



Cloud Builders' Day

Cloud Technology Workshop

Author:
Tomasz Stachlewski (stachlew@amazon.pl)

Table of Contents

Introduction.....	3
Lab 1 – Preparing Cloud Environment	4
Lab 2 – Autoscaling and IaaS.....	13
Lab 3 – Developer Tools. CD in practice	18
Lab 4 – Containers in cloud	20
Lab 5 – Collecting data in real-time	26
Lab 6 – Analytics – Hadoop in practice	31
Lab 7 – Business Intelligence in Cloud	39
Lab 8 – Building serverless app with AI capabilities	40
Bonus Lab – Neural network and digits recognition	49
Cleaning.....	51

Introduction

This document contains instructions for 9 labs. Your mission, should you choose to accept it, is to complete the labs and understand what can be achieved thanks to the Cloud and AWS. During this full day workshop, you will be working with many different IT technologies that you can come across during your daily work. We will be discussing topics such as *Artificial Intelligence*, *Containers*, *Serverless*, *Autoscaling*, and many more. If you do not have any experience with AWS, then don't worry: all the more reason as to why this workshop is for you. Thanks to this document you will be able to solve the labs step by step - so, let's start!

Lab 1 – Preparing Cloud Environment

Your first lab will focus on preparing your cloud environment and will take around 20 minutes. You need to complete this first lab, otherwise it might be harder for you to complete the following labs.

Requirements:

- Prepare your cloud environment: thanks to which you will be able to manage your cloud infrastructure.
- Launch your first cloud server using AWS CLI.

Services that you will be using in this lab:

- *AWS Identity and Access Management (IAM)* - AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions that allow or deny their access to AWS resources. IAM is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS services by your users. You can find more information here: <https://aws.amazon.com/iam/>
- *AWS Cloud9* - AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. You can find more information here: <https://aws.amazon.com/cloud9/>
- *Amazon EC2* - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. You can find more information here: <https://aws.amazon.com/ec2/>

- **AWS Command Line Interface (CLI)** - The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts. You can find more information here: <https://aws.amazon.com/cli/>

Steps:

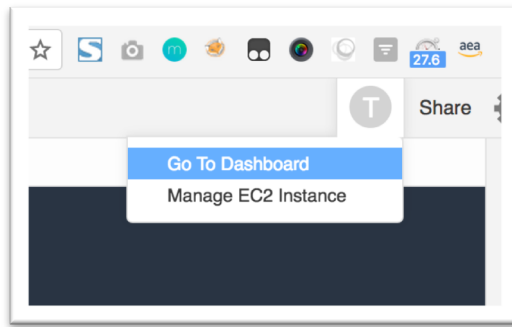
- 1) Log in to AWS Console
- 2) Make sure that you are in EU **Ireland** region. The Ireland region will be used for the duration of the workshop.
- 3) Open the AWS **Cloud9** service.
- 4) Click on **Create environment** button to create your first cloud environment.
- 5) In the first step of the wizard provide name of your environment – for example **CloudyEnv**. Then select **Next step**.
- 6) Do not change any of the **Configuration settings**: leave them as set by default. Select **Next step**

*Note: In **Configuration settings** you choose the size of server on which your environment will be hosted. For the purpose of this workshop, a **t2.micro** instance will be more than enough. We will also leave the **cost-saving settings** alone; by default the cost-saving settings will auto-hibernate an instance after 30 minutes of inactivity.*

- 7) On the **Review** page, review the **Environment name and settings** for the Cloud9 service and select **Create Environment** to start the creation of the environment.
- 8) It will take approximately 2-3 minutes for AWS Cloud9 to complete the creation your environment. Open another AWS Console session in another or new browser tab.

Alternatively wait until AWS Cloud9 completes and use the **Go To Dashboard** option on right side of the Cloud9 menu bar: **Go To Dashboard** opens another AWS Console session in a new browser tab – this option is demonstrated in the Cloud9 screenshot below:

Cloud Builders' Day
Cloud Technology Workshop



Note: In our Cloud9 environment we will be using AWS CLI – which is already installed there. We will want to configure it with a new user which we will create in next steps.

- 9) On the AWS Console, go to **Services** and open the **IAM** service and then select the **Users** page.
- 10) On the **Users** page, select **Add User**.
- 11) You will be taken to the 1st step of the **Add user** wizard. This user will be used for accessing the AWS CLI in your Cloud9 environment. Provide a **User name*** for your user e.g. `ide_user`
- 12) Select the **Programmatic access** option and then select **Next: Permissions**

A screenshot of the AWS IAM 'Add user' wizard, Step 1: Set user details. The wizard has four steps indicated by numbered circles at the top right: 1 (active), 2, 3, and 4. The main heading is 'Add user'. Below it, the section 'Set user details' is highlighted. A sub-heading reads: 'You can add multiple users at once with the same access type and permissions. [Learn more](#)'. There is a text input field for 'User name*' with the value 'ide_user'. Below the input field is a link 'Add another user'. The next section is 'Select AWS access type', with a sub-heading: 'Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)'. Under 'Access type*', there are two options: 'Programmatic access' (selected with a blue checkmark) and 'AWS Management Console access' (unselected). The 'Programmatic access' option has a description: 'Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.' The 'AWS Management Console access' option has a description: 'Enables a **password** that allows users to sign-in to the AWS Management Console.' At the bottom left, there is a note '* Required'. At the bottom right, there are two buttons: 'Cancel' and 'Next: Permissions'.

- 13) On the 2nd step of the wizard select **Attach existing policies directly** and choose the **AdministratorAccess** policy. Then select **Next: Review**

Add user

Set permissions for ide_user

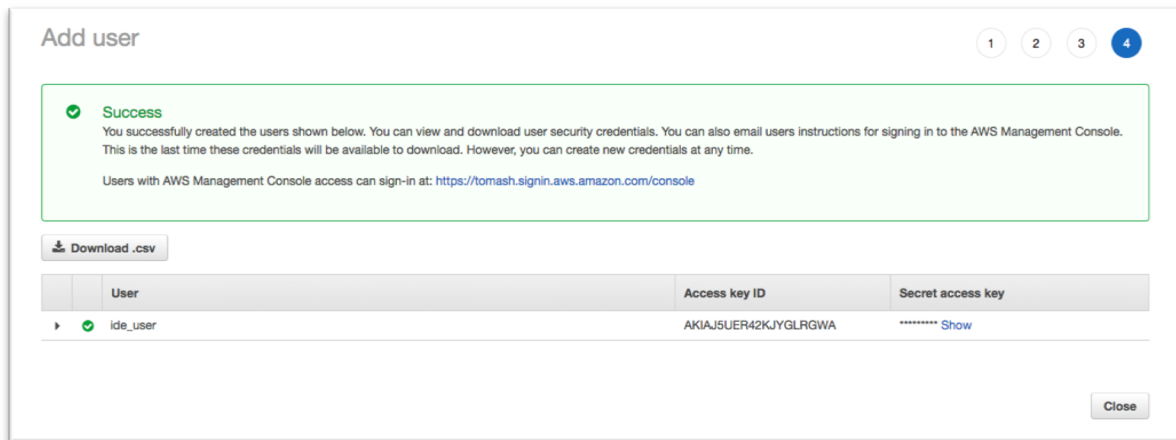
Attach one or more existing policies directly to the users or create a new policy. [Learn more](#)

Filter: Policy type Search Showing 379 results

	Policy name	Type	Attachments	Description
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	8	Provides full access to AWS services and resources.
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	0	Provide device setup access to AlexaForBusiness services
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	0	Grants full access to AlexaForBusiness resources and access to related AWS Services
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	0	Provide gateway execution access to AlexaForBusiness services
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	AWS managed	0	Provide read only access to AlexaForBusiness services
<input type="checkbox"/>	AllowStorageGatewayAssumeBucketAcc...	Customer managed	1	Allow storage gateway to access: dgsdfgsfg
<input type="checkbox"/>	AllowStorageGatewayAssumeBucketAcc...	Customer managed	1	Allow storage gateway to access: dasfadsfdasfdasfadfad
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	AWS managed	0	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Managem...
<input type="checkbox"/>	AmazonAPIGatewayInvokeFullAccess	AWS managed	0	Provides full access to invoke APIs in Amazon API Gateway.
<input type="checkbox"/>	AmazonAPIGatewayPushToChurnWatchl	AWS managed	0	Allows API Gateway to push logs to user's account.

- 14) On the 3rd step of the wizard, select **Create user** and confirm the creation of the user.

- 15) On the 4th step you will need to download the credentials for your newly created user. This is done using the **Download .csv** button

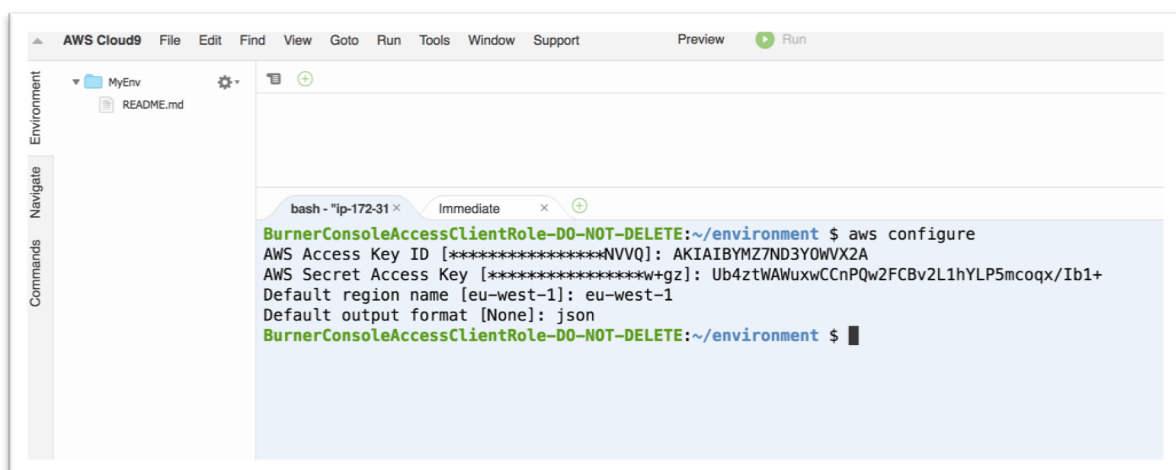


- 16) Now let's return to Cloud9 environment.

