Lauri Niemi

Cloud Solutions in Non-software Company

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

8 March 2020

Author	Lauri Niemi
Title	Cloud Solutions in Non-software Company
Number of Pages	32 pages
Date	8 March 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Esa Ravantti, IT Manager Janne Salonen, Principal Lecturer

In this study, a cloud solution was created for a company that does not mainly produce software. The study was based on the documentation created by the service provider and the experience gained from the user perspective. The solution is made for Ramirent Finland and is based on the needs of the company.

The thesis generally describes the different services of the cloud solution in order to be familiar with different services when designing a new platform. As the main outcome of the study, guidance was provided on deploying a cloud solution and, in the case of Ramirent, modifying an existing solution to fit best practices.

As a result, it was found that it is difficult for a small development team to make architectural changes. As a result, it is advisable to spend sufficient time developing a good solution when deploying the platform. These architectural challenges are typically encountered on experimental platforms.

Keywords	AWS, cloud solution, cloud computing, cloud architecture

Tekijä Otsikko Sivumäärä Päivämäärä	Lauri Niemi Pilivipalveluratkaisut yrityksessä, joka ei pääasiallisesti tuota ohjelmistoja 32 sivua 08.03.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Software Engineering
Ohjaajat	IT manager Esa Ravantti Tutkintovastaava Janne Salonen

Insinöörityössä luotiin pilvipalveluratkaisu yritykselle, joka ei pääasiassa tuota ohjelmistoja. Työn pohjana käytettiin palvelun tarjoajan luomaa dokumentaatiota sekä käyttökokemuksen myötä syntynyttä kokemusta ratkaisun kokonaisuudesta. Ratkaisu on tehty Ramirent Finlandille ja se pohjautuu yrityksen tarpeisiin.

Työssä kuvailtiin yleisesti pilvipalveluratkaisun eri osa-alueita, jotta ratkaisua suunniteltaessa uuden alusta erinimiset palvelut olisivat tuttuja. Työn pääasiallisena tuotteena luotiin ohjeistus pilvipalveluratkaisun käyttöönotosta, sekä Ramirentin tapauksessa olemassa olevan ratkaisun muokkaamisesta parhaiden käytäntöjen mukaiseksi.

Lopputuloksena todettiin, että arkkitehtuuri muutosten tekeminen on vaikeaa pienellä kehitystiimillä. Tämän tuloksena alustaa käyttöönottaessa on suositeltavaa käyttää riittävästi aikaa hyvän ratkaisun kehittämiseen. Tämän kaltaiset arkkitehtuuriset haasteet kohdataan tyypillisesti juuri kokeilumielessä aloitetuilla alustoilla.

Avainsanat	AWS, pilvipalvelu, pilviprosessointi, pilviarkkitehtuuri

Contents

List of Abbreviations

AWS	Amazon Web Services
IAM	Identity and Access Management
REST	Representational State Transfer
S3	Simple Storage Service
ERP	Enterprise Resource Planning
PWA	Progressive Web Application
API	Application Programming Interface
EC2	Elastic Compute Cloud
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
RDS	Relational Database Service

1 Introduction

Many companies operating in more traditional industries have now started to develop their own digital solutions. Some companies build up a development team and others buy custom solutions outsourced. Nevertheless, the outcome is the same. They need an easy and cost-efficient way to create, publish and maintain the digital solutions to customers or employees.

In the present project an example case was made to a company that does not produce software as their main product. The project was commissioned by Ramirent Finland Oy. Ramirent Finland Oy is a leading Finnish service company in equipment rental for construction and other industries. Ramirent Finland is part of Ramirent Group which is now owned by a French family company Loxam.

Ramirent has started to create their digital solutions for optimizing internal processes and customer needs. Ramirent has multiple platforms for applications and multiple platform owners. Amazon Web Services, shortly AWS is fully managed by Ramirent Finland. Beside AWS, Ramirent has locally hosted servers, Enkora hosted services and Microsoft Azure cloud service, which is the newest comer. Azure is partly managed by Advania and partly by Ramirent Finland.

The scope of the project was restricted to Ramirent Finland's case with a small development team focusing mainly on AWS. As a result of this project it is expected to see improved security, cost allocation and maintainability of applications. More uniform and controlled access to solutions helps to debug issues and maintain overall security and reliability.

2 Cloud Solutions

Cloud solutions are a selected set of available cloud services provided by cloud service providers such as Amazon Web Services or Google Cloud Computing. Cloud solutions consist of cloud architecture in one or more cloud platforms. There are as many cloud solutions as there are cloud users.

The most recognizable names on the cloud service providers are Amazon Web Services, Microsoft Azure and Google Cloud Engine. Amazon Web Services (AWS) is currently the leading cloud platform. AWS has over million customers around the world and its almost 200 different cloud services are trusted. [1]

Below are introduced the most commonly and widely used services. These services can be used in different scopes ranging from mobile app to machine learning task. In May 2019 AWS has 169 products to offer and AWS is far ahead everyone else [2]. This huge variety of different services can be overwhelming but here are the most important ones.

2.1 Cloud Servers

Cloud servers can be roughly divided into two different types. Those are traditional servers that run around the clock and trigger-based functions that run on demand. On demand functions are a cost-efficient way to build a wide range of cloud computing service that can be triggered various ways. Triggers can be change in database, traditional REST call, change in storage or by some other cloud service. Traditional servers are great and cost-efficient when the use is heavy, and the processing need is continuous. In the AWS infrastructure on demand functions are called Lambda functions and traditional servers are called Elastic Cloud Compute or EC2. [1]

Lambda functions are an easy and cost-efficient way to solve event-based processing. Functions can be written in Node.js, Java, Python, Go, C#, Ruby or PowerShell [3]. Example use cases where Lambda functions could be used are simple REST services or data processing on background.

