D3.3.1
Evaluation of ICT hosting environments

June 28, 2019

EcoGrid 2.0 is a research and demonstration project funded by EUDP (Energiteknologisk Udviklings- og Demonstrationsprogram). The 9 partners in the project are:

Main Authors:

Name/Partner	Email
Michael Baentsch/IBM	mib@zurich.ibm.com
Peter Buhler/IBM	bup@zurich.ibm.com

TABLE OF CONTENTS

1 Overview

This document aims at providing an overview of potential alternative IT deployment means of the software components in the EcoGrid 2.0 project. It begins with an overview of the various high-level software components comprising EcoGrid 2.0, commences with a review of a motivation and short history of current cloud technology and closes with recommendations on to how to deploy each component, taking into consideration business as well as data protection aspects, security aspects and elements concerning practical viability within the constraints of the project.

2 EcoGrid 2.0 components

2.1 High-level component list

The following software components comprise EcoGrid 2.0 IT infrastructure:

2.1.1 Data Warehouse

The data warehouse component stores all the data collected during the operation of EcoGrid 2.0: Information about the participating households, time-series information on their energy consumption, temperatures, heating system switching states, etc. In short, this is a classical system of record where a potential treasure drove for data scientists has accumulated over the past several years. Accordingly, this system has to be subject to stringent monitoring, access control and backup regimes. For practical and commercial reasons this component is hosted in the data center at BEOF.

2.1.2 Green Wave Reality - GWR Home Energy Management System (GWR HEMS)

This component relays all sensor information and actuator commands to and from the Green-Wave Reality (GWR) devices installed in households in Bornholm. This GWR product is outside the realm of control of EcoGrid 2.0, the interaction with this component is based on the API specification provided by GWR.

2.1.3 Siemens Home Energy Management System (Siemens HEMS)

This component relays all sensor information and actuator commands to and from the Siemens devices installed in households in Bornholm. This Siemens product is outside the realm of control of EcoGrid 2.0, the interaction with this component is based on the API specification provided by Siemens.

2.1.4 Internal Data Importers

This set of components retrieves and stores project internal data into the EcoGrid 2.0 data warehouse.

Participant data: This component integrates the standard BEOF customer management system with the EcoGrid 2.0 participants management. As obviously all participant households in EcoGrid 2.0 are also BEOF customers, this component not only handles sensitive personal information such as home addresses and user names but also is directly hooked up to BEOF's core IT, this component arguably is the one component that has to be operating within the realms of control of BEOF network boundary.

Power consumption of EcoGrid 2.0 households: This component is co-located with the data warehouse within the data center at BEOF as this data is originating there and can be considered to be most sensitive from both the commercial as well as privacy perspective. This data permits linking individual households to their living patterns, e.g., times of longer absences are equally discernible as are times of presence and heavy use.

GWR and Siemens HEMS data: Based on interface specifications of GWR and Siemens HEMS the relevant data such as state of heating equipment, indoor temperature, etc. is retrieved from these systems. This component is co-located with the data warehouse for efficiency reasons.

EcoEx data: This data on planned and executed activations of households is collected from the aggregators and is stored in the EcoGrid 2.0 data warehouse.

2.1.5 External data importers

This set of components stores data from external sources into the EcoGrid 2.0 data warehouse.

In order to be able to link power consumption patterns with weather patterns, EcoGrid 2.0 collects and stores meteorological data for the region in the data warehouse. The destination of storage is within the BEOF data center, it is logical to co-locate this component.

In order to facilitate linking simple weather-prediction-driven control models developed during EcoGrid (EU) with more advanced market-driven models, data from the energy spot market as well as data on CO2 emission intensity needs to be collected. The destination of storage is within the BEOF data center, it is logical to co-locate this component.

2.1.6 FIP Platform

The Flexibility Interoperability Platform (FIP) provides the infrastructure necessary to test interoperability in EcoGrid 2.0. It enables aggregators to talk to DERs using the EcoGrid 2.0 interoperability protocol. FIP uses GWR and Siemens HEMS control interfaces to provide this functionality.

2.1.7 IBM and Insero aggregators

In order to collate the consumption of single households into a smooth, aggregated form, the aggregator component brings together the individual and mostly erratic energy usage patterns into more stable aggregated form of groups of households. An aggregator represents an aggregated set of one or more DERs and acts as seller of the combined flexibility.