- 17) In terminal window of Cloud9 write the command **aws configure**

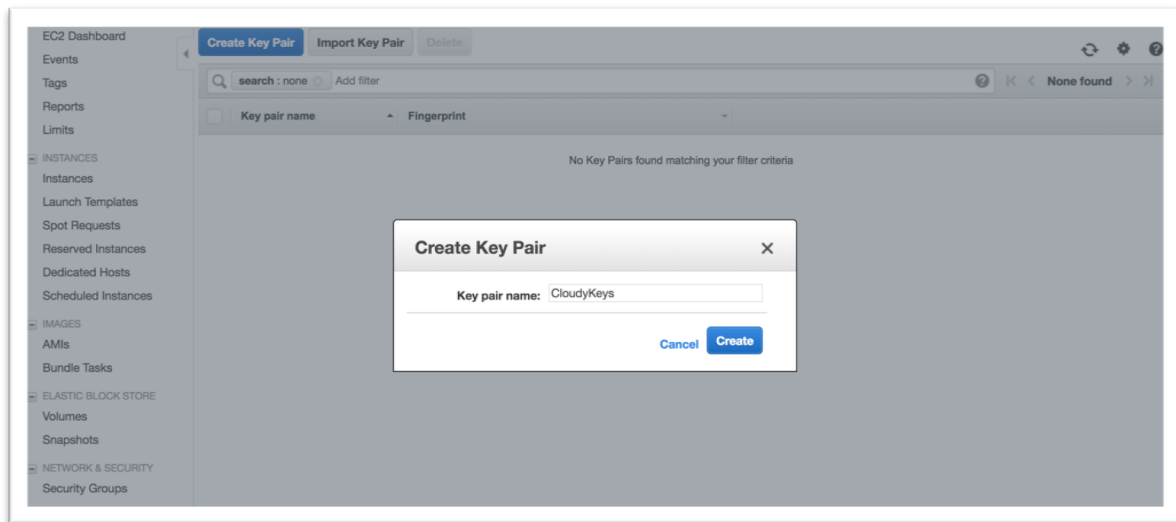
- 18) You will need to provide 4 values:

1. & 2. Are the **Access Key ID** and **Secret Access Key**. These are found in the **Download .csv** file.
3. Is the name of your default region: Therefore, in our case this is **eu-west-1** for EU Ireland
4. Is the default output format. We will use: **json**



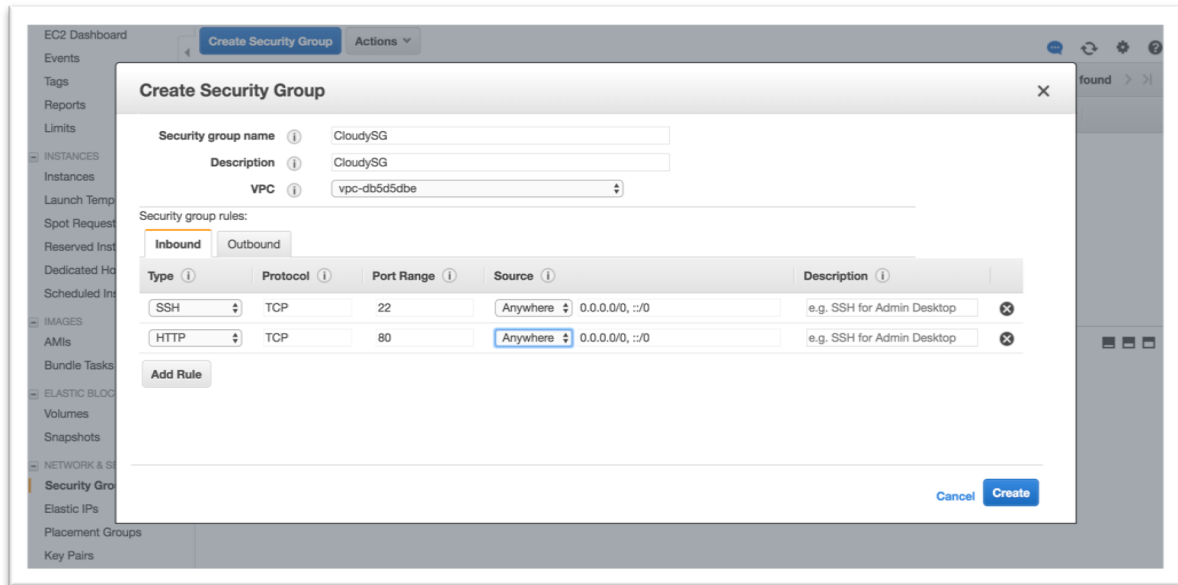
- 19) If you see a popup window informing about updating credentials: choose the **Cancel** option on the first window, and then choose **Permanently disable** on the second window.

- 20) Before we create our first cloud server instance (EC2 instance), we will first create two other resources in the AWS Console. Leave Cloud9 open, and in another browser tab return to the AWS Console and open the **EC2** Service.
- 21) On the left side menu of EC2 Services, find and select **Key Pairs**.
- 22) Select the **Create Key Pair** button.
- 23) Provide a **key pair name**: (we suggest *CloudyKeys*) and then select the **Create** button.



- 24) After selecting the **Create** button, You should see that a .pem file is downloaded to your computer. This file will be named *CloudyKeys.pem* or whatever you entered for the **key pair name** in the last step.
- 25) Now we will create Security Group. Security Groups behave like a native firewall on AWS. Select the **Security Groups** item on the left side menu of EC2 Services window.
- 26) Select the **Create Security Group** button.
- 27) Provide a **Security group name** (we suggest *CloudySG*) and then provide a similar **Description**.
- 28) In the **Inbound** roles tab create two rules for SSH and HTTP traffic. Do this by using the **Add Rule** button, selecting the appropriate **Type**, and setting the **Source** for both rules as **Anywhere**.

Cloud Builders' Day Cloud Technology Workshop



*Note: In a working environment it is not good practice to generally open an SSH port to the whole world using the **Anywhere** source. It is done in this workshop for simplicity.*

29) Let's return to **Cloud9** environment.

30) In the **Cloud9** terminal window, execute following command:

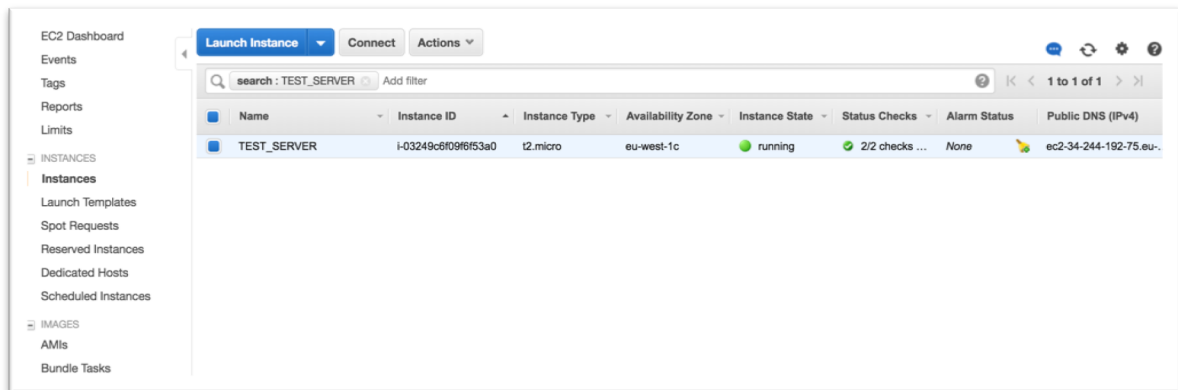
```
aws ec2 run-instances \
  --image-id ami-3bfab942 \
  --instance-type t2.micro \
  --key-name CloudyKeys \
  --security-groups CloudySG \
  --tag-specifications \
  "ResourceType=instance,Tags=[ {Key=Name,Value=TEST_SERVER} ] "
```

*Note: The parameter **image-id** defines AMI image of the server. In our example this will be a Linux server. The **instance-type** parameter defines size of the server – a **t2.micro** [instance type](#) is a server with 1 vCPU and 1 GiB of memory. The last parameter **tag-specifications** defines additional metadata that will be assigned to this server. You can read more about*

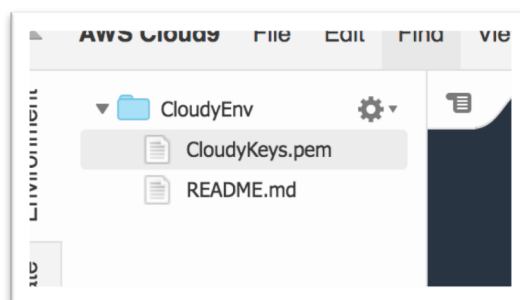
available command line interface options here:

<https://docs.aws.amazon.com/cli/latest/reference/ec2/run-instances.html>

- 31) Return to EC2 Services window. Select the **Instances** option on the left side menu and you should see your new server being created.



- 32) Now you will try to login to the server that you have created. Let's return to **Cloud9 IDE** environment.
- 33) Copy your **.pem** file that you created earlier to your Cloud9 IDE environment. You can do this via drag-and-drop to the **CloudyEnv** folder on the directory pane on the left hand side of your environment.