Amazon Fargate provides a serverless approach using containers. Amazon Fargate allows to build using traditional methods used to build on servers, but paying as serverless, only when instances are used. This is an easy way to lower costs of old applications and ease transformation from old servers to new cloud infrastructures. Scaling is done automatically to provide the best experience without provisioning over capacity on silent times. [4] Amazon Fargate is the easiest way to approach to take a step into a serverless world when making transformation using existing software.

2.2 Database

In a cloud solution environment, databases can be found in different shapes. Databases can be hosted in the traditional way in servers or in database services such as DynamoDB. These services are secure by default and all access to data is permitted. AWS provided database services are easy to use and very maintainable. These solutions are suited for smaller development teams because they need very little managing and can be run for long periods of time without maintain. These services are billed ‘as you go’. [5].

2.3 Storage

Storage solutions in cloud services are usually created to look and function similarly to client-side cloud storages such as Google Drive or Dropbox. In addition to a similar user interface there are multiple different use cases or configuration options. These storage solutions can be easily defined as public, block all access or everything between. The storages are easy to maintain and secure to use. In the AWS infrastructure one example is Simple Storage Service S3 which is convenient for website hosting because it has built in website hosting capabilities and is cost efficient. There are also multiple other storage solutions for archiving or for very large data amounts. [6]

2.4 Content Distribution

Cloud service providers typically offer multiple solutions to provide content to user or application. Content delivery network, CDN is like a door to a house. APIs, websites or data can be accessed through one secure portal. Through CDN services can be distributed to users securely and efficiently. In AWS CDN is called CloudFront and it is often used to provide website or application that is hosted from S3 bucket. For example, CloudFront caches site for fast loading times and provides custom domains for that endpoint. [7]

2.5 Access Management

Access management is an important part of cloud services. Access to cloud services is done using roles. Roles can be given to user, group or service. Roles are easier to manage than IP address rules or username password type access to classic server.

In the AWS infrastructure, IAM, Identity and Access Management is a service where all access and identification to services is managed. For example, access to AWS console and command line interface access is done using IAM. IAM is a key element in every interaction between AWS services. By default, services cannot communicate between each other without granting access rights.

3 Current Status of Cloud Solution

Ramirent uses AWS as the selected service provider. In the new Enterprise Resource Planning (ERP) project Microsoft Azure comes to the side. Ramirent uses AWS in group level projects as well as in the country level. AWS was selected because it has a relatively cheap pricing, good documentation and locations near the countries where Ramirent operates.

Ramirent’s AWS overall architecture is quite loose. Ramirent Group have their own solutions and they for example provide only a website for all countries. Ramirent Finland's own AWS account is not controlled by Ramirent Group and they do not even share that many common components or architecture. There are a few duplicates in services and this overall solution is not as cost efficient as possible.

3.1 Applications

Applications that are served in Ramirent Finland's account are mainly web applications. The meeting room reservations server is only exception to this. Table 1 shows Ramirent's current application status in AWS and cloud services that are in use.

Table 1. Raiment's applications and used AWS services

Name	**Devel-oper**	**Type**	**API Gateway**	**S3**	**Lambda**	**Cloud-Front**	**Dyna-moDb**	**RDS**	**EC2**	**Inte-gra-tion**
Ra-miSmart App	Digia	PWA	X	X	X	X	X			Loca-tion, ERP, Auth.
Rami-Forms	Ramirent Finland	Web app	X	X	X	X	X			Auth.
Authenti-cation	Server Side	API / Web app	X	X		X		X	X	
eTeline	Server Side	API / Web app	X	X		X		X	X	Auth.
Raimo	Digia	An-droid	X	X	X		X	X		ERP

Raimo User management	Digia	Web app	X	X	X	X	X			
Evoko	Ramirent Finland	Room reservation system							X	
Orders-API	Ramirent Finland	API	X	X	X		X			

RamiSmart App

RamiSmart App is developed by Digia which is one of Ramirent Finland’s partners in digital development. RamiSmart App is Progressive Web App, PWA for customers and salespersons to order and control orders for worksites. Application is serverless and it uses API Gateway, S3, Lambda, CloudFront, CodePipeline, DynamoDB, React and GraphQL.

RamiForms

RamiForms is an in house developed web application to smooth processes and collect valuable data. It contains management and user site and is usable also via API. Application is serverless and it uses API Gateway, S3, Lambda, CloudFront, CodePipeline, DynamoDB, React and REST.

Authentication

Ramirent created their own authentication platform for the needs of the applications. Authentication platform controls users and their roles. Currently eTeline, RamiSmart App and RamiForms are using authentication service. Authentication uses EC2, S3, CloudFront and React. Authentication platform is created by Server Side Oy.

eTeline

eTeline is a scaffolding management web application. In this app users can inspect and manage inspections of their scaffoldings using QR codes. eTeline uses EC2, S3, CloudFront and React. eTeline is created by Server Side Oy.

Cloudinary tool

Cloudinary tool is a temporary web application for product image management in Cloudinary. It is made for easier management of products with multiple images and documents. Cloudinary contains 175 000 image and document which is divided for about 60 000 products. Cloudinary tool uses S3, API gateway, Lambda, Cloudinary and HTML, JavaScript, CSS combination. Cloudinary tool is developed in house and is deprecating when new Product Information Management system, PIM is ready to take control of product images and documents.