2.1.8 Market clearing house

The flexibility clearinghouse market platform is a generic platform for different established and new actors of the energy market to trade their energetic flexibility. The market platform provides common basic services like communication, authentication, settlement and other necessary services to market products on this platform. The EcoGrid 2.0 market platform can host multiple requests for flexibility of different buyers concurrently.

2.2 Review

In an era of Cloud Computing for each software system component the decision regarding the appropriate deployment environment is required. For some components the Public Cloud is the environment of choice; there are however many system components which benefit from deployment in a private data center of a participating organization, and some organizations require on-premise solutions for regulatory reasons. From a network and data security perspective, there is a need to ensure that on-premise system components handling sensitive information generated within the perimeters of one organization, such as BEOF household meter data, can be integrated into the overall system without compromising their security and functionality. Significant volumes of data used in EcoGrid 2.0 are generated in components outside the EcoGrid 2.0 perimeter, e.g., GWR/Siemens HEMS data, weather and temperature forecasts and current weather reports, Nordpool spot market data. The system design of EcoGrid 2.0 supports efficient cooperation of components running on on-premise IT platforms of participating organizations as well as on Public Cloud platforms.

As soon as EcoGrid 2.0 is to grow beyond Bornholm and the realm of reach of BEOF, re-deploying many of its components might be required from a commercial as well as a technical perspective. The alternatives deployment models will be discussed in the subsequent chapters.

3 Alternatives

The core advantages of cloud IT technology such as scaling-on-demand, easy, standardized systems integration, or performance-optimized computing are generally accepted today. At the same time, some issues, mostly around pricing and strategic control over data make the concept of public clouds, regardless of the inherent level of security, appear like a double-edged sword.

This chapter shows the long history of cloud as seem from a pioneer in IT technology as well as the drawbacks and solutions to the dilemmas posed by public cloud.

3.1 A short history of cloud

This section is an adaptation of an excellent blog post by IBM [1]: The concept of “cloud computing” has been around much longer than you think. Let’s dive into its history.

3.1.1 The humble beginnings of cloud

Believe it or not, the modern-day idea of “cloud computing” dates back to the 1950s, when large-scale mainframes were made available to schools and corporations. The mainframe’s colossal hardware infrastructure was installed in what could be called a “server room” (since the room would generally only be able to hold a single mainframe). Multiple users were able to access the mainframe via “dumb terminals”—stations with the sole function of facilitating access to the mainframes.

Due to the cost of buying and maintaining mainframes, an organization wouldn’t be able to afford a mainframe for each user. It became practice to allow multiple users to share access to the same data storage layer and CPU power from any station. By enabling shared mainframe access, an organization would get a better return on its investment in this sophisticated piece of technology.

3.1.2 Virtualization changes everything

Twenty years later in the 1970s, IBM released an operating system called VM that permitted admins on its System/370 mainframe systems to have multiple virtual systems, or "virtual machines (VMs)" on a single physical node. The VM operating system took the 1950s application of shared access of a mainframe to the next level by allowing multiple distinct compute environments to live in the same physical environment.

Most of the basic functions of any virtualization software that you see nowadays can be traced back to this early VM OS. Every VM ran custom operating systems or guest operating systems that had their own memory, CPU, and hard drives, along with CD-ROMs, keyboards, and networking—despite the fact that those resources were shared. "Virtualization" became a technology driver, and it became a huge catalyst for some of the biggest evolutions in communications and computing.