- 34) Execute `chmod 400 CloudyKeys.pem` in the terminal window.
- 35) Then execute `ssh -i "CloudyKeys.pem" ec2-user@SERVER_IP` – replacing SERVER_IP with the IP of the server that you have created.

36) If everything is correct, then you should see that you have successfully logged into the server on the terminal window.



```
bash - "ec2-user" x (+)
ec2-user:~/environment $ ssh -i "CloudyKeys.pem" ec2-user@172.31.17.38
Last login: Sat Apr 28 10:03:43 2018 from 172.31.27.17

  _|  _|_ )
 _| (  _| /  Amazon Linux AMI
 _|\_|_|_|

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
11 package(s) needed for security, out of 20 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2018.03 is available.
[ec2-user@ip-172-31-17-38 ~]$
```

Summary

In this lab you managed to create and configure your cloud environment using AWS Cloud9. You also used AWS Command Line Interface (CLI) to quickly provision new virtual server. Finally, you successfully logged into your new virtual server. Congratulations!

Lab 2 – Autoscaling and IaaS

One of the most important benefits of cloud is the ability to perform autoscaling, which means that we can adjust the number of resources to our requirements in automatic fashion. In this lab, we will deploy a web application that we will then configure so that it will scale automatically based on our needs. Our application will be deployed according to an Infrastructure as a Code (IaaS) practice, using the AWS CloudFormation service.

Requirements:

- Deploy web application using an IaaS approach.
- Modify your application so that it will scale automatically based on requirements.

Services that you will be using in this lab:

- Amazon Virtual Private Cloud (VPC) - Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can find more information here: <https://aws.amazon.com/vpc/>
- AWS CloudFormation - AWS CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment. CloudFormation allows you to use a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts. This file serves as the single source of truth for your cloud environment. You can find more information here: <https://aws.amazon.com/cloudformation/>
- AWS Elastic Load Balancer - Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security

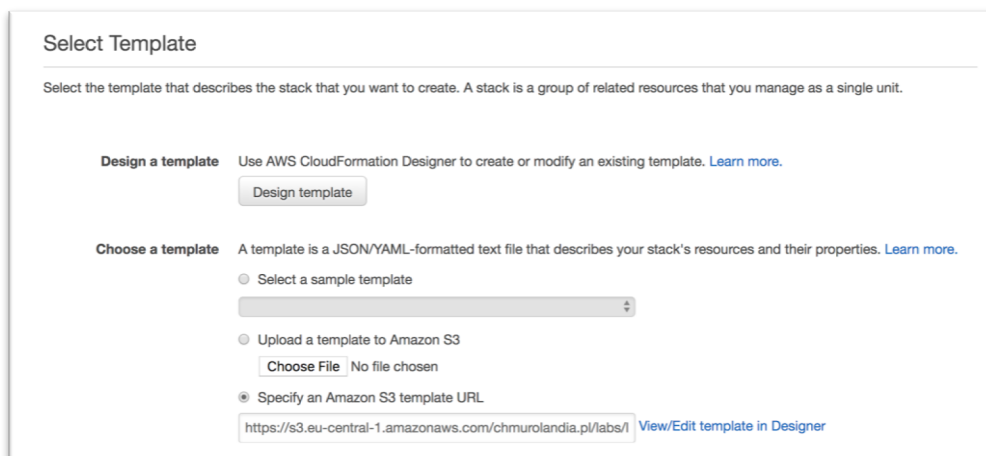
necessary to make your applications fault tolerant. You can find more information here:

<https://aws.amazon.com/elasticloadbalancing/>

Steps:

- 1) Make sure you are in the **Ireland** region.
- 2) Go to the **AWS CloudFormation** service and select the **Create Stack** button.
- 3) Select **Specify an Amazon S3 template URL** and input the following url:

<https://s3-eu-west-1.amazonaws.com/cloudbuildersday/lab-autoscaling/WebApplication.json>



Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

Choose a template A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

☐ Select a sample template

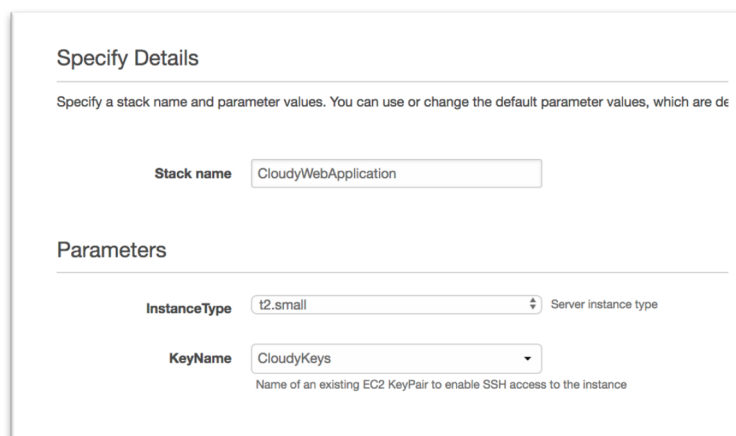
☐ Upload a template to Amazon S3

☒ Specify an Amazon S3 template URL

No file chosen

[View/Edit template in Designer](#)

- 4) Select the **Next** button, and in the next screen (**Specify Details**) of wizard, input a **Stack name** e.g. **CloudyWebApplication**. Then from the **KeyName** menu select your key pair e.g. **CloudyKeys**



Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are de

Stack name

Parameters

InstanceType Server instance type

KeyName Name of an existing EC2 KeyPair to enable SSH access to the instance

- 5) Select the **Next** button on the (**Specify Details**) page

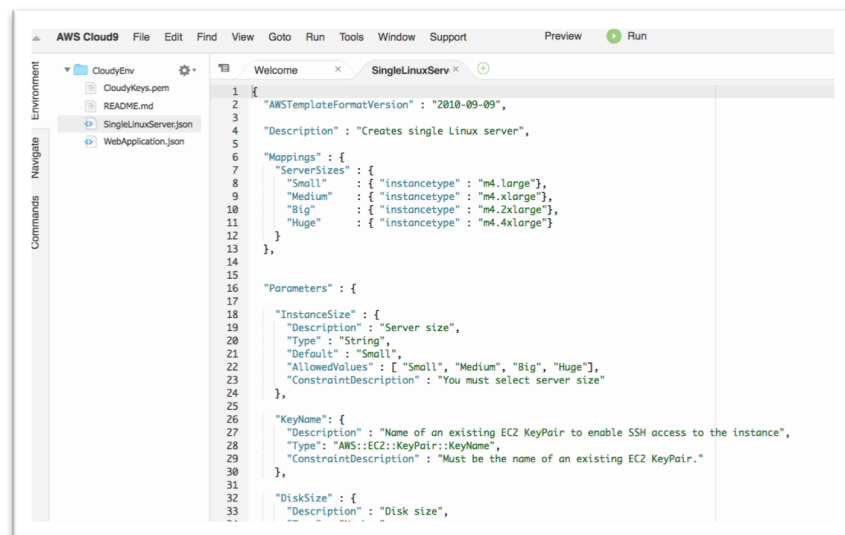
- 6) Select the **Next** button on **Options** screen, moving to the last step of the wizard, which is the **Review** screen.
- 7) Select the **Create** button for the stack. It will take around 10-15 minutes to create all the resources.
- 8) While the stack is being created, we will evaluate the template which is being used for this lab. Return to the **Cloud9** service.
- 9) Execute following command in the Cloud9 terminal window, which will copy template that was used in CloudFormation to create the stack:

```
aws s3 cp s3://cloudbuildersday/lab-autoscaling/WebApplication.json  
WebApplication.json
```

- 10) Let's also check another and easier template. Execute the following command. This will copy the template to your Cloud9 environment:

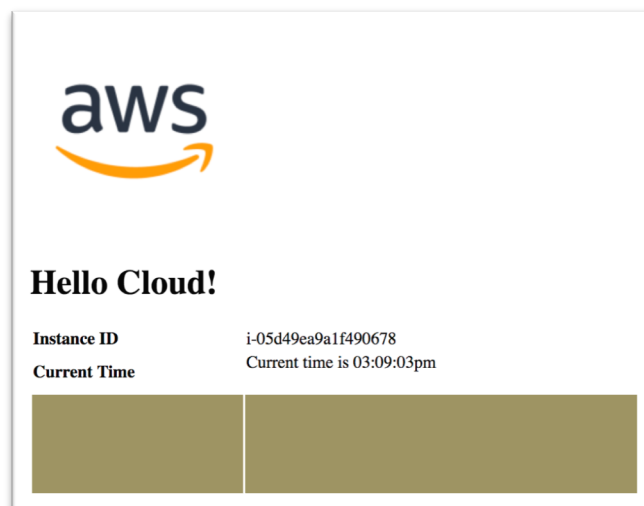
```
aws s3 cp s3://cloudbuildersday/lab-autoscaling/SingleLinuxServer.json  
SingleLinuxServer.json
```

- 11) **Open** templates in Cloud9 and analyze the files.



- 12) Back in the **AWS CloudFormation** service, your stack should be ready with a **CREATE_COMPLETE** status. Select the stack, and in the **Output** tab you should see one **LoadBalancerDNSName** parameter. For the parameter, copy **Value** and try to open it within a browser.

- 13) You should see page similar to those shown below. Try to refresh the page, you should see that the color of the bar will change.



- 14) In the **EC2** service, try to locate the autoscaling group that was created by CloudFormation. Then modify its parameters, so it can scale automatically based on CPU usage.
- 15) Select Scaling Policies tab and click **Add policy**. Name your policy "**CPU scale-out**", type "**70**" as a Target value and click **Create**

Summary:

Congratulations! You have managed to create a web application in few simple steps, you have also modified it so that it can scale automatically based on need. You have also familiarized yourself with the Iaas approach for deploying and building systems.