Raimo

Raimo is a mobile application for product management. It is connected to the ERP system and greatly simplifies product management. Raimo is developed by Digia and it uses API gateway, S3, Lambda, RDS and React native

Raimo user management

Raimo user management exists because Ramirent’s Authentication platform was not in use that time for common solutions. That is why the Raimo application does not use Ramirent’s own authentication platform to control users for now. Raimo has its own user management web application. This application is developed by Digia. Raimo user management uses API gateway, S3, RDS and React.

Evoko

Evoko is a room reservation service. This application manages room reservation tablets and Outlook calendar connection to book meetings. This is a ready-made solution and contains the image of instance to deploy on server. Evoko uses one EC2 instance.

Orders-API

Orders API is an interface for new orders process. This API handles all incoming orders from different platforms and simplifies the process of ordering. This application stores data about the order and generates a pdf document from the order. Orders-API uses API Gateway, S3, Lambda, DynamoDB, CodePipeline, CloudFormation and Simple Notification Service. This application is developed by Ramirent.

3.2 Control of Environment

The overall control is currently not in a very good condition. Ramirent Finland's director of development owns the main account. Ramirent's account is connected directly to persons email address. This is not a good solution in the long term because if that person leaves Ramirent there is a possibility that the email address is deleted accidently before the linked email is changed and billing can be interrupted.

There are no administrators for the AWS. Someone creates something when someone asks for it. Roles are given for the whole Ramirent account which creates a significant risk for misbehavior or human error. Every user has access to every project and its resource.

3.3 Other Platforms

Microsoft Azure came into picture during the project. Azure is managed partly by Ramirent Finland and partly by their partner Advania. Azure is used to provide SQL database for ERP data. This data can be linked directly to Microsoft Power BI to create data analysis about business. Because of this native implementation to Power BI Azure is used over the existing AWS.

Traditional server capacity still exists and will continue to do so for a while. This capacity is currently used by the current ERP system Rami and its reporting servers. This is being actively reduced, though, and when the new ERP system is fully running it will be reduced to minimum. Locally hosted server capacity requires more labor to maintain and that is why is not as cost efficient as Cloud computing.

4 Cloud Solution for Ramirent

This part of the thesis follows the AWS documentation as to the best practices of architecture [8] with specific solutions to Ramirent or any other non-software company. AWS makes it possible to have multiple different architecture solutions. Figure 1 shows a diagram of a suitable architecture for a small company. The master account is divided into organizational units OU's. OU's are project based and every OU can contain different stages of project. For example, OU 1 contains developing stage of project 1 in one account and production stage of project 1 in other account.

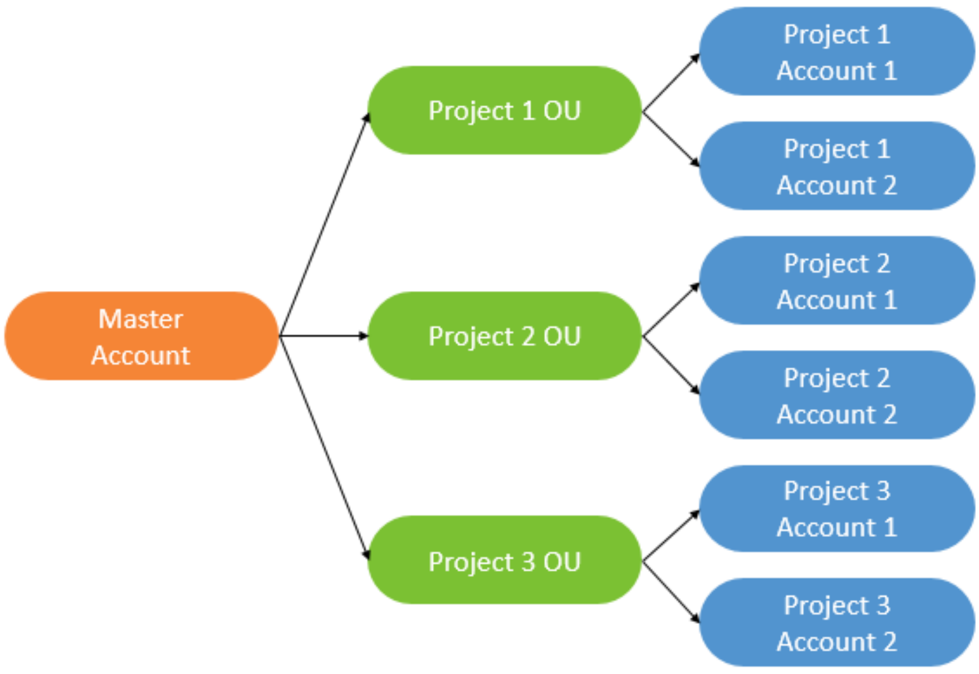


Figure 1. Account structure

This account structure suits Ramirent's solution best. Depending on the application this can be streamlined even further to only one account per application combining the development and production environments. The streamlining reduces the account creation load at project start.