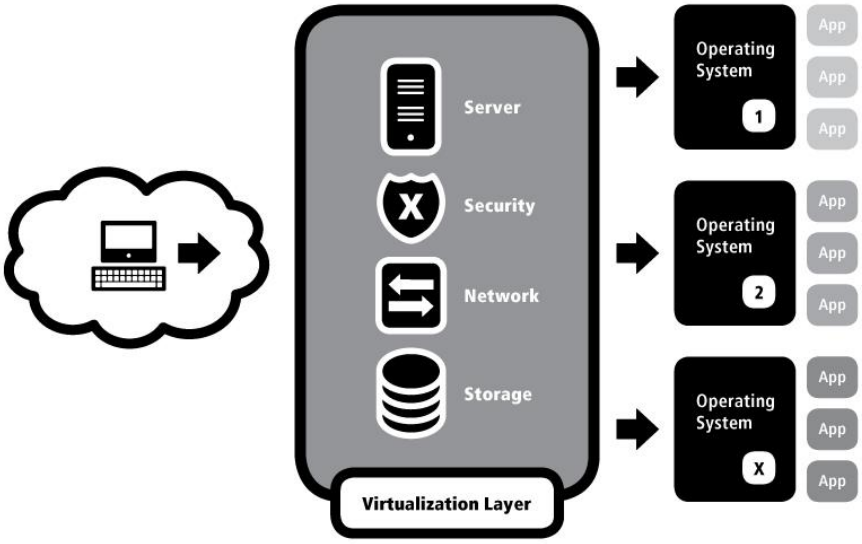
In the 1990s, telecommunications companies that historically only offered single dedicated point-to-point data connections began offering virtualized private network connections—with the same service quality as dedicated services at a reduced cost. Rather than building out physical infrastructure to allow more users to have their own connections, telecommunications companies provided users with shared access to the same physical infrastructure. This change allowed telecommunications companies to shift traffic as necessary, leading to better network balance and more control over bandwidth usage.

3.1.3 Virtualization meets the Internet

Meanwhile, virtualization for PC-based systems started in earnest. As the Internet became more accessible, the next logical step was to take virtualization online. If you were in the market to buy servers 10 or 20 years ago, you know that the costs of physical hardware—while not at the same level as the mainframes of the 1950s—were pretty outrageous. As more and more people expressed the demand to be online, the costs had to come out of the stratosphere and into reality.

One of the ways that happened was through—you guessed it—virtualization. Servers were virtualized into shared hosting environments, virtual private servers, and virtual dedicated dervers using the same types of functionality provided by the VM OS in the 1950s.

What did this look like in practice? Let's say your company required 13 physical systems to run your sites and applications. With virtualization, you can take those 13 distinct systems and split them up between two physical nodes. Obviously, this kind of environment saves on infrastructure costs and minimizes the amount of actual hardware you would need to meet your company's needs.



As the costs of server hardware slowly came down, more users could afford to purchase their own dedicated servers. But they ran into a different kind of problem: One server isn't enough to provide the necessary resources. The market shifted from a "These servers are expensive; let's split them up" belief to a "These servers are cheap; let's figure out how to combine them" mentality. Because of that shift, the most basic understanding of "cloud computing" was born online.

3.1.4 The cloud is born

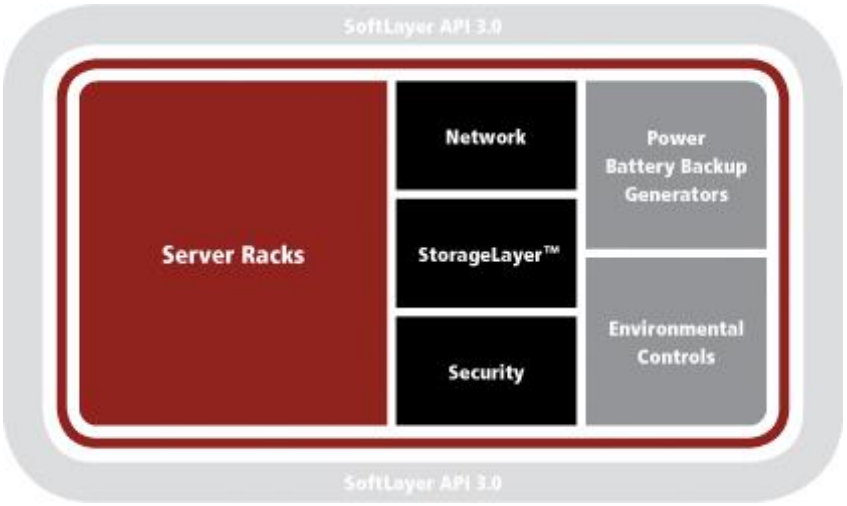
By installing and configuring a piece of software called a hypervisor across multiple physical nodes, a system would present the environment's entire resources as though those resources were in a single physical node. To visualize that environment, technologists used terms like "utility computing" and "cloud computing," since the sum of the parts seemed a nebulous blob of computing resources you could then segment out as needed (like telecommunications companies did in the 1990s). In these cloud-computing environments, adding resources to the "cloud" was easy—add another server to the rack and configure it to become part of the bigger system.