Lab 3 – Developer Tools. CD in practice

Your management team has decided to increase productivity of the IT department. To do this, they want to introduce a Continuous Delivery (CD) process. A CD process should improve the work of developers and allow the introduction of new features much faster than before. To accomplish this task, you need to provision a couple of different services from the AWS portfolio that will enable you to create CD process at your organization.

Requirements:

- We will soon be working on new project for our new webpage. The CD process which you will build, will be the base for this new project.
- We want you to use a git repository. This will be used for storing our source files.

Services which you will be using in this lab:

- **AWS CodePipeline** – CI/CD service: <https://aws.amazon.com/codepipeline>
- **AWS CodeCommit** – private GIT repository. More about: <https://aws.amazon.com/codecommit>
- **AWS EC2** – public cloud virtual server. More about: <https://aws.amazon.com/ec2/>
- **AWS Command Line Interface (CLI)** – command line AWS communication interface. More about: <https://aws.amazon.com/cli/>
-

Steps:

- 1) Make sure you are in **Ireland** region.
- 2) Log into AWS console and open **CodeCommit** service, in the console click the **Create repository**. Name your new repository and confirm creation.
- 3) On the right side you should see the **Clone URL** button – select it and click **Clone HTTPS** next.
- 4) In Cloud 9 IDE type the below command. Use the copied data with your repository URL to replace the yellow part of the command.

```
git clone https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/tomek-repo
```

- 5) You'll be prompted to provide access keys. Open **IAM** console, click **Users** from the menu and select your user.
- 6) Select **Security credentials** tab and go to the **HTTPS Git credentials for AWS CodeCommit** section.
- 7) Click the **Generate** button. This should open a new window with your access keys displayed.

- 8) Provide these keys in Cloud9. Your empty repository should be copied to the Cloud9 environment.
- 9) Browse to the folder.

```
cd FOLDER_NAME
```

- 10) Type the below command:

```
git config credential.helper 'cache --timeout 900'
```

- 11) In AWS console, open **CodeDeploy** service and click **Create Application**.
- 12) Provide the name for your application (np. WWW) and select **EC2/On-premises** as Compute platform.
- 13) Click **Create deployment group**
- 14) Name your server group (e.g. "servers")
- 15) Select available IAM role.
- 16) In the **Environment configuration** section, select your autoscaling group.
- 17) In the **Load balancer** section select your existing load balancer.
- 18) Type below commands:

```
aws s3 cp s3://cloudbuildersday/lab-deploy/Application.zip .  
unzip Application.zip  
rm Application.zip  
rm -r MACOSX/
```

- 19) Type below commands:

```
git add -A  
git commit -m "First Commit"  
git push
```

- 20) When prompted provide your GIT access keys.
- 21) Make sure you see new files in your folder.
- 22) Open **CodePipeline** service in the AWS console.
- 23) Click the **Create Pipeline**.
- 24) Name your pipeline *CloudCICD* and confirm to the next window.
- 25) Select your CodeCommit repository in the second step.
- 26) In the third step click **Skip Build Stage**
- 27) In the forth step select your CodeDeploy app (www).
- 28) Confirm the selection and create the pipeline.
- 29) Observe how your application updates itself with CI/CD process.
- 30) Bonus task: Add manual approval step in the pipeline.

Summary

Congratulation! In this lab you managed to use the AWS Code family services to easily integrate your application with Continuous Integration and Continuous deployment pipeline with Code Pipeline. You also used a cloud GIT repository (Code Commit) where you stored your code. Finally, you deployed your application artifact with Code Deploy.

Lab 4 – Containers in cloud

In this lab, your management team have given you another task and this time it is related to containers. You will create your own private docker repository where you will push a container image. Then you will deploy it to production, without provisioning any servers, by using AWS Fargate. Good luck!

Requirements:

- Create your own private docker repository
- Deploy your docker containers without provisioning servers.

Services which you will be using in this lab:

- Amazon Elastic Container Registry (ECR) - Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon Elastic Container Service (ECS), simplifying your development to production workflow. Amazon ECR eliminates the need to operate your own container repositories or worry about scaling the underlying infrastructure. Amazon ECR hosts your images in a highly available and scalable architecture, allowing you to reliably deploy containers for your applications. You can find more information here: <https://aws.amazon.com/ecr/>
- AWS Fargate - AWS Fargate is a technology for Amazon ECS and EKS* that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing. AWS Fargate removes the need for you to interact with or think about servers or clusters. Fargate lets you focus on designing and building your applications instead of managing the infrastructure that runs them. You can find more information here: <https://aws.amazon.com/fargate/>

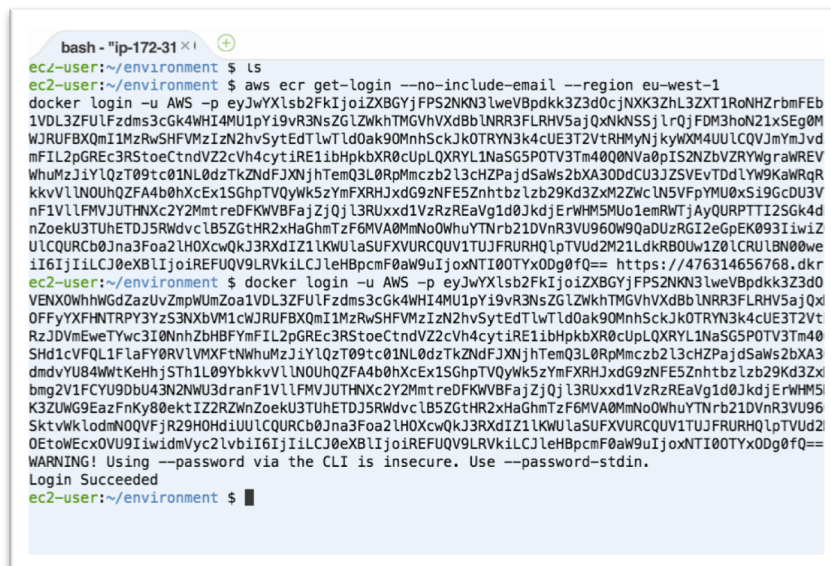
Steps:

- 1) Make sure that you are in the **Ireland** region.

- 2) Open the **Elastic Container Service** service, and then on the **Repositories** tab select the **Get Started** button.
- 3) Provide name for your repository e.g. **cloudy-repo**
- 4) Your repository will be created
- 5) Select your repository and click **View push commands**
- 6) Return to your Cloud9 environment.
- 7) Execute the first command that was provided by the ECR service.

```
aws ecr get-login --no-include-email --region eu-west-1
```

- 8) You will see a command, starting from the **docker login ...** - you should copy and paste this command again in Cloud9 terminal. Complete the copy and then execute the command.
- 9) If everything went well, you should see a similar output to the one below:



```
bash - "ip-172-31" ×
ec2-user:~/environment $ ls
ec2-user:~/environment $ aws ecr get-login --no-include-email --region eu-west-1
docker login -u AWS -p eyJwYXlsb2FkiIjoiazXBGYjFPS2NKN3lweVBpdDk3Z3d0cXNkX3ZlL3ZXT1RoNHZrbmFEB
1VDL3ZFULFzdm3cGk4WHI4MU1pYi9vR3NsZG1ZwkhTMGVhVXdBbUNRR3FLRHV5ajQxNkN5SjlrQjFDM3hoN21xSEg0M
WJRUF8XQmI1MzRwSHFVMzIzN2hvSyEtLWtld0ak90MnhScKJkOTRYN3k4cUE3T2VtRHMjYjkyWXM4UULCQVJmYmJvd
mF1L2pGREc3RStoeCtndVZ2cVh4cytiRE1ibHpkbXR0cUpLQXRYL1NaSG5POTV3Tm40Q0NVa0pIS2NZbVZRYWgrREV
WhuMzJiYlQzT09tc01NL0dzTkZndFJXNjhTemQ3L0RpMmczb213cHZPajdSaWs2bXA3ODdCU3JZSVEdTddlYw9KawRqR
kkvVl1N0UHQZFA4b0hXcEx1SGhpTVQyWk5ZyMFXRHJxdG9zNFE5Znhtbzlb29Kd3ZmM2ZzcWlNSVFPYMU0xS19GcDU3V
nF1Vl1FMVJUTHNxc2Y2MmtrdFkwVBFajZjQj13RUxxd1VzRzREaVg1d0JkdjErWHM5MU01emRWtjAyQURPTTI2SGk4d
nZoekU3TUhETDJS5RwDvcLB5ZGtHR2xHaGhmTzF6MVA0MmNo0WhuYTNrb21Dvnr3VU960W9QaDUzRGIEGpEK093I1wiZ
ULCQURCb0Jna3Foa2LH0XcwQk3RXdIZ1LKWUlaSUFxVURCQUV1TUJFRURHQlpTVUd2M21LdkRB0Uw1Z0LCRU1BN00we
iI6IjIiLCJ0eXBliIjoireFUQV9LRVkiLCJleHBpcmF0aW9uIjoxNTI0OTYxODg0f0== https://476314656768.dkr
ec2-user:~/environment $ docker login -u AWS -p eyJwYXlsb2FkiIjoiazXBGYjFPS2NKN3lweVBpdDk3Z3d0
VENX0WhhWgdZazUvZmpWUzoa1VDL3ZFULFzdm3cGk4WHI4MU1pYi9vR3NsZG1ZwkhTMGVhVXdBbUNRR3FLRHV5ajQx
QFFyYXFHNTY3YzS3NXbVM1cWJRUF8XQmI1MzRwSHFVMzIzN2hvSyEtLWtld0ak90MnhScKJkOTRYN3k4cUE3T2Vt
RzJ0VnEweTYwczI0NnhZbHBFYmF1L2pGREc3RStoeCtndVZ2cVh4cytiRE1ibHpkbXR0cUpLQXRYL1NaSG5POTV3Tm40
SHD1cvFQl1FlaFY0RVlVhXFTNWhuMzJiYlQzT09tc01NL0dzTkZndFJXNjhTemQ3L0RpMmczb213cHZPajdSaWs2bXA3
dmdvYU84WwtKeHhJSTh1L09YbkKvVl1N0UHQZFA4b0hXcEx1SGhpTVQyWk5ZyMFXRHJxdG9zNFE5Znhtbzlb29Kd3Zx
bmg2V1FCYU9DbU43N2NWU3dranF1Vl1FMVJUTHNxc2Y2MmtrdFkwVBFajZjQj13RUxxd1VzRzREaVg1d0JkdjErWHM5
K3ZUW69EazFnKy80ektIZ2RZwnZoekU3TUhETDJS5RwDvcLB5ZGtHR2xHaGhmTzF6MVA0MmNo0WhuYTNrb21Dvnr3VU96
SktwWkldmNQVFjR29H0HdiUULCQURCb0Jna3Foa2LH0XcwQk3RXdIZ1LKWUlaSUFxVURCQUV1TUJFRURHQlpTVUd2
0EtoWecx0VU9IiwidmVyc2lvbiI6IjIiLCJ0eXBliIjoireFUQV9LRVkiLCJleHBpcmF0aW9uIjoxNTI0OTYxODg0f0==
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
ec2-user:~/environment $
```

