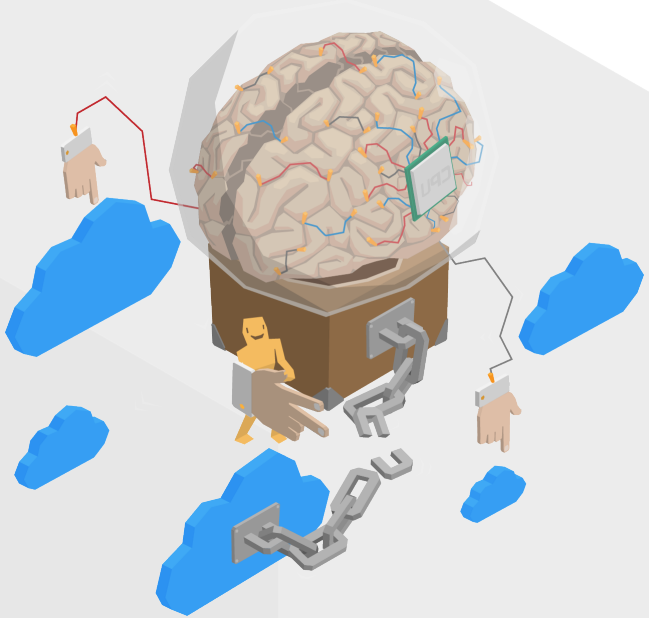


Deployment is automatic

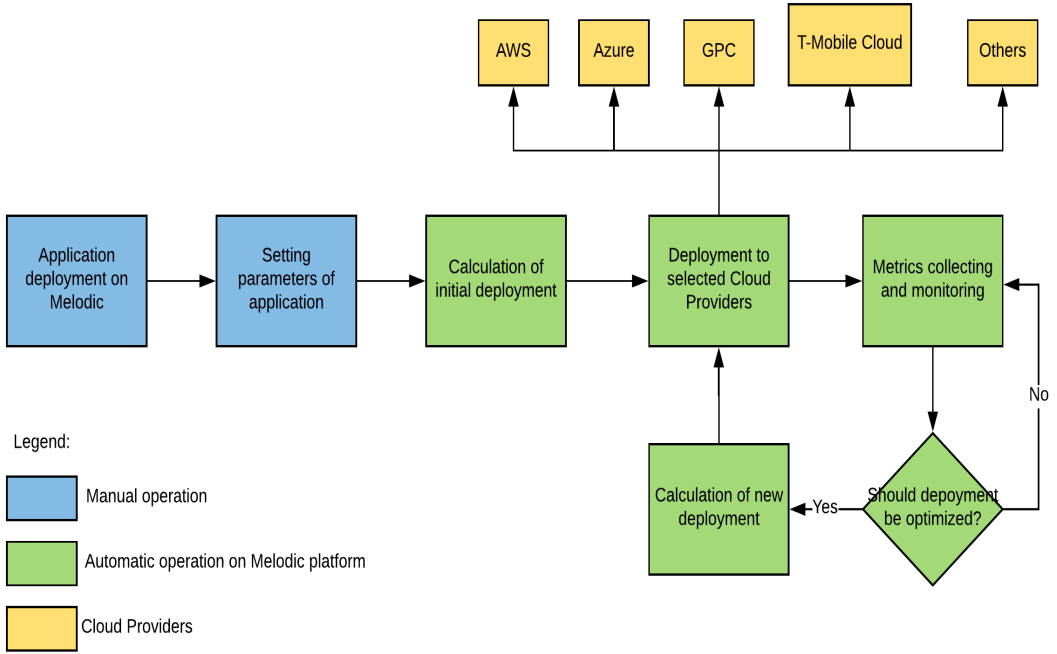


You save time and money

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 731664



Melodic - how does it work?



Safely in the clouds (SiC)

The commonly used perimeter security is no longer sufficient:

- susceptibility of mass solutions to break-in from outside,
- the owner of the stored content has no influence on the infrastructure,
- the infrastructure administrator's staff has access to the stored content,
- also in the case of encrypting databases - keys are stored on servers,
- high safety standards, confirmed statistically by the low error rate are, from the perspective of the victim, worthless

SiC™ - Safely in the Cloud

Security from scratch - security by design:

- the highest possible, for a distributed environment, degree of securing the confidentiality, authenticity and integrity of the information being processed,
- three-layer architecture in accordance with the client-server model,
- independence of the security level from the location of the application server and database server, as well as the legal title to use them,
- unavailable content for the infrastructure administrator's staff,
- hierarchical access management,
- accountability of operations on objects deposited in SIC™.

SiC™ - securing information

- encryption with asymmetric algorithms (public key cryptography),
- encryption and decryption only in the client application at the Subscriber's station, requires the use of a user's private key,
- saving information in a relational database in the form of cryptograms,
- private key stored on a cryptographic card or (not recommended) in an encrypted file on the authorized person's station,
- two-way communication under the secure TLS protocol in the currently recommended version, including the exchange of data and authentication information

SiC™ - access management

- access to the repository is only available to people with appropriate private registrations registered in the system,
- the infrastructure provider has no access to information, so there is no need to enter into contracts for entrusting the processing of personal data (in accordance with Article 31 paragraph o.o.d.o.),
- it is possible to define a pool of IP addresses from which you can log in to the system,
- certificates are registered by an indicated person representing the Subscriber of the service

SiC™ - access management

Hierarchical structure of rights:

• the solution provider registers in the process of implementation a superior certificate, ensuring continuity of management, assigning it to the Subscriber,

• other certificates are registered by the indicated person representing the Subscriber,

• it is not possible to modify own rights or rights of persons at parent levels,

Full accountability of user activities:

• access control lists,

• full history of changes,

• availability of all previous versions of objects

SiC™ - Functionality

- Implemented the option of electronic signatures:
- unqualified,
- qualified using a key on a cryptographic card,
- the possibility of using the server qualified signature is prepared,
- Possibility of unlimited extension of the repository structure,
- the possibility of dynamic reconfiguration of the repository,
- the ability to automatically map the subscriber's organizational structure, create any workgroups and grant access rights in accordance with current needs,
- the possibility of sharing dedicated resources with external partner organizations, taking into account their organizational structures.