As technologies and hypervisors improved upon reliably sharing and delivering resources, many enterprising companies decided to carve up the bigger environment. They wanted to make the cloud's benefits available to users who didn't have an abundance of physical servers available to create their own cloud computing infrastructure. Those users could order "cloud computing instances" (also known as "cloud servers") by ordering the resources they needed from the larger pool of available cloud resources. Because the servers were already online, the process of

“powering up” a new instance or server is almost instantaneous. Because little overhead is involved for the owner of the cloud computing environment when a new instance is ordered or cancelled (since it’s handled by the cloud’s software), management of the environment is much easier.

3.1.5 Go beyond the standard cloud computing environment

Most companies today operate with the aforementioned definition of “the cloud” as the end-all, be-all—but Bluemix isn’t “most companies.” Bluemix took the idea of a cloud computing environment and pulled it back one more step. Instead of installing software on a cluster of machines to let users grab pieces, we built a platform that automated the manual aspects of bringing a server online without a hypervisor on the server. We call this platform “IMS.” What hypervisors and virtualization do for a group of servers, IMS does for an entire data center. As a result, you can order a bare metal server with the resources you need and without any unnecessary software installed—and that server will be delivered to you in a matter of hours.



Without a hypervisor layer between your operating system and the bare metal hardware, your servers perform better. Because we automate almost everything in our data centers, you’re able to spin up load balancers and firewalls and storage devices on demand and turn them off when you’re done with them. We have ambitious goals for the future. We’ve come a long way from the mainframes of the 1950s.”

The closing statements of the document above point to a potential shortcoming of the general cloud concept: The schism between losing control over one’s data versus necessary abstractions and simplifications useful for most modern IT deployments and users. One can address this schism in one of two ways: The first is to keep running a traditional IT system as is presently done for the EcoGrid 2.0 warehouse. The second is detailed below.

3.2 Cloud delivery models

Abridged from [2] this section discusses different modern cloud delivery models: “Cloud delivery models refer to how a cloud solution is used by an organization, where the data is located, and

who operates the cloud solution. Cloud computing supports multiple delivery models that can deliver the capabilities needed in a cloud solution.

The cloud delivery models are as follows:

- Public cloud
- Private cloud
- Hybrid cloud
- Community cloud

Host	Provider	Enterprise, 3rd Party	Enterprise, 3rd Party	Community, 3rd Party
Owner	Provider	Enterprise	Enterprise	Community
Access	Internet	Intranet, VPN	Intranet, VPN	Intranet, VPN
Users	Public Individuals Organizations	Business Units	Business Units	Community Members

3.2.1 Public clouds

A public cloud is one in which the cloud infrastructure is made available to the general public or a large industry group over the Internet. The infrastructure is not owned by the user, but by an organization that provides cloud services. Services can be provided either at no cost, as a subscription, or as a pay-as-you-go model.

Examples of public clouds include IBM Cloud, Amazon Elastic Compute Cloud (EC2), Google AppEngine, and Microsoft Azure App Service.

3.2.2 Private clouds

A private cloud refers to a cloud solution where the infrastructure is provisioned for the exclusive use of a single organization. The organization often acts as a cloud service provider to internal business units that obtain all the benefits of a cloud without having to provision their own infrastructure. By consolidating and centralizing services into a cloud, the organization benefits from centralized service management and economies of scale.

A private cloud provides an organization with some advantages over a public cloud. The organization gains greater control over the resources that make up the cloud. In addition, private clouds are ideal when the type of work being done is not practical for a public cloud because of network latency, security, or regulatory concerns.

A private cloud can be owned, managed, and operated by the organization, a third party, or a combination. The private cloud infrastructure is usually provisioned on the organization's premises, but it can also be hosted in a data center that is owned by a third party. IBM uses the term *Local* when referring to on-premises private clouds that are owned, managed, and operated by the organization, and the term *Dedicated* when referring to off-premise third-party managed private clouds.