4.1 Organizations and Accounts

The AWS infrastructure contains at least one master account. The master account is used to control billing and organizations. In some circumstances the master account could be used for developing and production but when multiple applications are running with different developers it is highly recommended that the architecture is divided into organizations and accounts. Organizational units are a layer between accounts. An account can contain multiple organizational units and an organizational unit can contain numerous accounts. Figure 2 illustrates the relationship of cloud services in the account and how users and roles are used to manage this infrastructure. [9 p. 8- 11]

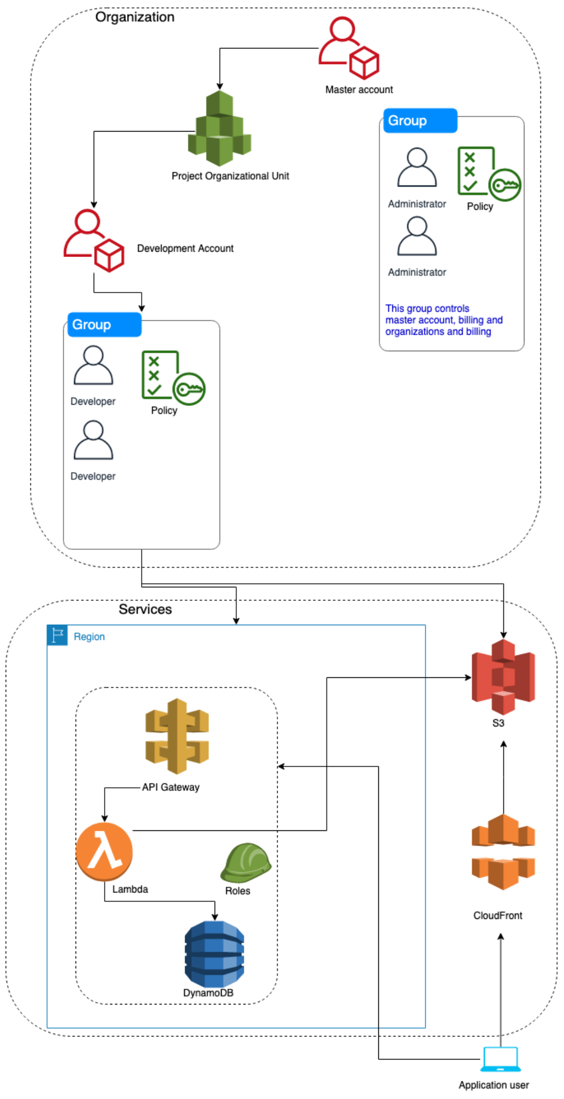


Figure 2. AWS account structure with roles and services

When following an infrastructure described in Figure 2 the products in the AWS are safe from human errors and misbehaving in the service. One of the biggest risks in the cloud services are a stolen laptop.

4.2 Cost Allocation

The control over costs is increased when the architecture is divided into organizations and accounts. When using only the master account to develop and host applications, detailed per application billing details are difficult to get. One option is to tag every resource with describing tags, but it leaves space for human errors and it is hard to maintain. When tags are used to identify resources and tags are missing it is very hard to allocate where the solution belongs and if it is still in use. [10]

When creating a new organization for every project and account for every stage of the project the company can achieve precise cost information on how much the development of the application costs and how much the production version is creating bill. This can be easily simplified if the project has only one account, then account maintaining does not become a load in small development teams. [9. p. 7]

4.4 Security

Often companies that do not make software as their main product have small software development teams and part of the software development is bought outside. When multiple companies and developers are working in multiple projects managing all users is hard. Often in a software company, employees switch between different projects and sometimes developers change. When projects are divided into separate accounts it is easier to maintain the roles and users in such partition. For example, if a user's laptop is stolen and a hostile user gets access to the account it only affects one project. User management can also be outsourced to the company developing that account. This way the load on small development teams can be eased which in turn improves productivity and reliability [11].

The account structure also gives protection from misused application level access rights. For example, in AWS every cloud service action to other must be authorized with the Identity and Access Management service IAM. The IAM roles can be easily overpowered and grant access to unwanted parts of cloud solutions. With the application level account structure this is minimized to most often just to the right level of security.
Account access can be granted for partners by linking the partner’s own AWS account to Ramirent’s account. This way all access right can be removed at once from a partner without any risk of leaving some access rights. This is obviously not convenient if the partner does not have their own AWS account or the partner company is so big that their AWS infrastructure is managed too strictly and heavily for this use case and giving user-based access is an easier solution.

A part of the security is that just enough access rights is given to certain tasks. For example, if the development manager is logged in to root account to review the month’s current bill. In this case the manager needs only view access to services so they can dig deeper into services and have full access to billing details.

4.5 Owner

Every process should have an owner. The owner owns the system. This means that that person is responsible for decisions regarding the given process. When the process is owned by a single person, decisions can be made quickly without heavy bureaucracy. The owner is responsible for the state of the process whether it is making coffee in the morning or maintaining a cloud platform. [12] The owner should be always one person, because shared responsibility is nobody’s responsibility [13 p. 65]. In a cloud platform case, the owner should be selected before even selecting the cloud service provider. The owner should have an overall vision about the solution needs and technology, especially at the beginning of new platform decisions that have long term affect are made. The owner can be assigned later, but the sooner the better.

4.6 Root Account Email and Name

An email address that is linked to a root account should never be linked to a person directly. The best way to attach an email address to a root account is to use some sort of an email group or mailing list. This way it is unambiguous that this email account is attached to a root account. Root account access rights should not be used after initialization. Also, when the account is linked to an always existing email address it is not dependent on individuals and whether that person who started the use of the cloud platform even works at the company. The name of the root account should be convenient and indicate that it is a root account. A root account can be used to access all billing or maintain organizations and accounts.

4.7 Sandbox Account

A sandbox environment is important when developing new applications. A sandbox account should be created and used when prototyping new technologies and testing new applications before they are pushed into new project related accounts. In a sandbox account every setting can be opened without risking any security vulnerabilities as long as data used in there is appropriate. This way of open platform prototyping is fast and required environment setups are easily tested without waiting for tickets to go through multiple acceptance procedures in the ticketing system. When the developer knows what to ask precisely it fastens the overall process of development.