Security - IT threats - STRIDE model

Event	In the cloud
Spoof of identity	
Tampering	
Repudiation	
Information disclosure	
Denial of service	
Elevation of privilege	

Legend:  better  worse  the same

Continuous business operations

Event	In the cloud
Physical security - human and natural threats	
Recovery center	
Possibility of increase of resources	
Backup and replication	
Vendor lock-in	 / 
Costs of running IT business	

Legend:  better  worse  the same

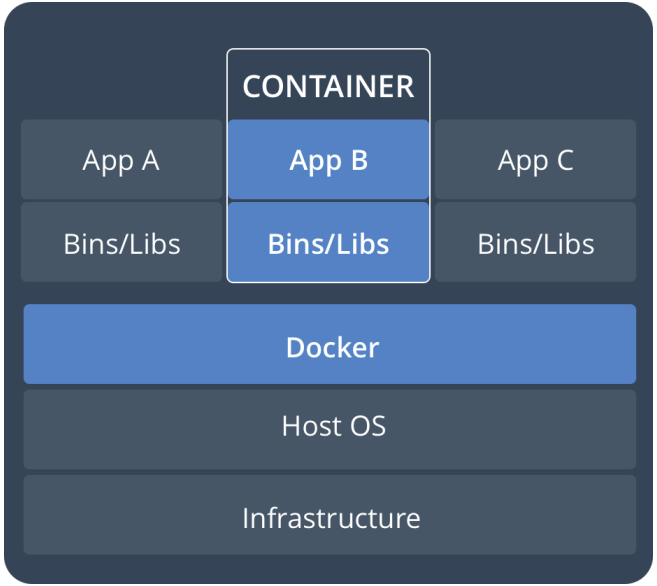
Questions, remarks, suggestions, open issues…

backup

7bulls.com

CaaS – Container as a Service

CaaS – Container as a Service – CaaS services provider offers a platform that enables **running containers** according to given parameters.



Continuous business operations

- Physical security - human and natural threats 
- Recovery center 
- Possibility of increase of resources 
- Backup and replication 
- Vendor lock-in  ()
- Costs of running IT business 

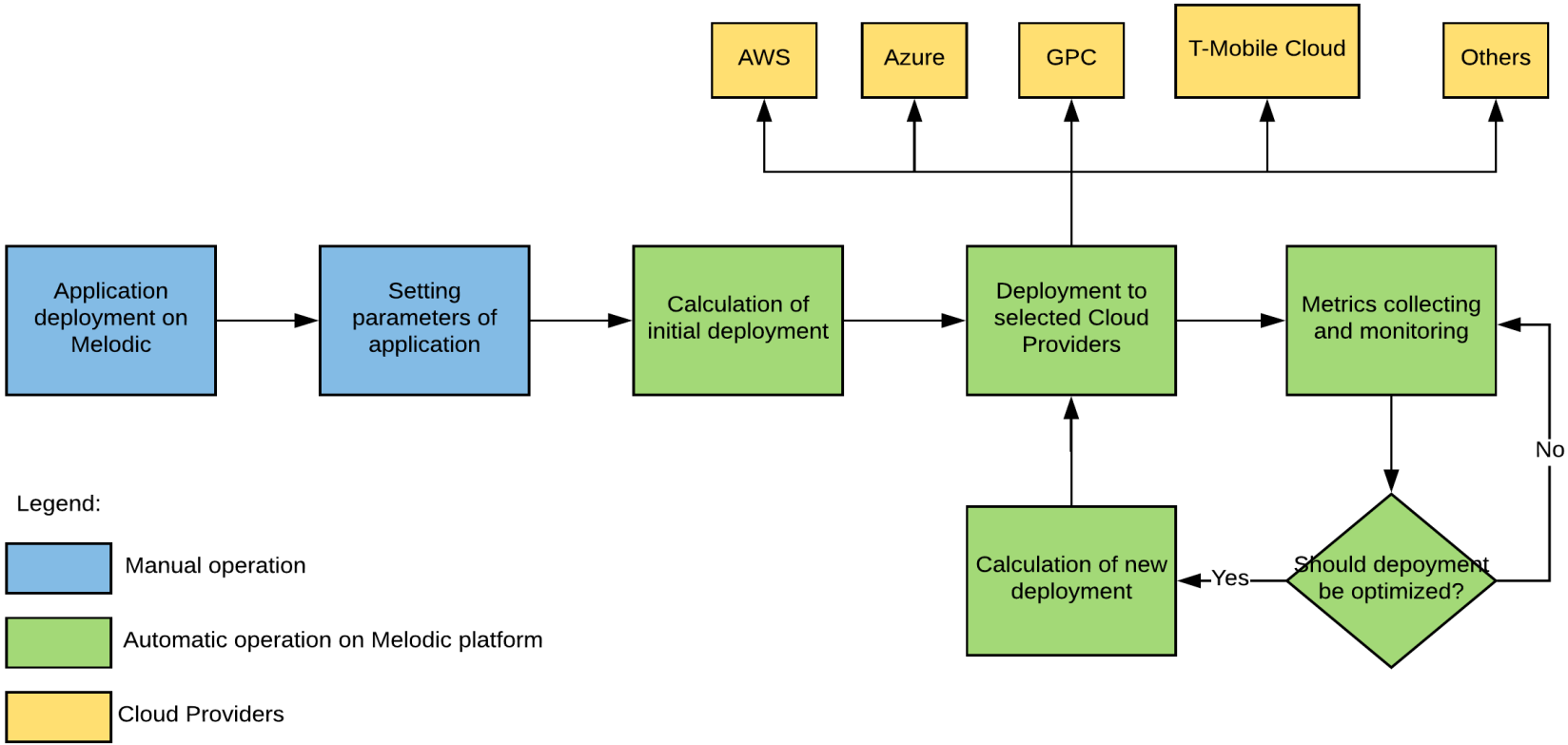
Security - IT threats - STRIDE model

- Spoof of identity
- Tampering
- Repudiation
- Information disclosure
- Denial of service 
- Elevation of privilege