3.2.3 Hybrid clouds

A hybrid cloud, as the name implies, is a combination of various cloud types (public, private, and community). Each cloud in the hybrid mix remains a unique entity but is bound to the mix by technology that enables data and application portability.

The hybrid approach allows a business to take advantage of the scalability and cost-effectiveness of off-premise third-party resources without exposing applications and data beyond the corporate intranet. A well-constructed hybrid cloud can service secure, mission-critical processes, such as receiving customer payments (a private cloud service), and secondary processes, such as employee payroll processing (a public cloud service).

The challenge for a hybrid cloud is the difficulty in effectively creating and governing such a solution. Services from various sources must be obtained and provisioned as though they originated from a single location, and interactions between on-premises and off-premise components make the implementation even more complicated.

3.2.4 Community clouds

A community cloud shares the cloud infrastructure across several organizations in support of a specific community that has common concerns (for example, mission, security requirements, policy, and compliance considerations). The primary goal of a community cloud is to have participating organizations realize the benefits of a public cloud, such as shared infrastructure costs and a pay-as-you-go billing structure, with the added level of privacy, security, and policy compliance that is usually associated with a private cloud.

The community cloud infrastructure can be provided on-premises or at a third party's data center and can be managed by the participating organizations or a third party.

3.2.5 Cloud considerations

The following guidance is provided from NIST.GOV:

'Carefully plan the security and privacy aspects of cloud computing solutions before engaging them. Public cloud computing represents a significant paradigm shift from the conventional norms of an organizational data center to a de-perimeterized infrastructure open to use by potential adversaries. As with any emerging information technology area, cloud computing should be approached carefully with due consideration to the sensitivity of data. Planning helps to ensure that the computing environment is as secure as possible and in compliance with all relevant organizational policies and that privacy is maintained. It also helps to ensure that the agency derives full benefit from information technology spending.

'The security objectives of an organization are a key factor for decisions about outsourcing information technology services and, in particular, for decisions about transitioning organizational data, applications, and other resources to a public cloud computing environment. Organizations should take a risk-based approach in analyzing available security and privacy options and deciding about placing organizational functions into a cloud environment. The information technology governance practices of the organizations that pertain to the policies, procedures, and standards used for application development and service provisioning, and the design,

implementation, testing, use, and monitoring of deployed or engaged services, should be extended to cloud computing environments

‘To maximize effectiveness and minimize costs, security and privacy must be considered throughout the system lifecycle from the initial planning stage forward. Attempting to address security and privacy issues after implementation and deployment is not only much more difficult and expensive, but also exposes the organization to unnecessary risk.

Understand the public cloud computing environment offered by the cloud provider. The responsibilities of both the organization and the cloud provider vary depending on the service model. Organizations consuming cloud services must understand the delineation of responsibilities over the computing environment and the implications for security and privacy. Assurances furnished by the cloud provider to support security or privacy claims, or by a certification and compliance review entity paid by the cloud provider, should be verified whenever possible through independent assessment by the organization.

Understanding the policies, procedures, and technical controls used by a cloud provider is a prerequisite to assessing the security and privacy risks involved. It is also important to comprehend the technologies used to provision services and the implications for security and privacy of the system. Details about the system architecture of a cloud can be analyzed and used to formulate a complete picture of the protection afforded by the security and privacy controls, which improves the ability of the organization to assess and manage risk accurately, including mitigating risk by employing appropriate techniques and procedures for the continuous monitoring of the security state of the system.

Ensure that a cloud computing solution satisfies organizational security and privacy requirements. Public cloud providers' default offerings generally do not reflect a specific organization's security and privacy needs. From a risk perspective, determining the suitability of cloud services requires an understanding of the context in which the organization operates and the consequences from the plausible threats it faces. Adjustments to the cloud computing environment may be warranted to meet an organization's requirements. Organizations should require that any selected public cloud computing solution is configured, deployed, and managed to meet their security, privacy, and other requirements.

Non-negotiable service agreements in which the terms of service are prescribed completely by the cloud provider are generally the norm in public cloud computing. Negotiated service agreements are also possible. Similar to traditional information technology outsourcing contracts used by agencies, negotiated agreements can address an organization's concerns about security and privacy details, such as the vetting of employees, data ownership and exit rights, breach notification, isolation of tenant applications, data encryption and segregation, tracking and reporting service effectiveness, compliance with laws and regulations, and the use of validated products meeting federal or national standards (e.g., Federal Information Processing Standard FIPS 140). A negotiated agreement can also document the assurances the cloud provider must furnish to corroborate that organizational requirements are being met.