4.8 Standards for Processes

Standards create longevity. When processes are standardized debugging becomes easier. A new project requires a new AWS account. For security and maintainability reasons every project should have their own account.

Create account

AWS accounts are created from My Organizations page using master account._Figure 3 describes how to access My Organizations page.

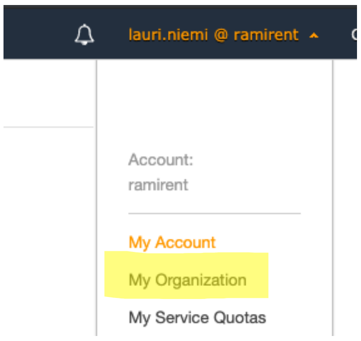
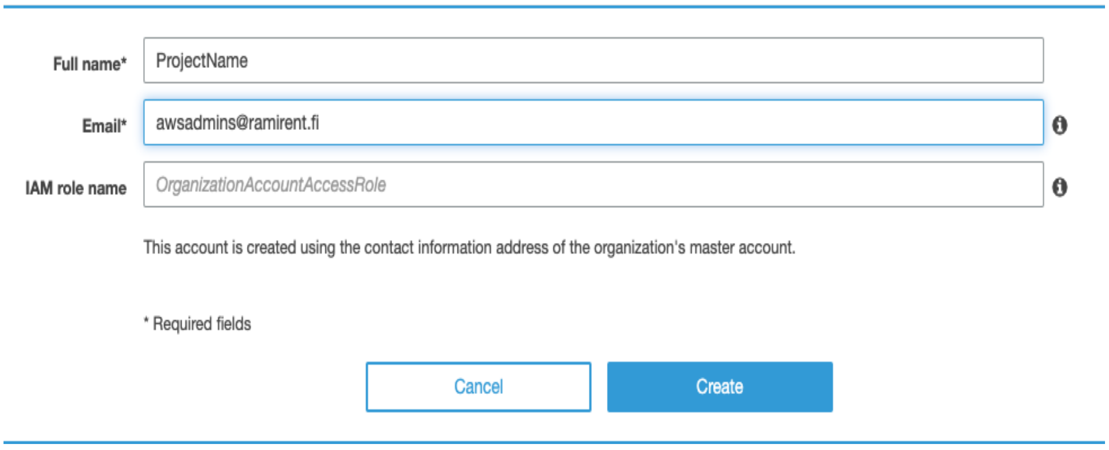


Figure 3. Screenshot of AWS console menu where My Organizations are accessed.

Full name is the name of the account. Account is created for project that is why logical naming for account is ProjectName i.e. RamiForms. In Ramirent case <account_name>.c ns@ramirent.fi email is used for the account. It is important to make sure that the email account can receive emails outside the organization, with Microsoft outlook the default is "don't allow emails outside of organization". When using this common email address individuals are not linked directly to the account and an employee leaving company does not affect the AWS usage. IAM role name can be leaved blank. Figure 4 shows an example of this step.



Full name*	ProjectName
Email*	awsadmins@ramirent.fi
IAM role name	OrganizationAccountAccessRole

This account is created using the contact information address of the organization's master account.

* Required fields

Figure 4. Screenshot of account naming process.

Sign in as root user

Root account is used to view from Organizations page list of Account ID:s and copy the new identifier. New id can be used to sign-in in console.aws.amazon.com. Figure 5 shows an example of this process.

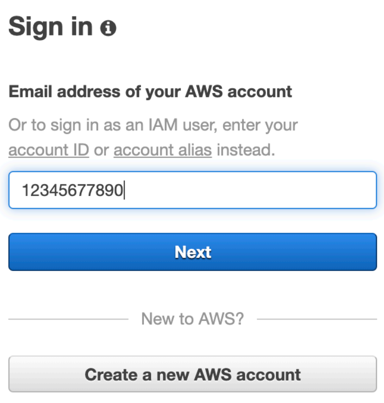

Sign in
Email address of your AWS account
Or to sign in as an IAM user, enter your account ID or account alias instead.
12345677890
Next
New to AWS?
Create a new AWS account

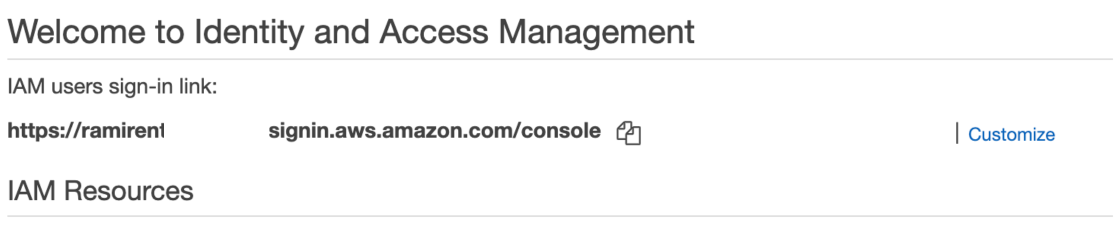
Figure 5. Screenshot of sign in selection to AWS console.

The “Forgot password” feature is used to create a new password for the account. An email will be sent to the email address of the account. If the email does not arrive in a few minutes, email settings could be too restricted. The email will contain a link to reset password and, in this case, give the root account a password.

The root user of the account should not be used after creating the first administrator user. Accounts root credentials are only used to remove account or transfer the ownership to another user.