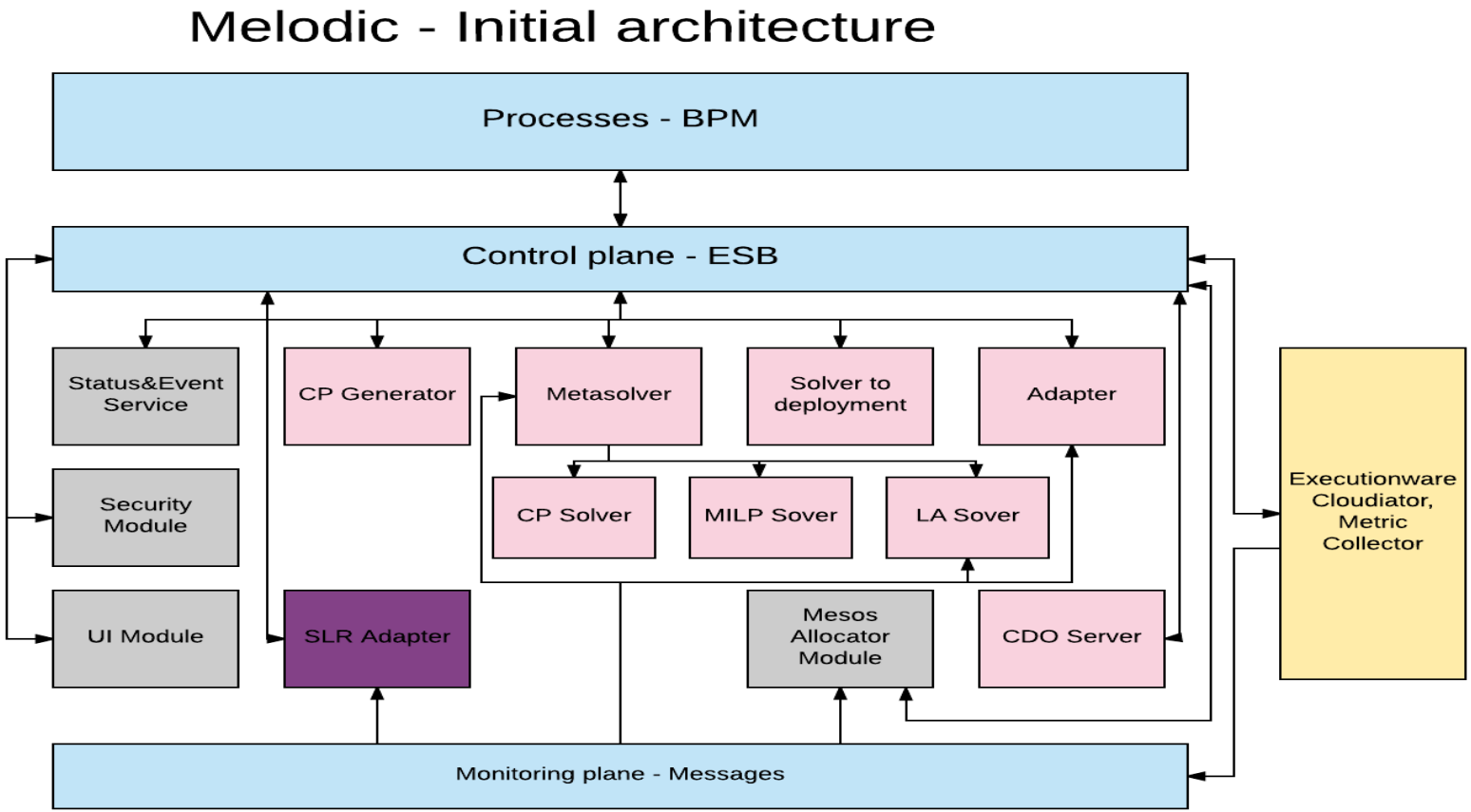
Example: Melodic - optimum multicloud use

- Managing Multicloud/Big Data
- Application migration
- Optimization of infrastructure and application
- Running application on multiple cloud platforms of multiple providers
- Security of application connections
- No vendor lock
- Open source
- European research and development project

Melodic - how does it work?