- 10) Execute following command, which will copy sample docker project to your Cloud9 environment.

```
aws s3 cp s3://cloudbuildersday/lab-docker/docker-sample-page.zip docker-sample-page.zip
```

- 11) Unzip the file with following command:

```
unzip docker-sample-page.zip
```

- 12) Analyze the content of the **docker-sample-page** directory.

13) Go to the docker-sample-page directory using following command:

```
cd docker-sample-page
```

14) Now, we need to build our docker image. To do this, execute command number [2] from ECR.

```
docker build -t cloudy-repo .
```

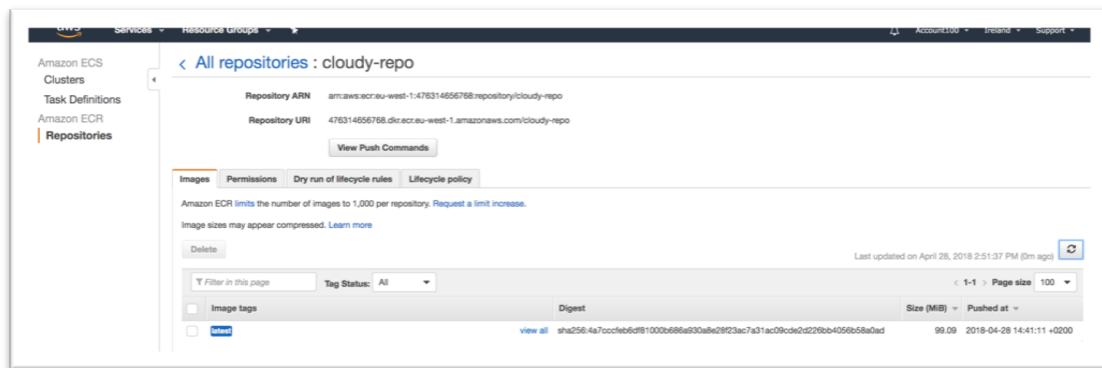
15) After the build completes, tag your image so that you can push the image to this repository.

Execute command number [3] from ECR.

16) Run command number [4] to push this image to your newly created AWS repository:

17) Note and record the third parameter of the previous command [number [4]].

18) Now, back in ECR repository, you should see that your docker image was successfully pushed to the registry and is available for use.



19) Congratulations! The first step of the lab is now done. Next we will need to create a task definition. To create a task definition, select **Task Definition** from the left menu and then select the **Create new Task Definition** button.

20) In the first step of this Task Definition wizard for **Step 1: Select launch type compatibility** chose the **Fargate** option and select the **Next step** button

Cloud Builders' Day
Cloud Technology Workshop

Select launch type compatibility

Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

FARGATE

Price based on task size
Requires network mode awsvpc
AWS-managed infrastructure, no Amazon EC2 instances to manage

EC2

Price based on resource usage
Multiple network modes available
Self-managed infrastructure using Amazon EC2 instances

*Required

Cancel Next step

21) Provide a **Task Definition Name** e.g.: *MyDocker*

22) Scroll down and change the **Task memory (GB)** and **Task CPU (vCPU)** values to the lowest possible values.

Task size ?

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB) 0.5GB

The valid memory range for 0.25 vCPU is: 0.5GB - 2GB.

Task CPU (vCPU) 0.25 vCPU

The valid CPU for 0.5 GB memory is: 0.25 vCPU

23) In **Container Definition** section select the **Add container** button.

24) In new window complete the wizard with the following information:

- i. **Container name:** Apache
- ii. **Image:** Image ID/link which you have copied before.
- iii. **Port mappings:** 80

25) Select the **Add** button, to add a container and close the window. Finally select the **Create** button to create a Task Definition.

26) Congratulations! Your Task Definition is ready. Now we need to create a cluster.

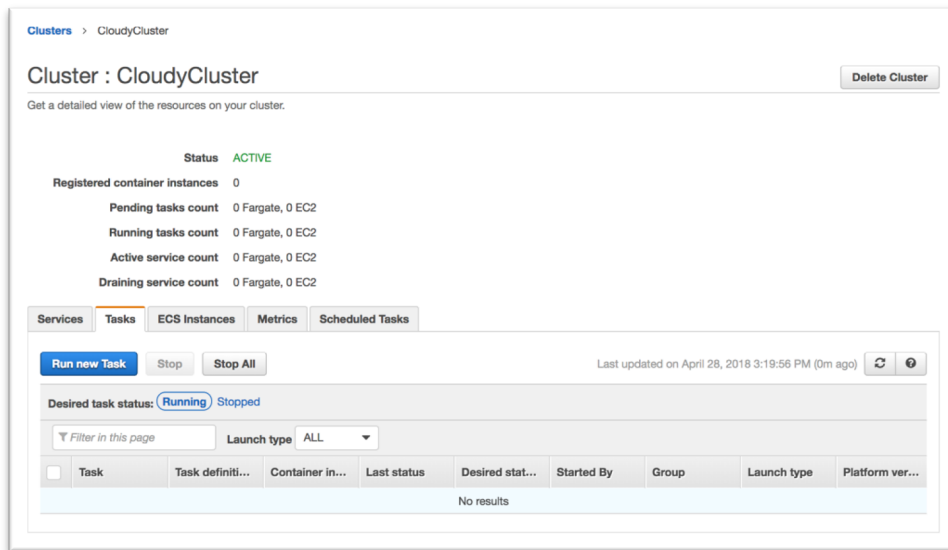
27) On the ECS services window select the **Clusters** tab on the left side, and then select the **Create Cluster** button.

28) Choose the **Networking only** option, select the **Next Step** button.

29) Complete the required **Cluster name** field e.g. with the value *CloudyCluster*, and finally select the **Create** button.

30) Your cluster is ready. Now the last step is to deploy your container.

31) Select your cluster, in the view of your cluster select the **Tasks** tab and then the **Run new Task** button.



32) In the new wizard window, select configurations as shown in the screen below. You should choose your default VPC network and the first available subnet.

Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To Options.

Launch type ☒ FARGATE ☐ EC2 ⓘ

Task Definition MyDocker:1 ▾

Platform version LATEST ▾ ⓘ

Cluster CloudyCluster ▾

Number of tasks 1

Task Group ⓘ

33) It will take about 1-2 minutes for your container to be deployed.

34) Now, your container should display the status **RUNNING**. Select the Task.

35) On the next page copy the Public IP address and then use this IP address as a URL by trying to open it in a browser. You should see a web page:

Cloud Builders' Day
Cloud Technology Workshop

Clusters > CloudyCluster > Task: f1f43e2b-7042-4214-8b06-e71cbf3c7f5c

Task : f1f43e2b-7042-4214-8b06-e71cbf3c7f5c

Details Logs

Cluster CloudyCluster

Launch type FARGATE

Platform version 1.1.0

Task definition MyDocker:1

Group family:MyDocker

Task role None

Last status RUNNING

Desired status RUNNING

Created at 2018-04-28 15:21:44 +0200

Network

Network mode awsvpc

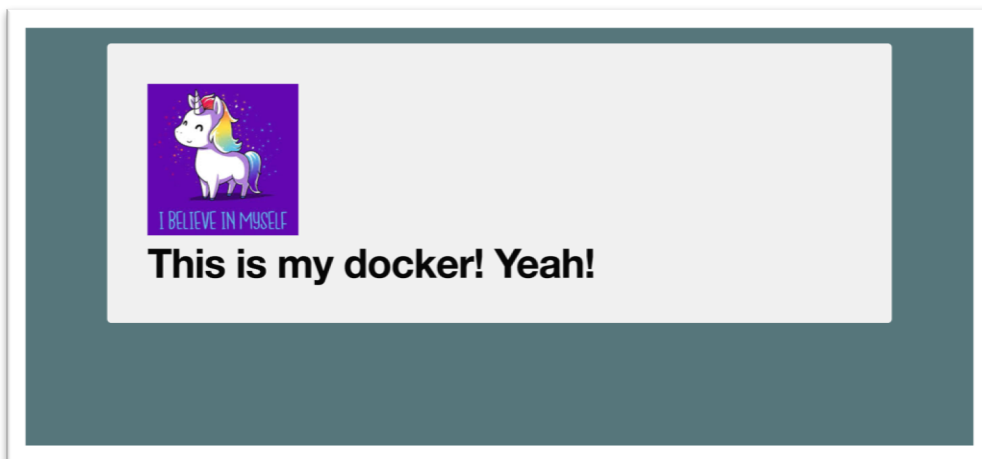
ENI Id eni-a6555d94

Subnet Id subnet-5b09b813

Private IP 172.31.16.62

Public IP 34.241.41.222

Mac address 06:ac:9b:9d:af:ce



Summary

Congratulations! You have managed to create your own private docker repository, where you have deployed your first docker image. Finally, you have managed this deployment without the need to provision any servers.