Critical data and applications may require an agency to undertake a negotiated service agreement in order to use a public cloud. Points of negotiation can negatively affect the economies of scale that a non-negotiable service agreement brings to public cloud computing, however, making a negotiated agreement less cost effective. As an alternative, the organization may be able to employ compensating controls to work around identified shortcomings in the public cloud service.

Other alternatives include cloud computing environments with a more suitable deployment model, such as an internal private cloud, which can potentially offer an organization greater oversight and authority over security and privacy, and better limit the types of tenants that share platform resources, reducing exposure in the event of a failure or configuration error in a control.'

For more information, see
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf"

3.3 Cloud baseline technology: Docker

In order to baseline the proposal made for the deployment of the new EcoGrid 2.0 market and aggregator components, a quick review of the Docker technology is in order as it we see it as the lowest common infrastructure denominator guaranteeing efficient portability across operating systems and cloud deployment models discussed above. We intentionally do not discuss the merits of the more traditional virtual-machine based deployment models as those are generally accepted and undergird the already deployed EcoGrid components, thus would not truly earn the moniker 'novel' or 'cloud native' which we propose for the most up-to-date components in EcoGrid 2.0.

3.3.1 Overview

A good introduction to Docker (www.docker.io) is given in [3] from which the following has been abridged:

The Problem

The first question to answer for any technology is "what problem(s) does it solve?" Docker tackles these nagging pain points in DevOps:

Dependency matrix – applications have direct dependencies. Each of these dependencies has their own dependencies, and so on.

"Works on my machine" – says your coworker, as you struggle to deploy the latest code from the source repository. This may be the most annoying, non-helpful (yet honest) answer you can get from a coworker. I've heard (and said) this quite often. You know the situation: you do whatever it takes to get the code to run locally, and you try to document everything you did, but this problem keeps cropping up again and again.

Application maturity – as the application matures, and you upgrade dependencies, you have to deal with the dependency tree (i.e., the "matrix" of dependencies), and make sure that these dependencies are reflected in every target environment in which you will test.

Integration Challenges – as you migrate the application to different environments, you have to be aware of other Line-of-Business applications running on the target host and resolve those one-at-a-time in each environment as part of every migration.

"Let the hunt begin!" – as you migrate to higher environments, and new issues shake out, you have to ask (often) "What's different between this environment and the last one where everything worked fine?" and manually match up the environments to resolve the differences.

"Um, this application is ready for the cloud, right?" – asks your manager (and their manager, and so on). The traditional one-environment-at-a-time migrate/test strategy is hard enough when DevOps maintains control of every environment. But when it comes to Platform-as-a-Service (PaaS) – and the environments are now outsourced – we are abstracted from the target environment, and the problems above compound and make it virtually impossible to reliably deploy an application.

What Docker is NOT

If you're familiar with Server Virtualization (e.g., VMWare), it is tempting to think of Containerization and Server Virtualization as synonymous. And while there is some overlap like application isolation and resource management efficiency improvements, they are not the same.

Server Virtualization through the use of Virtual Machines (VMs) solves problems like:

- Data center energy use optimizations (aka, Greening)
- Reducing vendor lock-in
- Faster server provisioning
- Increased uptime

Server virtualization depends on a component known as the hypervisor, which abstracts the Virtual Machine (VM) from the underlying hardware. This hardware abstraction requires the hypervisor to be a fairly substantial piece of software indeed! Containers like Docker on the other hand, abstract applications from the OS, and in the case of Docker this is achieved through the Docker Container Engine. Applications can run in isolation on the same OS instance, resulting in a smaller footprint. In short, server virtualization and containerization are similar, but have different design goals and achieve different results in the data center.

The Solution

Like shipping containers revolutionized the overseas import/export industry by making the shipping payload opaque and standardized, Docker containers allow the kind of application isolation developers seek without the overhead of the hypervisor.