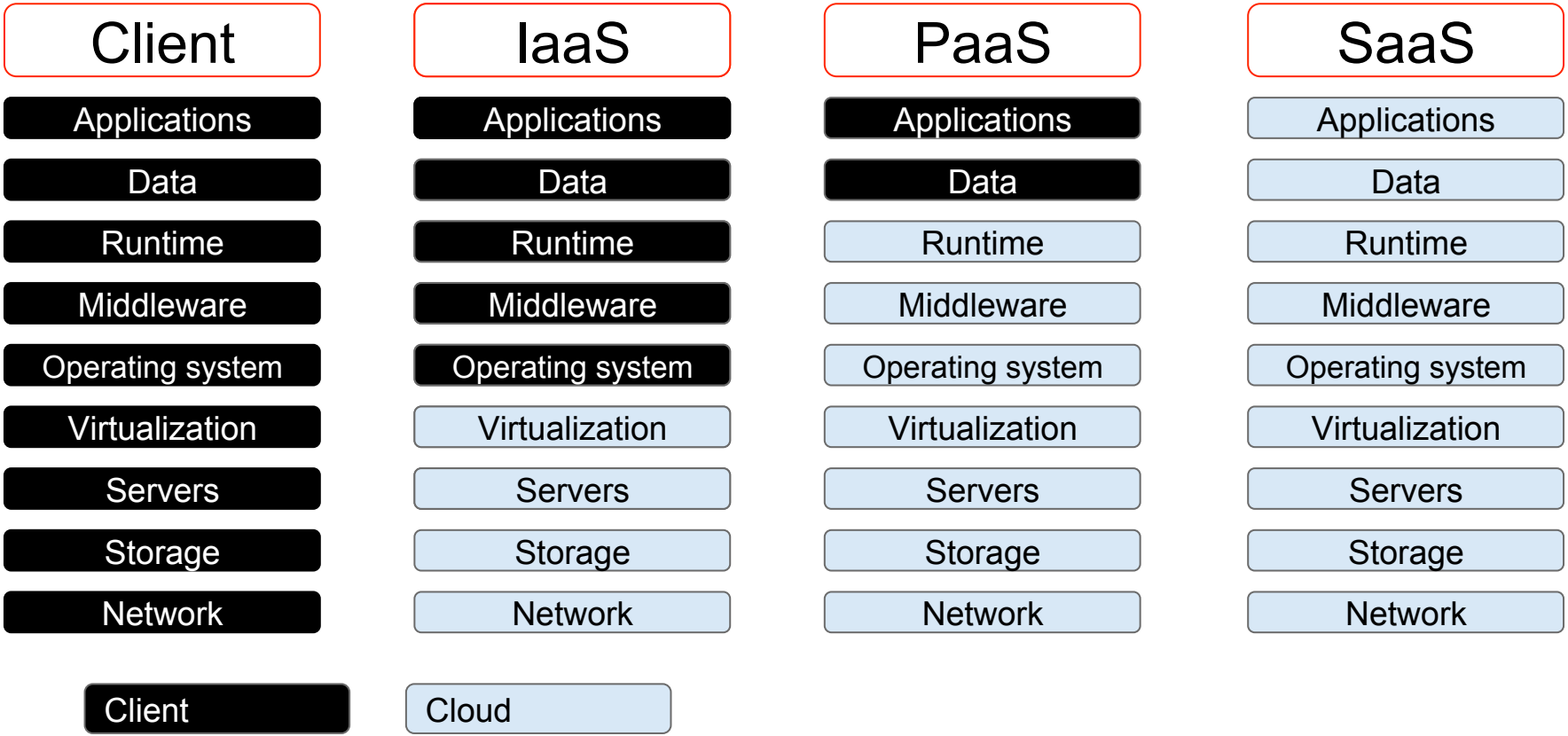


Melodic - how it looks



7bulls.com

Cloud computing models



Melodic - project

- The project is funded through European Commission programme
- It was launched on 01/12/2016
- Duration: 3 years
- First iteration after 18 months
- System will be available as an open-source software
- 7bulls.com is responsible for quality, integration and visualization of the solution

Diversification - multicloud

- Solution tailored to application/data specific needs
- Avoiding performance barriers
- Continuous innovations as a part of specialized solutions of a given provider

Multicloud:

automation and optimization of processing in

cloud environment

Agnostic approach to particular cloud providers' offers makes it possible for us to dynamically optimize operation of application environment based on these services. It refers to both operational and cost dimensions. Modern solutions allow for automatic creation and balance of workload between different cloud platforms. Thanks to that, it is possible for one app to be running in many cloud environments at the same time.

Cloud computing model, meaning possibility of transferring systems and processes of data processing to allocated, shared or external resources that belong to cloud infrastructure, has introduced changes into the IT paradigm. That however entails new challenges connected to, for instance, concept of programmatically defined infrastructure. It makes it possible for IT resources – including servers, disk drives and networking hardware – to be treated not as physical devices, but as highly flexible programmatically defined beings.

At the same time experience shows that depending on cloud resources of one provider poses real business risks. This is where multicloud approach comes in. It involves using many different services offered by many different cloud providers as a part of one heterogeneous and dynamically changing architecture. Multicloud-based approach gives us, among others, possibility of transferring processing to resources and cloud environments, which prove more adequate, safer or simply cheaper at a given time. Such approach allows using the best features of each cloud solution and keeping the price as low as possible as well as prevents the client from the so-called „vendor lock-in".

[DIFFERENT CLOUD MODELS AND DIFFERENT CLOUD SERVICES]

New model of providing IT services has revolutionized concept of IT resources administration in many companies. Today cloud computing is an obvious choice for many organizations. This is due to the aforementioned flexibility, possibility of better exploitation of resources (private cloud) and limited costs of acquiring IT resources needed and maintaining existing infrastructure (public cloud). Cloud model also facilitates creation of new applications and services, data storage, creation and restoration of backup copies as well as processing large sets of data. Maintaining such services as a part of classic IT architecture – when there are cheaper, more secure and flexible solutions available – often becomes unprofitable. It's worth remembering that there are many different kinds of cloud solutions available on the market. The bigger the knowledge of their specific nature, possibilities and limits, the

easier it will be to achieve expected business objectives. Most of cloud services available on the market are of one of three general categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) or Software as a Service (SaaS). Whereas cloud infrastructure can be shared (public), local (private) or a combination of the two (hybrid).

IaaS model (Infrastructure as a Service) is the most basic category of cloud computing services, where IT infrastructure – servers, virtual machines, disk space, networks and operating systems – is leased from the cloud provider and the cost reflects the actual use of provided resources.

PaaS model (Platform as a Service) on the other hand was designed to make it easier for developers to create applications quickly, without so much as a thought of configuring basic server infrastructure, mass storage memories, networks and databases and to facilitate managing such environment.

One variety of PaaS are the services known as FaaS (Function as a Service). Key differentiating factor between those two models is the fact that in the FaaS model, the components are created only for the time it takes for the completion of the task and are deleted afterwards. Such approach significantly reduces costs as they do not go up when the workload is low. It also increases scalability and enables on-demand creation of components in a number adequate for task completion. FaaS model is the most recent cloud solution, currently being commercially offered by limited number of providers. It does however have its limitations, i.e. maximum length of component lifecycle.

SaaS (Software as a Service) services can be described as providing only certain functionalities of a software as a service. The client pays for each use of the service and gets access on demand only. It's worth noticing that SaaS is the fastest growing segment of technology able to function basing on PaaS and IaaS.

Very interesting solution being developed in terms of CAMEL language support is the Melodic platform. It is an environment enabling optimization and deployment of application written in CAMEL language in a multicloud model. Melodic platform is a European project, funded through Horizon 2020 programme and developed in the open-source model. **Its functionality allows running applications using CAMEL language** in a completely agnostic way in respect of cloud provider and selecting most optimal deployment model depending on application's profile.