Sign in link creation

The sign in link is important when using multiple accounts. With sign in links users can easily change between accounts and projects. Sign-in links can be created in the IAM service. For convenience, the same sign-in name for the link and account name is used. Figure 6 shows an example of how to customize the sign-in link.


Welcome to Identity and Access Management

IAM users sign-in link:

https://ramirent signin.aws.amazon.com/console | Customize

IAM Resources

Figure 6. Screenshot of IAM service where sign-in link can be modified.

Creating group

Groups are used to manage users' access. A typical need for groups consists of three groups. Groups are account dependent and are accessed through a specific account. The following list explains these three groups.

1. Account_Admin - controls account. Access to billing and roles
2. Account_Developer – access to typical AWS services, console and programmatic
3. Account_ViewOnly – access to view data. Useful for integration use when only view is

Figures 7 and 8 describe how to setup group policies.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Figure 7 Policy for Admin group

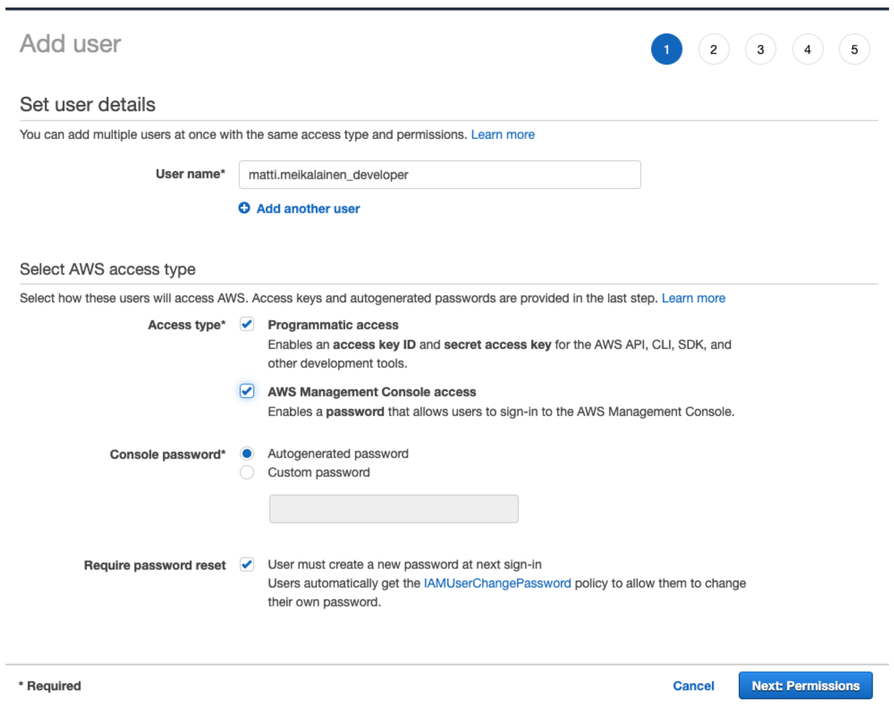
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "sns:*",
        "rds:*",
        "apigateway:*",
        "s3:*",
        "dynamodb:*",
        "cloudformation:*",
        "wellarchitected:*",
        "autoscaling-plans:*",
        "iam:*",
        "cloudfront:*",
        "ses:*",
        "cloudwatch:*",
        "lambda:*",
        "ec2:*"
      ],
      "Resource": "*"
```

Figure 8 Policy for developer group

When using an account for every project the user can have more rights to the environment without affecting security. Every use case should have its own user, for example developer, admin and view only.

Basics

The user name should be firstname.lastname_group to ease users' login and maintainability. Programmatic access means that the user has access to the AWS account using development tools. This must be selected. In most cases the console access is also very useful for the user. In a project-based account structure console access can be granted for users. Figure 9 shows how to create use in the AWS console.



Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* matti.meikalainen_developer

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* [✓] **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

[✓] **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ● Autogenerated password
○ Custom password

Require password reset [✓] User must create a new password at next sign-in
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

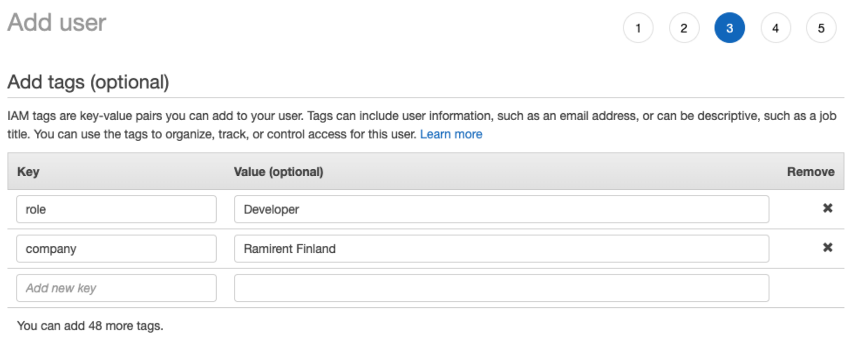
* Required

Cancel Next: Permissions

Figure 9 Screenshot of user creation

Grouping and tagging