Lab 5 – Collecting data in real-time

In the next lab you will need to build application which will enable your company to collect data in real-time about what is happening on the webpage. Our application will consist of two parts: backend – which will be able to collect data in real-time and store it on S3, and frontend – which will be simple HTML page, which you will run from your local computer.

Requirements:

- Create your streaming back-end application, which will be able to collect data about events from your web application in real-time. It's need to be simple, fast, and not-expensive solution.
- All collected data should be stored on S3.

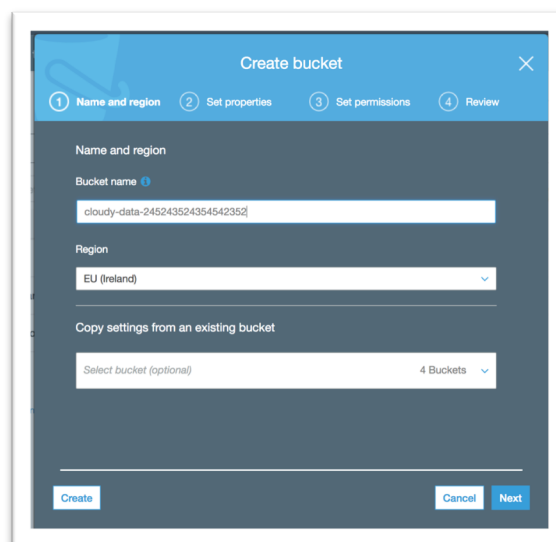
Services which you will be using in this lab:

- Amazon Simple Storage Service (S3) - Amazon S3 is object storage built to store and retrieve any amount of data from anywhere – web sites and mobile apps, corporate applications, and data from IoT sensors or devices. It is designed to deliver 99.999999999% durability, and stores data for millions of applications used by market leaders in every industry. S3 provides comprehensive security and compliance capabilities that meet even the most stringent regulatory requirements. You can find more information here: <https://aws.amazon.com/s3/>
- Amazon Kinesis Data Firehose - Amazon Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security. You can find more here: <https://aws.amazon.com/kinesis/data-firehose/>
- Amazon Cognito - Simple and Secure User Sign-Up, Sign-In, and Access Control Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps

quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0. Amazon Cognito provides solutions to control access to backend resources from your app. You can define roles and map users to different roles so your app can access only the resources that are authorized for each user. You can find more information here: <https://aws.amazon.com/cognito/>

Steps:

- 1) Make sure you are in **Ireland** region.
- 2) In first part we will create S3 bucket where we will store our data. Let's open **S3** service.
- 3) Click on **Create bucket** button and provide name for your bucket. It will need to be unique across all AWS users and regions.



- 4) Click on **Create** button to create it.
- 5) Don't close this window, and in the next browser tab open **AWS Service**.
- 6) The next step will be to create streaming backend application. Open **Kinesis** service.
- 7) We will be creating Kinesis Firehose, so click on **Create delivery stream** button.
- 8) Provide name of your stream, for example: *data_stream* and click on **Next** button.
- 9) On the second step of the wizard don't change anything and click on **Next** button.
- 10) On the third step of the wizard we will stay with S3 as destination of our data, we will also specify our S3 bucket – which we have created before.

Cloud Builders' Day Cloud Technology Workshop

Select destination

Destination* ☒ Amazon S3 ⓘ
☐ Amazon Redshift ⓘ
☐ Amazon Elasticsearch Service ⓘ
☐ Splunk ⓘ

Firehose to S3 data flow overview

Source → Firehose delivery stream → S3 bucket (destination)

Source records → Transformed records

If transformation fails → S3 bucket (optional backup)

----- Optional

S3 destination

S3 bucket* cloudy-data-245243524354542...
[View bucket cloudy-data-245243524354542352 in S3 console](#)

Prefix Specify prefix ⓘ

Required

- 11) On the fourth step, we will modify how often Kinesis will save data on S3. Change **Buffer size** to 1 and **Buffer interval** to 60.

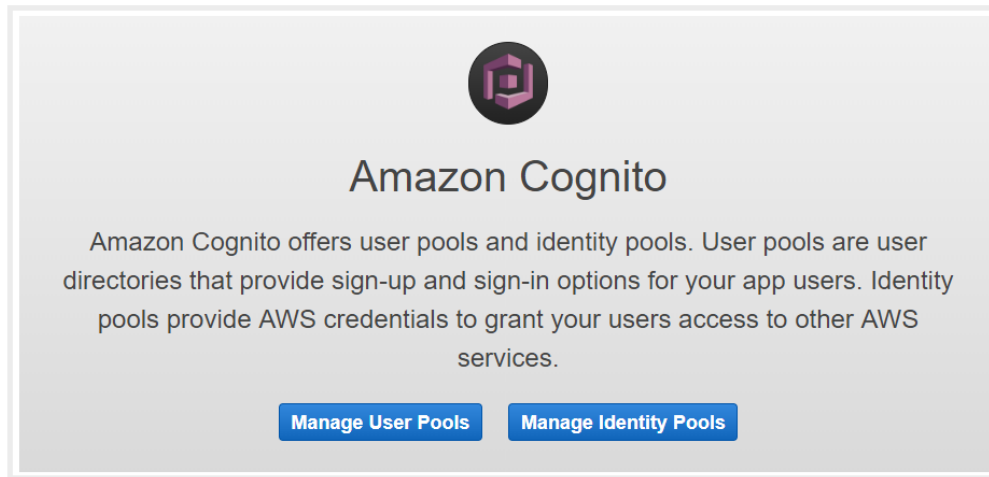
S3 buffer conditions

Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery will be triggered once either of these conditions has been satisfied. [Learn more](#)

Buffer size* 1 MB
Specify a buffer size between 1-128MB

Buffer interval* 60 seconds
Specify a buffer interval between 60-900 seconds

- 12) Still on the fourth step, you will need to create IAM Role, click on **Create new or Choose** button and then on **Allow** button. Move to the last step.
- 13) On the last step review configuration and click on **Create delivery stream**.
- 14) The last part will be Cognito. Open **Cognito** service.
- 15) Click on **Manage Identity pools** button.

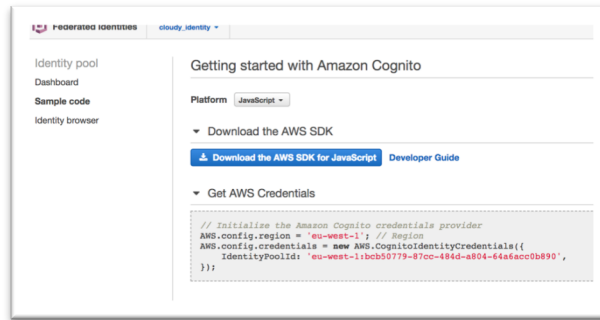


- 16) Provide name for your Identity pool, for example: *cloudy_identity*
- 17) Check **Enable access to unauthenticated identities** checkbox.
- 18) Click on **Create Pool** button.
- 19) On the next step you will see a pop-up with information that your pool requires access to your resources. Click **View Details** We will modify the policy for authenticated and unauthenticated identities. Click on **View Policy Document** button and then click on **Edit** button.
- 20) Provide following IAM policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyForCloudyApp",
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": "*"
    }
  ]
}
```

- 21) Click on **Allow** button.
- 22) On **Getting started with Amazon Cognito** window change **Platform** to **JavaScript**.
- 23) Note somewhere **IdentityPoolId** provided in **Get AWS Credentials** section.

Cloud Builders' Day Cloud Technology Workshop



24) In your IDE. Leave the current folder:

```
cd ..
```

25) Create new folder for your new application and browse to it:

```
mkdir kinesis_app  
cd kinesis_app
```

26) Download following file and unzip it on your IDE:

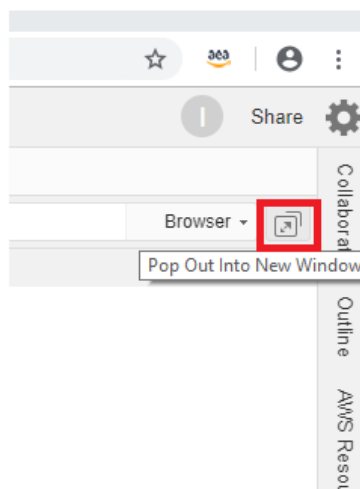
```
wget https://s3-eu-west-1.amazonaws.com/cloudbuildersday/lab-  
realtime/kinesis_app.zip  
unzip kinesis_app.zip
```

27) Open **index.html** file and modify it on line 10. Change **MODIFY_THIS** to **IdentityPoolId** which you have copied before.

28) Open now **index.html** file in a browser in your IDE. **Right click** on **index.html** and select **Preview**.

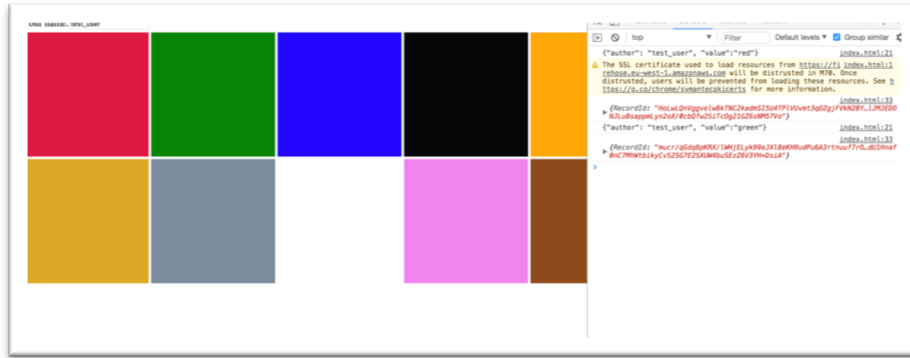
29) Provide your name and start clicking on boxes.

30) Click **Pop Out into new Window** button



31) If you will open JS Console in browser, you should see that communication between web and backend is working.

Cloud Builders' Day Cloud Technology Workshop



32) Now wait a minute and go back to **S3** Console.

33) In your S3 bucket you should see new files which represents click events on webpage.

Summary:

Congratulations! In just a couple of minute, you managed to build system, which is able to collect data in real-time and save it on S3 storage – where you could analyze them!

Lab 6 – Analytics – Hadoop in practice

More and more companies are launching Big Data projects. They often they use Hadoop ecosystems to quickly analyze big collection of data which they have collected. In this lab we have prepared a sample data set 1GB in size, which we will analyze using two methods: both with servers and without them.

Requirements:

- You need to launch a small Hadoop cluster
- Analyze the provided data set with a cluster.
- Analyze the provided data set using a serverless approach.

Services which you will be using in this lab:

- Amazon EMR - Amazon EMR provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2

instances. You can also run other popular distributed frameworks such as Apache Spark, HBase, Presto, and Flink in Amazon EMR, and interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB. You can find more information here: <https://aws.amazon.com/emr/>

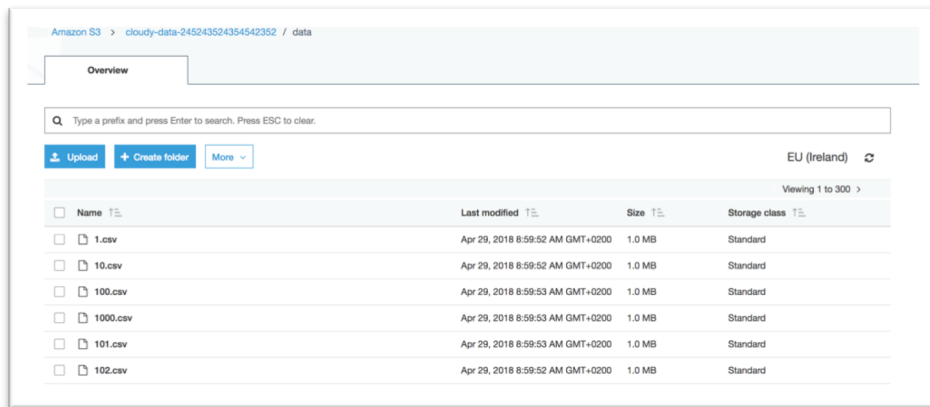
- *Amazon Athena* - Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets. You can find more information here: <https://aws.amazon.com/athena/>

Steps:

- 1) Make sure that you are in the **Ireland** region.
- 2) Open the EMR service where we will create our own Hadoop cluster.
- 3) Select the **Create cluster** button.
- 4) Provide a name of your cluster e.g. CloudyHadoop
- 5) Unbox **Logging**
- 6) In the **Security and access** section choose **EC2 key pair** which you created in the first lab.
- 7) Click on the **Create cluster** button. It will take about 15 minutes to make the cluster.
- 8) In order to not waste time, we will now try to analyze the data in Serverless way. To do this, first you will copy data and put it in your own S3 bucket (the can be one that you have created before, or new one). Then return to the **Cloud9** service.
- 9) Execute the following command in the Cloud9 environment. It will copy the sample data set (1GB) to your own S3 bucket. Correct the command by providing your own S3 bucket name.

```
aws s3 cp s3://cloudbuildersday/lab-bigdata s3://NAZWA_TWOJEGO_BUCKET_S3 --recursive
```

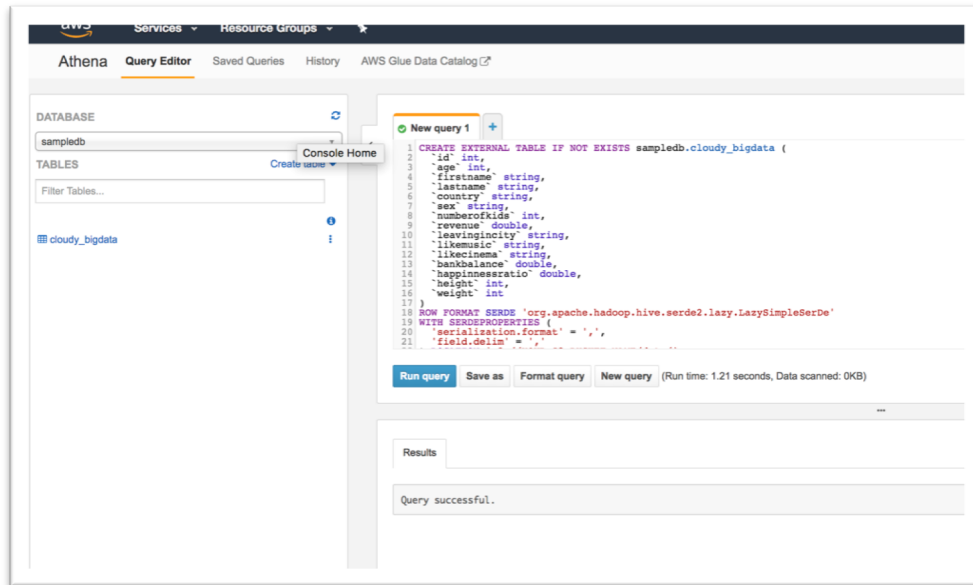
- 10) After couple of seconds, you should see in the S3 bucket a new directory with name **data** containing 1000 files. Try to download one of the files and analyze it's content. You can also modify the file and then re-upload it to S3 bucket.



- 11) Now, let's open the **Athena** service.
- 12) If you see splash page, click **Get Started** and go through wizard by clicking **X**
- 13) Execute following command in the Athena console. Correct the last part of the query to point to your S3 bucket.

```
CREATE EXTERNAL TABLE IF NOT EXISTS sampledb.cloudy_bigdata (  
  `id` int,  
  `age` int,  
  `firstname` string,  
  `lastname` string,  
  `country` string,  
  `sex` string,  
  `numberofkids` int,  
  `revenue` double,  
  `leavingincity` string,  
  `likemusic` string,  
  `likecinema` string,  
  `bankbalance` double,  
  `happinnessratio` double,  
  `height` int,  
  `weight` int  
)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'  
WITH SERDEPROPERTIES (  
  'serialization.format' = ',',  
  'field.delim' = ','  
) LOCATION 's3://YOUR_S3_BUCKET_NAME/data/'
```

Cloud Builders' Day Cloud Technology Workshop



14) Next let's see if we can perform some sample queries on the data that we are storing on S3.

Execute following commands:

```
SELECT * FROM "sampledb"."cloudy_bigdata" limit 10;
```

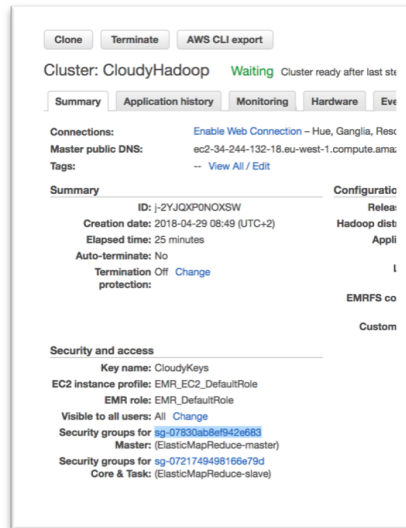
```
SELECT count(*) FROM "sampledb"."cloudy_bigdata";
```

```
SELECT country, count(*) FROM "sampledb"."cloudy_bigdata" group by country order by 2 desc;
```

15) Return to the **EMR** service. Its status should be **Waiting**.

16) Before we will be able to login to our cluster, we will need to modify the security group of the master node and open an SSH port. Select **Security groups for Master**.

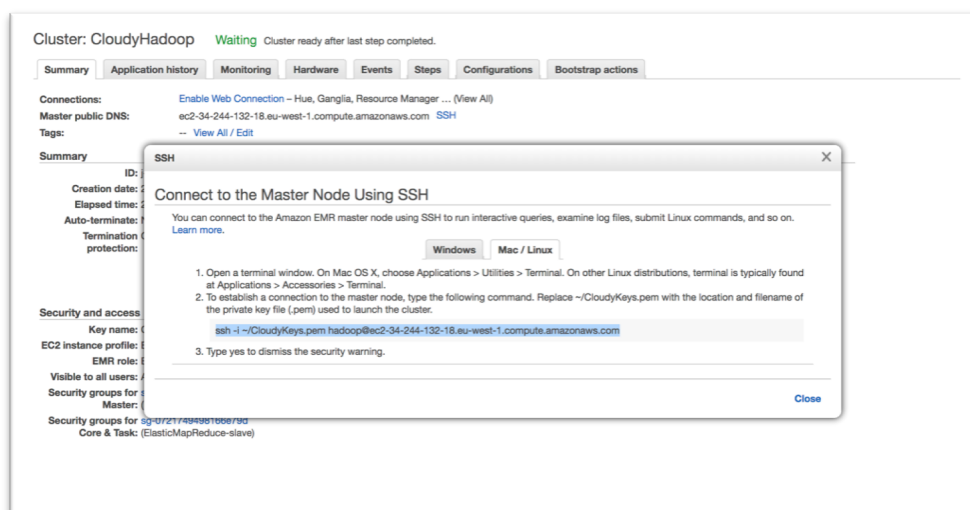
Cloud Builders' Day Cloud Technology Workshop



- 17) In the new window modify (**Edit**) the security group of your **Master** node by adding a new rule that opens an **SSH** connection from Anywhere.

Note: It is not best practice to open SSH to any source, we do this only for simplicity in the lab.

- 18) Next go back to the EMR console and select the SSH button. A new popup window will appear, from this window copy the command provided.



19) The next step will be to paste and execute the command in the Cloud9 environment.

However, modify the command slightly by removing tilde and slash from beginning.