[FIVE PILLARS OF CLOUD COMPUTING] Even though more and more companies use cloud environments and a large part of those companies chooses multicloud solutions, there are some difficulties that might appear during deployment and maintenance processes. These are usually connected to managing many different services and licenses at once, the need of building and managing successive environments and inconsistent security rules. To ensure proper and safe deployment as well as further administration of services in clouds, we have many different methodologies that might prove helpful. One of those is the philosophy of five pillars of well-architected framework developed by Amazon. These are:

1 **Security** – user and app authentication and authorization, data encryption, certification and other security-related issues.

2 **Reliability** – system availability and stability, redundancy (also geographic), resistance to disruptions.

3 **Performance efficiency** – productivity and scalability of resource use, especially using autoscaling.

4 **Cost optimization** – approaching cloud architecture design in a way that minimizes its overall costs by, i.e., dynamic adjustment of size (scaling depending on requirements) and optimization of backup creation processes.

5 **Operational excellence** – many tasks connected to operational procedures of system maintenance and development, especially automation of Continuous Integration and Continuous Delivery.

Along with the development of cloud computing one can observe a couple of other significant trends, including: expansion of the concept of programmatically defined infrastructure, aiming to shorten average time of software delivery as well as automation of testing and deployment processes.

Typical Continuous Integration/ Continuous Delivery (CI/CD) solution includes following steps:

1 **Building the software** – the software is built from particular branch of version control system using selected build automation tool (usually Maven or Gradle).

2 **Verification of solution** – in terms of static code quality analysis with SonarQube or similar software.

3 **Storing the software in a repository**.

4 **Performing unit tests** – tests are performed on a built software verifying if the methods and key elements of the software work as expected.

5 **Deployment to test environment** – automatic deployment of the software to the selected test environment, especially along with creation of the infrastructure (virtual machines, disk space, etc.)

6 **Performing automatic acceptance tests** – functional, performance, security and more.

7 **Supporting manual tests** – manual testing of test cases.

8 **Deployment to selected production environment** – automatic deployment to production environment after accepting testing process.

Leading cloud providers usually have their own specific API interfaces. Their uniqueness however makes it difficult to transfer big systems between cloud platforms of different providers. The solution to that problem is TOSCA (Topology and Orchestration Specification for Cloud Application) **standard and CAMEL** (Cloud Application Modeling and Execution Language) **language created on its basis,** allowing users to describe the application, its requirements and infrastructure independently of cloud provider.

Deployment of fully automated CI/CD solution significantly speeds up the process of delivering software and primarily, it allows for its reproducibility. Delivered version is always built and deployed in the same way and in a completely automatic manner, minimizing the risk of error occurrence. Most popular CI/CD solutions are open source solutions such as Jenkins or Bamboo by Atlassian.

At the same time Continuous Integration solutions, whose task is to build and test software using unit testing, have been for some time used in on-premise environments and bare-metal infrastructures as well, while the development of Continuous Deployment solutions is only possible using cloud model. The reason is that traditional IT environments usually do not allow for programmatic creation of infrastructure, which is essential to Continuous Deployment process.

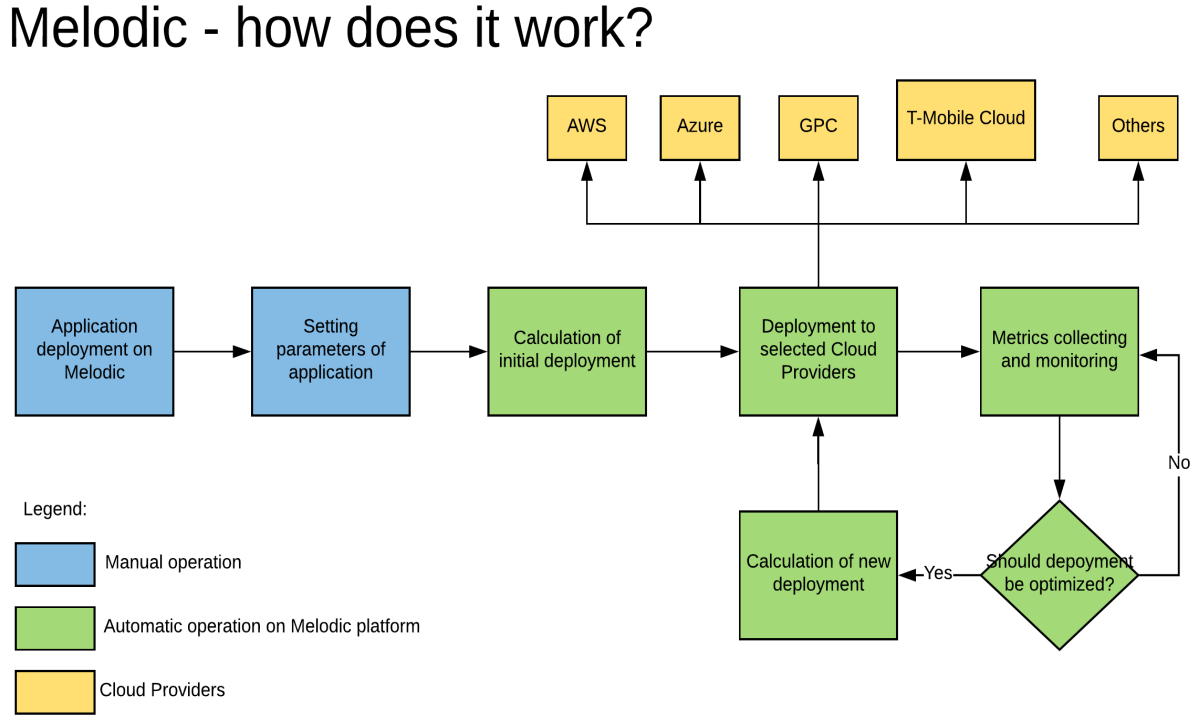
An important element to be concerned about when planning migration to cloud environment, is assuring the quality of IT services. Applying mechanisms of programmatic definition of infrastructure needs constant and multidimensional testing of application systems. Moreover, it concerns aspects beyond the scope of typical quality management solutions. It is necessary to introduce security tests, preferably along with penetration tests, especially in the context of connection between application components and the applications themselves. In the interest of flexibility, scalability and redundancy of cloud solutions, such tests should include, among others, obligatory verification if certain element does not turn out to be a bottleneck in a given solution.