Users are added to groups to give access rights. The user should be tagged to improve maintainability. Tags are not case sensitive but upper and lower cases will be stored. Tags should contain the role and company. Figure 9 shows how tags are added to the user.



Add user

1 2 3 4 5

Add tags (optional)

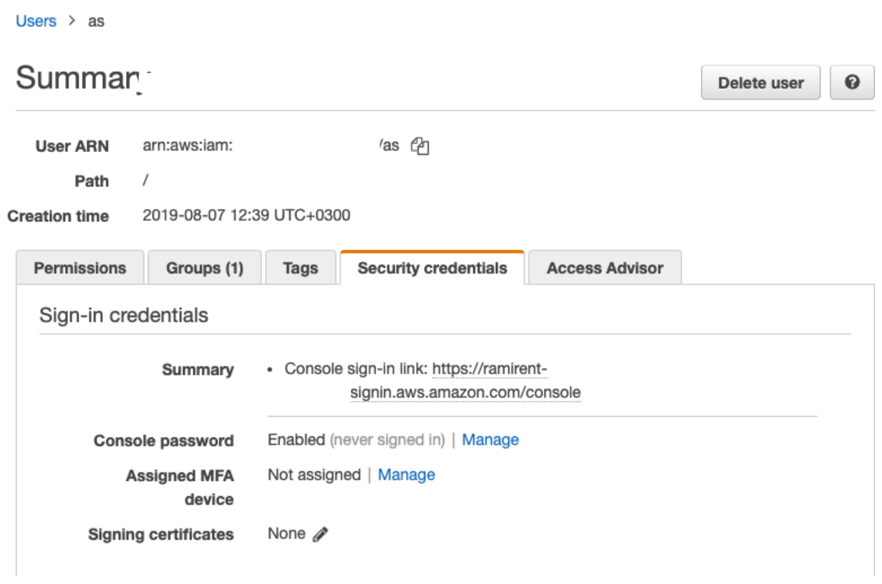
IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
role	Developer	✖
company	Ramirent Finland	✖
Add new key		

You can add 48 more tags.

Figure 9. Screenshot of tagging example.

Users credentials can be found under user in the Security credentials tab. Figure 10 shows an example where users' sign-in link, username and temporary password can be found.



Users > as

Summary

Delete user

User ARN	arn:aws:iam: [redacted] /as
Path	/
Creation time	2019-08-07 12:39 UTC+0300

Permissions | Groups (1) | Tags | Security credentials | Access Advisor

Sign-in credentials

Summary	• Console sign-in link: https://ramirent-signin.aws.amazon.com/console
Console password	Enabled (never signed in) | Manage
Assigned MFA device	Not assigned | Manage
Signing certificates	None

Figure 10. Screenshot of menu where the user's temporary password can be created.

Admin has full access and this policy could be given to a subcontractor project leader to grant access to create new users and maintain the project. The developer group has access rights to most typical services in Ramirent's environment. This instruction is created as a result of the present project.

By standardizing the available cloud services, maintenance of the applications becomes easier. Cloud platforms are always changing and when default services and technologies are selected, changes to these services can be followed and possible issues fixed. Technology standardizing eases developer changes. When technologies are always similar less time is spent on learning new technologies.

One standard for processes is to use CloudFormation templates. With templates it is possible to write down building instructions to cloud platform. This code can be pushed to a git repository which is web hooked to code pipeline that builds the cloud solutions automatically. This way applications are transformable in case of an architecture change or in case of a platform failure. AWS has also an open source Serverless Application Model (SAM) that transforms natively to a Cloud Formation template, but it is also common description that can be used in other cloud service providers. Cloud Formation is a service that builds an application from a recipe to a cloud platform. Cloud Formation can be combined with GitHub hooks to create continuous integration and continuous development pipelines. The same template works as documentation to an API or application itself because it is written in JSON or YAML which makes it human readable. Below in Listing 1, is an example of template.yml that corresponds to earlier description. [14 p. 7-11]