Dependency Matrix – your application has dependencies, and this is outside of the scope of Docker. As developers, it's our job to make sure our software has the right dependencies in place to run correctly. But once I have my dependencies resolved (through your Maven build, for example), Docker will ensure they are consistent from one environment to the next through the Docker image.

"Works on my machine" – the changes necessary to get the source to run in your environment are captured in the docker image. So when your colleague needs to run the code, they download the image, along with the code (and dependencies) and it just runs.

Application Maturity – as the application matures, its dependencies will naturally change. As with the dependency matrix, Docker can't help me decide this, but once I have it working, I modify the image, and now it just works in every environment to which I deploy the application.

Integration Challenges – again, the design of Docker saves the day. Just as my Docker image ensures consistency among various environments, it guarantees a level of isolation from other

applications running on the target host (without the overhead of the Server Virtualization Hypervisor). So integration becomes a non-issue.

“Let the hunt begin!” – the Docker Container image is the same for each environment, so there are no manual diffs to perform.

“Um, this application is ready for the cloud, right?” – Platform-as-a-Service (PaaS) providers like IBM Bluemix ensure that Docker Containers will deploy and run consistently from one environment to the next. Any cloud provider that runs Docker (e.g., IBM, Amazon WS, and Rackspace, just to name a few) can be configured so that you just push your Docker image to the cloud and your app runs.

The Bottom Line on Docker and Containers

Docker helps achieve faster development turnaround, application isolation, and consistency among deployments, resulting in ease of integration and better DevOps experiences. Docker does not solve every DevOps issue, certainly. But this exciting technology ensures consistency among all of your target environments, making life easier for DevOps personnel.”

3.3.2 Security properties

For a more complete discussion on the security properties of Docker technology, also as compared to virtual machines, see [4] from which the following has been abridged:

“Linux containers are implemented through virtualization at the system call level, applications running inside containers share the same underlying Linux kernel. Therefore, cloud services built using containers offer several benefits compared to virtual machines, specifically:

- An application running inside a container can be expected to have near bare metal performance while the same application running inside a virtual machine will not be able to reach that performance level. The reason for this lies in the fact that containers do not emulate devices but access system resources directly.
- The startup delay of containers is much shorter than that of a virtual machine since containers typically only start a few applications while a virtual machine may first run the firmware before booting an entire operating system.
- Since containers start only a few applications, they use resources, such as memory, more efficiently and can therefore be deployed with much higher density than virtual machines.
- Containers provide simplified management. The cloud operator takes responsibility for life cycle management of the operating system (optimization, updates, patching, security scans) allowing users to focus on application development and management.
- Containers provide better portability. Standardized and light-weight image formats such as Docker enable nearly perfect transfer of application across environments: from development to production and from on-premise and off-premise deployments.
- The reduced size of containers leads to a smaller attack surface for cloud customers' workloads.

- Access to a common Operating System Kernel provides higher visibility to the behavior of individual applications. Similarly, access to critical data and events may reveal anomalies and mis-configuration before they become evident through other means.
- Containers encourage microservice-based application architectures, which delegate persistent data to backend datastores and away from compute instances. This reduces the problems of unguarded proliferation of confidential content, which is a common side effect of image clone and copy in the virtual machine world."

4 EcoGrid 2.0 deployment strategy

4.1 Investigations in the context of EcoGrid 2.0

In order to make the above general recommendations concrete and implementable in the context of EcoGrid 2.0, we split the components along the lines of the project partners. In principle it could have been a possible architectural decision to develop an overall system as system of interacting microservices that can be deployed in any type of cloud as discussed above. This was however not the primary focus in early project stages and allows us to introduce the concept in a practical incarnation only for the new IBM components developed during EcoGrid 2.0 such as to not lose the focus on the actual functionality to be delivered in the project.