[MULTICLOUD AS A WAY OF RISK REDUCTION]

Along with growing popularity of cloud solutions, possibility of balancing workload between environments of different providers and creating applications with possibility of their automatic deployment in different environments becomes more and more important. Such approach brings significant profits to companies who use cloud solutions, since it allows them to select the best provider for a given application as well as optimize costs and increase security and stability of the solution by diversifying contracting parties on an ongoing basis.

It is worth remembering that leading cloud providers usually have their own specific interfaces for controlling infrastructure provided in the application itself. Their uniqueness however makes it difficult to transfer big systems between cloud platforms of different providers.

The solution to that problem is TOSCA (Topology and Orchestration Specification for Cloud Application) standard and CAMEL (Cloud Application Modeling and Execution Language) language created on its basis, allowing users to describe the application, its requirements and infrastructure independently of cloud provider. Hence it is possible to both deploy application to different environments and optimize its functioning in relation to its profile and requirements defined by the user.



[CAMEL IN PRACTICE]
Very interesting solution being developed in terms of CAMEL language support is the Melodic platform. It is an environment enabling optimization and deployment of application written in CAMEL language in a multicloud model. Melodic platform is a European project, funded through Horizon 2020 programme and developed in the open-source model. Its functionality allows running applications using CAMEL language in a completely agnostic way in respect of cloud provider and selecting most optimal deployment model depending on application's profile. Additional functionalities of the platform are its capability of optimizing big data solutions and data locality awareness.

Application modeling process in CAMEL language as a part of Melodic platform demands defining its components, the connection between them, requirements in terms of performance and resources as well as a way of its deployment. The next step is an automatic optimization of deployment configuration – the platform decides which and whose infrastructure would be the most optimal. Initial optimization is based on the parameters provided the user.

Basing on specific optimal configuration, a precisely defined infrastructure is created using services of selected cloud providers (virtual machines of defined parameters) and the application is deployed with chosen settings of connections between components. After its deployment, the application is being monitored – the data collected define the characteristics of its functioning and serve as the basis for automatic optimization of deployment of application running on the Melodic platform.

[AUTOMATION AS A FUTURE OF CLOUD COMPUTING] Cloud solutions are gradually becoming more and more common among certain companies and organizations as well as IT solutions providers around the world, including Poland. It is due to their help in reducing expenditures on IT solutions, both in terms of equipment (no need for server room and other equipment) and competence (significant part of tasks is being performed by cloud service provider). Making proper use of that potential however demands some knowledge on capabilities and risks coming from using cloud resources on a large scale as well as ability of adjusting resources and applications to the needs of particular organization.

Approach exploited in Melodic platform is the future of cloud computing solutions. Especially using advanced machine learning algorithms allows us to better optimize application deployment process, which is very important for big systems in terms of both performance and costs.

Multicloud – diversification, optimization and security in cloud environment

Abstract
Agnostic approach to particular cloud providers' offers encourages us to consider how to dynamically optimize the usage of IT environment based on these services. It concerns both operational and cost dimensions. Modern technologies allows for automatic creation and balance of workload between different cloud computing platforms. Thanks to that, it is possible for one app to be running in many cloud environments at the same time and acquiring smoothness and reliability of this process relieves recipients of the service of the serious risk of building dependence on its provider (the so-called *vendor lock-in)*.
Another independent trend of thought is leaning towards solutions that concern the possibility of losing control over important information transferred to resources stored in cloud.

MULTICLOUD – ALL IN ONE

The idea of transferring systems and data processing to allocated, shared or external infrastructure, while providing uninterrupted Internet access, called cloud computing, inspires thoughts about the need of change in the paradigm of the IT functioning. Economies of scale, not only in terms of costs, but also in terms of technology, that are the triggering factor for thinking in cloud categories, is however moderated by the awareness of the consequences of the lack of direct control over external resources and relatively high degree of user's independence from the provider. Such limitation obstructs many initiatives, especially in the context of the common opinion on difficult reversibility of cloud migration process.
On the other hand constant development of the cloud stimulates emergence of many different ideas aimed at perfecting ways of its use as well as reducing its limitations. One of such ideas is the concept of programmatically defined infrastructure. The vision where IT resources - including servers, disks, networking hardware etc. – can be treated not as physical devices but as highly flexible beings described and configured with the use of intelligent algorithms. The derivative of such vision is the approach referred to as the *multicloud,* which involves using many cloud computing services offered by many different providers as a part of one, heterogeneous and dynamically changing architecture. Such approach give us, among others, possibility of efficient, ultimately automatic transferring of processing processes to the resources that prove the most adequate, secure or simply the cheapest in given time and conditions. *Multicloud* allows for optimization - using best features of each of the cloud solutions available, keeping the cost as low as possible. At the same time it prevents the user from the consequences of forming dependency on one provider, the so-called „*vendor lock-in*" [i].

DIVERSIFICATION – DIFFERENT MODELS AND DIFFERENT CLOUD SERVICES

New model of providing IT services has revolutionized concept of IT resources administration, making cloud computing an obvious choice in many companies. This is due to the aforementioned flexibility, possibility of better exploitation of resources (private cloud) and