```
Transform: "AWS::Serverless-2016-10-31"
Resources:
# --- LAMBDA --- #
  AddOrder:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs10.x
      Timeout: 30
      CodeUri: ./lambdas/add-order/
      Policies:
        - DynamoDBCrudPolicy:
            TableName:  orders
        - S3CrudPolicy:
            BucketName:  orders
      Environment:
        Variables:
          BUCKET_NAME:  orders
          TABLE_NAME:  orders
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /add-order
            Method: post
  CreatePDF:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs10.x
      Timeout: 30
      CodeUri: ./lambdas/pdf-converter/
      Policies:
        - DynamoDBCrudPolicy:
            TableName: orders
        - S3CrudPolicy:
            BucketName:  orders
        - SESCrudPolicy:
            IdentityName: noreply@exaple.com
      Environment:
        Variables:
          BUCKET_NAME:  orders
          TABLE_NAME:  orders
      Events:
        ObjectCreated:
          Type: SNS
          Properties:
            Topic: !Ref OrderSNSTopic
# --- SNS --- #
  OrderSNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: New S3 event
      Subscription:
        - Endpoint:
            Fn::GetAtt:
              - CreatePDF
              - Arn
          Protocol: lambda
      TopicName: AddOrderTopic
# --- DYNAMODB --- #
  OrdersTable:
    Type: AWS::Serverless::SimpleTable
    Properties:
      TableName:  orders
      PrimaryKey:
        Name: id
        Type: String
    ProvisionedThroughput:
      ReadCapacityUnits: 5
      WriteCapacityUnits: 5
    Tags:
      Project: test
      Owner: Ramirent
# --- POLICIES --- #
  OrderSNSTopicPolicy:
    Type: AWS::SNS::TopicPolicy
    Properties:
      PolicyDocument:
        Id: OrderSNSTopicPolicy
        Version: '2012-10-17'
        Statement:
          - Sid: orderSnsTopicId
            Effect: Allow
            Principal:
              Service: s3.amazonaws.com
            Action: sns:Publish
            Resource: "*"
      Topics:
        - Ref: OrderSNSTopic
```

Listing 1. Template.yaml file for CloudFormation

In the template.yml file is created a simple API that adds order to S3 bucket and DynamoDB table. This event triggers the pdf-creator that creates a pdf document to the same S3 bucket and sends the file using Simple Email Service (SES). Simple Notification Service, SNS, is used to transfer the trigger event from S3 to Lambda.

4.9 Microsoft Azure

Microsoft Azure is now part of Ramirent’s cloud solution. This platform is used to connect with other Microsoft services including the new ERP system. Azure has been part of the cloud solution in the background from the start of Microsoft Office 365 use. Office 365 requires at least the hybrid Active Directory solution to work [15 p. 13]. This means that part of user access rights and device management is maintained in traditional servers and partly in Azure. During this project Azure has been attached to the overall development cloud solution. In future Azure will be handling part of ERP integration and analytics due to its native integration with Microsoft solutions, Dynamics and Power BI.

5 Results

When a cloud platform is already in use, changes to the architecture are not easy to handle. In Ramirent's case, the development team is small, consisting of only three persons, i.e. manager, developer and process specialist. During this project a new team member was added to the team to own and maintain cloud platforms as a Cloud Solution specialist. This was an improvement and has been already transforming pressure of the environment from the developers' shoulders.

Ramirent Finland has plans to shift part of the applications to a new architecture. New applications will be implemented to the new account structure. Current applications that are easy to move will be moved during the development pause. The first application to move to the new account will be the Evoko room reservation system. This application was selected due to its simple one instance footprint and effecting only internal users at the headquarters. New projects will be following AWS administrator instructions about account creation.

The start of Azure usage in Ramirent has been more controlled and planned due to the current project and it was owned by the cloud solution specialist at the day one. This move was timed perfectly when the new Cloud Solution specialist arrived. Azure is now fully owned by the Cloud Solution specialist and Advania is helping in the technical matters.

Root account related email is now changed to a non-dependent email address. During Microsoft One Drive renewal email addresses were flushed and Ramirent has to keep eye on the most important email addresses such as this. Also, the name of the root account is now changed to a more convenient name.

The sandbox account which was created to demonstrate this project's content is still in use and works perfectly to prototype and test securely new applications and technologies without creating a risk of affecting production applications. The sandbox account is used to develop one new application and also to test the guidance in action. This account has currently only one actual user.

6 Conclusion

Overall, the project can be considered successful. Despite oft this project having been created for Ramirent Finland's case hopefully other traditional companies now figuring out their own cloud solutions will find the study helpful. Cloud service providers' own documentation can be heavy to read when comparing platforms to each other and designing suitable cloud architecture.

Architecture changes are hard to make when the platform is in use. This means that the initial architecture and planning should be invested on at the beginning. Often a platform is started fast because the end result, the first application is in mind and the target. That is why the platform should have an owner right from the beginning. The owner thinks more about the architecture and cloud solution than the end result, the application.

In future Ramirent should measure account-based usage and cost efficiency of the application reflected to its business improvement or market value. With these parameters the development recourses can be adjusted according to cost saving improvements or increased sales.

References

1 Cloud computing with AWS. 2019. Internet source. Amazon Web Services. <https://aws.amazon.com/what-is-aws/?nc1=f_cc> Read 25.10.2019

2 Robbins, Ken. 2019. How Many AWS Services Are There? Internet source. <https://medium.com/cloudpegboard/how-many-aws-services-are-there-51dda44fa946> 22.5.2019. Read 26.10.2019

3 AWS Lambda FAQs. 2019. Internet source. Amazon Web Services. <https://aws.amazon.com/lambda/faqs/> Read 25.1.2019

4 AWS Fargate Serverless compute for containers. 2019. Internet source. Amazon Web Services. < https://aws.amazon.com/fargate/> Read 26.1.2020

5 Amazon DynamoDB. Internet source. Amazon Web Services. <https://aws.amazon.com/dynamodb/> Read 26.10.2019

6 Amazon S3. Internet source. Amazon Web Services. <https://aws.amazon.com/s3/> Read 26.10.2019

7 Amazon CloudFront. Internet source. Amazon Web Services. <https://aws.amazon.com/cloudfront/?nc2=type_a> Read 26.10.2019

8 Architechting for cloud. 2018. Internet source. Amazon Web Services.<https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf> Read 15.8.2019

9 AWS Organizations user guide. Internet source. Amazon Web Services. <https://docs.aws.amazon.com/organizations/latest/userguide/organizations-userguide.pdf> Read 15.8.2019

10 Software Engineering Team Lead at BBC. 2019. Conversation 16.10.2019

11 Samson Sunday Adaramola. 2012. Internet source. Job Stress and Productivity Increase. <https://pdfs.semanticscho-lar.org/37e0/e714280f06e2c32d148651e0f078f013952c.pdf> Read 25.10.2019

12 Tuoteomistajan pikastartti. Internet source. City of Helsinki. <https://kehmet.hel.fi/organisaatio/tuoteomistaja/> Read 26.10.2019

13 Sahi, Anniina. 2013. Kaikkien ja ei kenenkään vastuulla. Internet source. <https://jyx.jyu.fi/bitstream/han-dle/123456789/42065/1/URN%3ANBN%3Afi%3Ajyu-201309042227.pdf> Read 26.10.2019

14 AWS Well-Architected Framework. 2019 Internet source. <https://d1.aws-static.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf> Read 27.10.2019

15 Haakanen, Joonas. 2019. Internet source. Windows directoryn ja Active directoryn integraatio.<https://www.theseus.fi/bitstream/han-dle/10024/171600/Haakanen_Joonas.pdf?sequence=2> Read 24.10.2019