At the start of EcoGrid 2.0 we investigated the immediate introduction of pure cloud-based components to reap the benefits mentioned above. For EcoGrid 2.0 Flexibility Marketplace we performed deployments into IBM Bluemix during testing in initial heating seasons of EcoGrid 2.0. We had to acknowledge though, that several lessons learned during EcoGrid 2.0 ultimately called for an adaptation of the original design goals:

- In order to de-risk EcoGrid 2.0, particularly not risking the loss of continuous data collection, we retained the data warehouse in the form developed during EcoGrid EU. This decision also served EcoGrid 2.0 well with regard to compliance to European data privacy regulation in that with this architecture, the full control of user data within the confines of the EcoGrid 2.0 compute centre at BEOF. The alternative investigated, a cloud without strong guarantees regarding the whereabouts of data and legal assurances regarding particularly EU data privacy regulations, was one key reason letting us revise the initial architectural design in this area.
- The placement of some IT control components proved to be outside the control of the EcoGrid 2.0 team in the same way that they could not be changed during EcoGrid EU and that primarily applies to the components run exclusively on premises of outside component providers such as Green Wave Reality and Siemens. Accordingly, in EcoGrid 2.0 this placement had to be retained.
- The actual use of some public cloud components in the initial phases of EcoGrid 2.0 created some non-intuitive insights, namely that using presumably simple and stable cloud APIs can be a challenge due to their factually constantly evolving nature. Some unexpected access control issues in year two convinced the IBM team that service level improvements are more important than gaining further experience with open cloud technology.

In any case, the goal of the IBM team was to provide the IBM market place platform and the flexibility aggregator as cloud-native components that could be easily deployed on any cloud infrastructure. During the final project stages, we indeed ran the core components – for all the reasons mentioned above and based on our experiences gained in the context of EcoGrid 2.0 – within an IBM private cloud instance in the IBM Zurich Research Laboratory. At the same time, we decided to make the code available in an easily deployable Docker image such as to show the benefits of a hybrid cloud as introduced above. A worthwhile technical experiment concluding

EcoGrid 2.0 would have been to deploy the components at some arbitrary public cloud. But then again, we probably would not have learned anything specific for EcoGrid 2.0 and furthermore valued operational stability during the final project's heating season higher than additional operational lessons learned. Therefore, we conclude this report with discussions where each component should be running from the perspective of maximum long-term economic viability, including perspectives of cost, security, or separation of concerns for any potential successors of EcoGrid 2.0.

It certainly also would be advisable to transform all other components of the EcoGrid 2.0 IT infrastructure, the existing databases in Bornholm as well as all software developed by the project partners into microservices (Docker images) that can be deployed wherever dictated by business needs.

4.2 Ideal deployment

This section makes a proposal for implementation of all IT components of the EcoGrid 2.0 project in the light of the alternatives explored during the execution of EcoGrid 2.0 and as reviewed in this document.

4.2.1 Data warehouse

Given the complexities of complying with European data privacy regulations, we recommend retaining this component as a classic in-house system of record. Given sufficient focus on GDPR compliance, the deployment of suitable data anonymization technology at least on data clearly related to personal information is recommended.

4.2.2 GWR and Siemens HEMS management components

Could be transformed to hybrid cloud component – requiring the cooperation of the solution component providers.

4.2.3 Internal data importers

Clear private cloud component particularly to ensure streamlined adherence to privacy regulations.

4.2.4 External data importers

Hybrid cloud component possibly co-located either with data warehouse or weather data originator (if either indeed is deployed cloud-based).

4.2.5 FIP Platform

Hybrid cloud component; could be deployed publicly if proper separation between personal data and devices can be achieved, e.g., using data anonymization technology at least on data clearly related to personal information, e.g., power usage patterns in houses permitting data analysts inference on private lifestyle decisions (vacation absences, work schedules, etc.).

4.2.6 Aggregators

Hybrid cloud component; could be deployed publicly if proper separation between personal data and devices can be achieved, e.g., using data anonymization technology at least on data clearly related to personal information, e.g., power usage patterns in houses permitting data analysts inference on private lifestyle decisions (vacation absences, work schedules, etc.).

4.2.7 Market clearing house

Hybrid cloud component: Must run identically within private cloud environment for maximum security, but could also be deployed in public cloud, e.g., for further trials and actual deployments.

5 References

[1]: https://www.ibm.com/blogs/bluemix/2017/01/cloud-computing-history/

[2]: http://www.redbooks.ibm.com/redpapers/pdfs/redp4873.pdf

[3]: https://developer.ibm.com/dwblog/2016/what-is-docker-containers/

[4]: https://domino.watson.ibm.com/library/CyberDig.nsf/papers/040F7F7D5E62F0E58525804500433733/$File/rc25625.pdf

Read more at www.ecogrid.dk