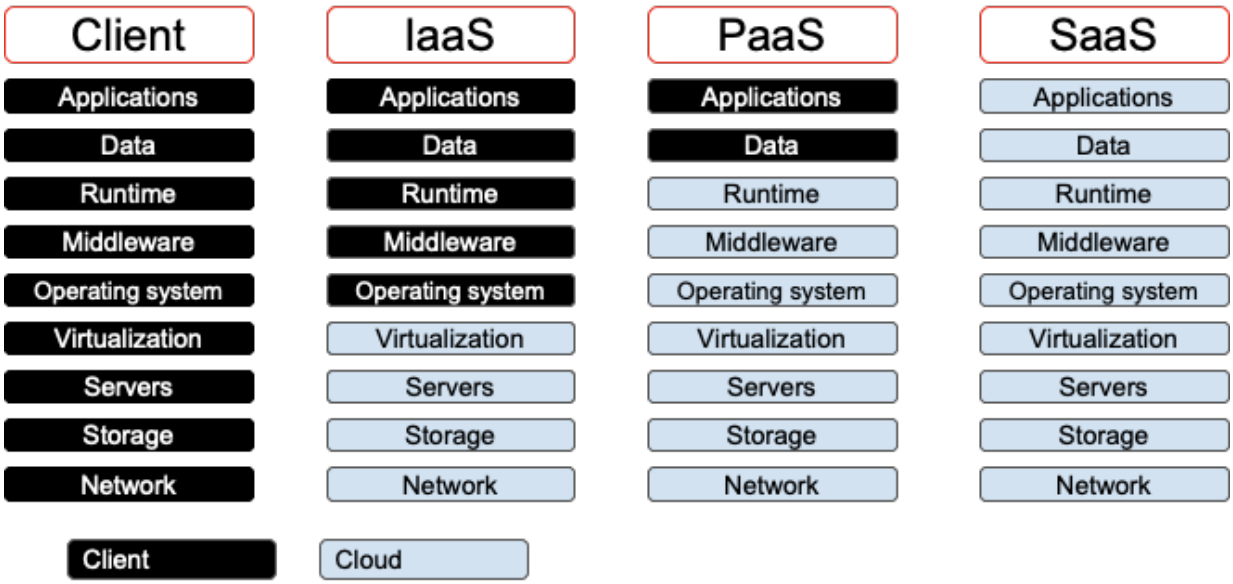
limited costs of acquiring IT resources needed and maintaining existing infrastructure (public cloud). Cloud model also facilitates creation of new applications and services, data storage, creation and restoration of backup copies as well as processing large sets of data. Maintaining such services as a part of classic IT architecture – when there are cheaper, safer and more flexible solutions available – often becomes unprofitable.
there are many different kinds of cloud solutions available on the market. The bigger the knowledge of their specific nature, possibilities and limits, the easier it will be to achieve expected business objectives. Most of cloud services available on the market are of one of three general categories:

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS) lub
- Software as a Service (SaaS).

Cloud infrastructure can be:

- shared (public),
- local (private) and
- combined (hybrid).

The criterion for determining the model of cloud services used is the division of administration of particular elements of IT (responsibility) between the provider and the user as visually explained on the chart below:



IaaS model (Infrastructure as a Service) is the most basic category of cloud computing services, where IT infrastructure – servers, virtual machines, disk space, networks and operating systems – is leased from the cloud provider where the cost reflects the actual use of provided resources.

PaaS model (Platform as a Service) on the other hand was designed to make it easier for developers to create applications quickly, without so much as a thought of configuring basic server infrastructure, mass storage memories, networks and databases and to facilitate managing such environment.
One variety of PaaS are the services known as FaaS (Function as a Service). Key differentiating factor between those two models is the fact that in FaaS model, the components are created only for the time it takes for the completion of the task and are deleted afterwards. Such approach significantly reduces costs as they do not go up when the load is low. It also increases scalability and enables on-demand creation of components in a number adequate for task completion. FaaS model is the most recent cloud solution, currently being commercially

offered by limited number of providers. It does however has its limitations, i.e. maximum length of component lifecycle.

SaaS (Software as a Service) services can be described as providing only certain functionalities of a software as a service. The client pays for each use of the service and gets access only on demand. It's worth noticing that SaaS is the fastest growing segment of technology able to function basing on PaaS and IaaS.

OPTIMIZATION – FIVE PILLARS OF CLOUD COMPUTING

Even though more and more companies use cloud environments and a large part of those companies chooses multicloud solutions, there are some difficulties that might appear during maintenance and deployment process. These are usually connected to managing many different services and licenses at once, the need of building and managing successive environments and inconsistent safety rules. To ensure proper and safe deployment as well as further administration of services in clouds, we have many different methodologies that might prove helpful. One of those is the philosophy of five pillars of well-architected framework developed by Amazon. These are:

1. **Security** – user and app authentication and authorization, data encryption, certification and other security-related issues.
2. **Reliability** – system availability and stability, redundancy (also geographic), resistance to disruptions.
3. **Performance efficiency** – productivity and scalability of resource use, especially using autoscaling.
4. **Cost optimization** – approaching cloud architecture design in a way that minimizes its overall costs, by i.e., dynamic adjustment of size (scaling depending on requirements) and optimization of backup creation processes.
5. **Operational excellence**– many tasks connected to operational procedures of system maintenance and development, especially automation of Continuous Integration and Continuous Delivery.

Along with the development of cloud computing one can observe a couple of other significant trends, including: expansion of the concept of programmatically defined infrastructure, aiming to shorten average time of software delivery and automation of testing and deployment processes.

SECURITY – MULTICLOUD AS A WAY OF RISK REDUCTION

Along with growing popularity of cloud solutions, possibility of balancing workload between environments of different providers and creating applications with possibility of its automatic deployment in different environments becomes more and more important. Such approach brings significant profits to companies using cloud solutions, since it allows them to select the best provider for a given application as well as optimize costs and increase security and stability of the solution by diversifying contracting parties on an ongoing basis.
leading cloud providers usually have their own specific interfaces for controlling infrastructure provided in the application itself. Their uniqueness however makes it difficult to transfer big systems between cloud platforms of different providers.

The solution to that problem is TOSCA (*Topology and Orchestration Specification for Cloud Application*)[ii] standard and CAMEL (*Cloud Application Modeling and Execution Language*)[iii] language created on its basis, allowing users to describe the application, its requirements and infrastructure independently of cloud provider. Hence it is possible to both deploy application to different environments and optimize its functioning in relation to its profile and requirements defined by the user.

CAMEL IN PRACTICE

Very interesting solution being developed in terms of CAMEL language support is the Melodic platform. (*Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing*).[iv]

It is an environment enabling optimization and deployment of application written in CAMEL language in a multicloud model. Melodic platform is a European project, funded through Horizon 2020 programme and developed in the open-source model. Its functionality allows running applications modeled using CAMEL language independently of cloud provider and selecting most optimum deployment model depending on application's profile. Additional functionalities of the platform are its capability of optimizing big data solutions and data locality awareness.

Application modeling process in CAMEL language as a part of Melodic platform demands defining its components, the connection between them, requirements in terms of performance and resources as well as a way of its deployment. The next step is an automatic optimization of deployment configuration – the platform decides which and whose infrastructure to use. Initial optimization is based on the parameters provided the user.

Basing on specific optimal configuration, a precisely defined infrastructure is created using services of selected cloud providers (virtual machines of defined parameters) and the application is deployed with chosen settings of connections between components. After its deployment, the application is being monitored – data collected define the characteristics of its functioning and serve as the basis for automatic optimization of deployment of application running on the Melodic platform.

AUTOMATION AS A FUTURE OF CLOUD COMPUTING

Cloud solutions are gradually becoming more and more common among certain companies and organizations as well as IT solutions providers around the world, including Poland. It is due to their help in reducing expenditures on IT solutions, both in terms of equipment (no need for server room and other equipment) and competence (significant part of tasks is being performed by cloud service provider). Making proper use of that potential however demands some knowledge on capabilities and risks coming from using cloud resources on a large scale as well as ability of adjusting resources and applications to the needs of particular organization.

Approach exploited in Melodic platform is the future of cloud computing solutions. Especially using advanced machine learning algorithms allows us to better optimize application deployment process, which is very important for big systems in terms of both performance and costs.

MULTICLOUD AND CONFIDENTIALITY

The idea of transferring resources and operations to external environments has emphasized the need of increasing the security of data considered sensitive or highly valuable for their holder or user. In the context of General Data Protection Regulation (GDPR), particular attention was given to the danger of dissemination of confidential information (that are given to providers when using *cloud computing*) as well as legal and economical consequences of security breaches.
It is a difficult and complex issue for cloud providers to ensure an adequate level of subscriber's data confidentiality, when they are stored and processed in a nonconfidential manner. In such case it is impossible to effectively differentiate between access rights of technical administrators responsible for proper functioning of the service and its interaction with IT infrastructure, business administrators (who manage access to the service and the service itself on behalf of the subscriber) and users, who should be the only ones to have access to the processed information. It is not easy to designate responsibility in case of disclosure of confidential information. The risk involved, in case of processing information of critical confidentiality level from the point of view of the client's vital interests, might reach an unacceptable level. Storing unencrypted information on the server also means high risk of losing its authenticity and integrity value: more or less free, low-level[v] access to data, often without logging such operations or while doing it in logs that are unavailable to the subscriber, allows for – if service applications do not use electronic signature extensively – undetectable or hardly detectable unauthorized changes or introduction of false information. In such case it is also very difficult to rely on built-in accountability and nonrepudiation ensuring mechanisms.

MINIMUM SECURITY REQUIREMENTS

Because of the reasons mentioned above, storing information given to the provider in an encrypted form using powerful standard algorithms is necessary to be accepted as the absolute minimum requirement. That requirement has to be met in case of databases, separate files stored outside of the bases (i.e. document scans, operation logs etc.) and backup copies of all of these data. There are many techniques and scenarios of implementation of this requirement, but this is not the time and place for their analysis and comparison.

An exemplary solution on the provider's side might be using the so-called *Transparent Data Encryption* technique (available, among others, in Oracle i Microsoft SQL Server databases) and encryption of whole data carriers – disk drives, disk arrays etc. In each of these cases the providers have to guarantee encryption of backup copies (databases and carriers), which might require them to come up with separate solutions.

In case of processing information of extremely high sensitivity[vi] (including confidentiality) it may turn out that the adequate security level might be only ensured by using solutions referred to as the *client-side encryption*. Encryption and decryption of data is carried out by the client with the use of keys available only to logged in users with appropriate level of authorization. Such solutions – even though highly recommendable – are not too widespread at the moment, because of, among others, significant difficulties ensuing from trying to ensure efficiency of search operations (that will be adequate for practical use) in encrypted databases. Examples of such approach are, for instance, solutions used by Thales e-Security (Vormetric Data Security Platform) and a local solution known by its trade name SiC (Safety in the Cloud) offered by, among others, 7bulls.com.

It should not be however presumed that using *client-side encryption* solution completely relieves the subscriber from the risk of disclosing protected information to unauthorized parties or its unauthorized modification – in such case however it takes place in the subscriber's environment, whose security should be ensured by the subscriber himself.

MULTICLOUD AND A DURABLE MEDIUM

Client-side encryption solutions mentioned above are usually based on the concept of the so-called keychain – each information is encrypted with a set (a chain) of cryptographic keys available for authorized users. Deleting user's key from the set would equal losing the access to adequate portion of information.

Clients' documents (including personalized documents) that are stored in bank repository and shared with the clients, might become irremovable before a certain deadline ensuing from regulations and agreements, also when they are encrypted with key that is known only to the client and are deprived of features allowing for assigning them to a particular client – for example by giving them random name that is not saved in the client's catalogue.

The depositor of such documents is deprived of the possibility of their selective removal basing on criteria different than the required retention time. Thus successful removal or substitution of single client's documents would require deleting entire contents of such repository, which no institution can afford for obvious reasons.

Comprehensive arrangement of methodology of preparing and storing documents and data, along with efficient use of available methods of managing keys used to encrypt sensitive information gives us possibility of relatively cheap and easily implemented solution to the issue of the so-called durable medium.

---

[i] https://www.techopedia.com/definition/26802/vendor-lock-in

[ii] http://searchcloudcomputing.techtarget.com/definition/TOSCA-Topology-and-Orchestration-Specification-for-Cloud-Applications

[iii] http://camel-dsl.org/

[iv] http://melodic.cloud/

[v] without engaging the application layer, using direct access to file system, operations in SQL etc.

[vi] In a sense defined by the ISO/IEC 27000 series standard.