```
ssh -i ./CloudyKeys.pem hadoop@ HADOOP_IP
```

```
ssh - "hadoop@i x i" (+)
ec2-user:~/environment $ clear
ec2-user:~/environment $ ssh -i CloudKey.pem hadoop@ec2-34-244-132-18.eu-west-1.compute.amazonaws.com
The authenticity of host 'ec2-34-244-132-18.eu-west-1.compute.amazonaws.com (172.31.22.16)' can't be established.
ECDSA key fingerprint is SHA256:3i08pgPpdDbrY8KNKGiHGK+ZYwQh9VqT2EKR2oZcYqI.
ECDSA key fingerprint is MD5:a3:18:1b:f3:75:80:a:6c:7e:e9:d3:4c:11:55:b1:76.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-244-132-18.eu-west-1.compute.amazonaws.com,172.31.22.16' (ECDSA) to the list of known hosts.
Last login: Sun Apr 29 07:09:19 2018

  _ | _ | _ |
  _ | ( _ /
 _ | \ _ | _ |
Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
15 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2018.03 is available.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R:::::::::R
EE::::::::EEEEEEEE::::E M::::::::M M::::::::M R:::::RRRRRR::::R
E:::E EEEEE M::::::::M M::::::::M RR::::R R::::R
E:::E M::::::::M M:::M::::M R:::R R::::R
E:::EEEEEEEEEE M:::M M:::M M:::M R:::RRRRRR::::R
E::::::::::::E M:::M M:::M M:::M R:::::::::RR
E:::EEEEEEEEEE M:::M M:::M M:::M R:::RRRRRR::::R
E:::E M:::M M:::M M:::M R:::R R::::R
E:::E EEEEE M:::M MMM M:::M R:::R R::::R
EE::::::::EEEEEEEE::E M:::M M:::M R:::R R::::R
E::::::::::::E M:::M M:::M RR::::R R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

[hadoop@in-172-31-22-16 ~]$
```

20) Congratulations! You are now on the master node of your Hadoop cluster! The first step will

be to copy data from S3 to local HDFS storage. Execute following command:

```
hadoop fs -cp s3://YOUR S3 BUCKET NAME/data /data
```

21) It will take a couple of minutes to copy data.

22) Now, we need to open a Hive shell. Execute following command in the Cloud9 terminal:

hive

23) Next, we will execute the hive command, which will create a table based on data that we saved on HDFS.

Cloud Builders' Day
Cloud Technology Workshop

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudy_data (
    `id` int,
    `age` int,
    `firstname` string,
    `lastname` string,
    `country` string,
    `sex` string,
    `numberofkids` int,
    `revenue` double,
    `leavingincity` string,
    `likemusic` string,
    `likecinema` string,
    `bankbalance` double,
    `happinessratio` double,
    `height` int,
    `weight` int
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
    'serialization.format' = ',',
    'field.delim' = ','
) LOCATION '/data';
```

24) Next let's execute some sample SQL queries.

```
SELECT country, count(*) from cloudy_data group by country;
```

25) After couple of seconds you will see results of your SQL query.

```

mapreduce country, country FROM country_data_group BY country;
query ID = hadoop_20180429073502_f188c05d-5b52-4af6-bda5-ef42ba7f38d6
total jobs = 1
Launching Job 1 out of 1
status: Running (Executing on YARN cluster with App id application_1524984691510_0001)


```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
lap 1	container	SUCCEEDED	22	22	0	0	0	0	0
reducer 2	container	SUCCEEDED	5	5	0	0	0	0	0

```

VERTICES: 02/02  [=====>>>] 100% ELAPSED TIME: 32.78 s

OK
anada 737170
rance 737475
ernany 1472966
olland 737922
oland 4419495
pain 738001
SA 736521
hina 736921
apan 737649
ustria 736585
exico 736561
ussia 735457
urkey 737277
Time taken: 36.848 seconds. Fetched: 13 row(s)

```

Summary

Congratulations! You have managed to launch your own Hadoop cluster where you analyzed 1GB of data. You also managed to analyze the data using Athena, which is a serverless approach.

Lab 7 – Business Intelligence in Cloud

Your management team have given you another important task. This time you are responsible for launching a new BI (Business Intelligence) project within your organization. You only have 10 minutes for this... It sounds improbable, but not with Cloud!

Requirements:

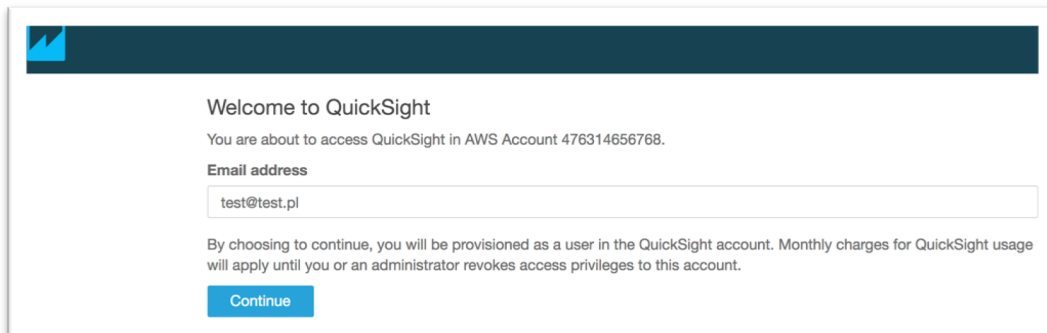
- Use the Amazon QuickSight service to build a BI solution.
- Use QuickSight to build some visualizations of data that you have previously made available through the Athena service.

Services which you will be using in this lab:

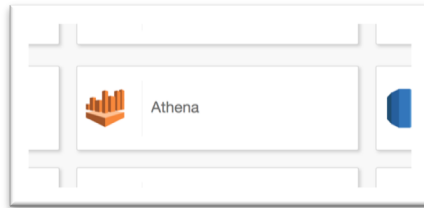
- *Amazon QuickSight* - Amazon QuickSight is a fast, cloud-powered business analytics service that makes it easy to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Using our cloud-based service you can easily connect to your data, perform advanced analysis, and create stunning visualizations and rich dashboards that can be accessed from any browser or mobile device. You can find more information here: <https://aws.amazon.com/quicksight/>

Steps:

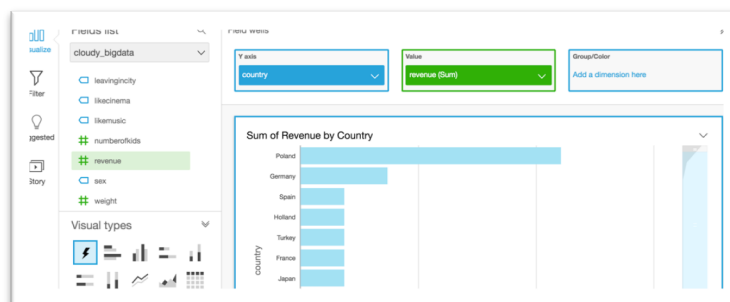
- 1) Make sure you are in the **Ireland** region.
- 2) Open the **QuickSight** service.
- 3) If you will see a welcome window asking for your email address, fill out the form and select the **Continue** button.



- 4) Select the **Manage data** button and then select **New data set**.
- 5) From list of available data sources choose Athena.



- 6) Provide a name for your data source e.g. *CloudyData*, and then select the **Create data source** button.
- 7) In the next window, we will choose our database: **sampledb** and then our table: **cloudy_bigdata**.
- 8) In the next window select **Directly query your data** option and then select the **Visualize** button.
- 9) Congratulations! You have now you can created your own dashboards! We suggest you play around with the tool to create some interesting diagrams.



Summary

Congratulations! In just 5 minutes you managed to create your own BI solution that was based on data which you have previously saved.

Lab 8 – Building serverless app with AI capabilities

Serverless and AI are popular terms that often appear in media and technology conversations. In this lab we will build a fully working application in the cloud with those two technologies. Applications should detect new text files that appear in a S3 bucket, then it should detect what language is being used in the text and convert it to an audio file (mp3). The newly

created audio file will then be saved in an S3 bucket along with the original text file. In addition, we should log the conversion in a NoSQL database. The last part of application will be a new RESTful webservice that expects to receive the user name from the file retuning an audio file in the browser.

Services that will be used in this lab:

- *AWS Lambda* - AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app. You can find more information here: <https://aws.amazon.com/lambda/>
- *Amazon DynamoDB* - Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model, reliable performance, and automatic scaling of throughput capacity make it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications. You can find more information here: <https://aws.amazon.com/dynamodb/>
- *Amazon Polly* - Amazon Polly is a service that turns text into lifelike speech, allowing you to create applications that talk, and build entirely new categories of speech-enabled products. Amazon Polly is a Text-to-Speech service that uses advanced deep learning technologies to synthesize speech that sounds like a human voice. With dozens of lifelike voices across a variety of languages, you can select the ideal voice and build speech-enabled applications that work in many different countries. You can find more information here: <https://aws.amazon.com/polly/>
- *Amazon APIGateway* - Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. With a few clicks in the AWS Management Console, you can create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS

Lambda, or any web application. You can find more information here:

<https://aws.amazon.com/api-gateway>

- *Amazon Comprehend* - Amazon Comprehend is a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. Amazon Comprehend identifies the language of the text; extracts key phrases, places, people, brands, or events; understands how positive or negative the text is; and automatically organizes a collection of text files by topic. You can find more information here: <https://aws.amazon.com/comprehend/>
- *Amazon Simple Notification Service (SNS)* - Amazon Simple Notification Service (SNS) is a flexible, fully managed pub/sub messaging and mobile notifications service for coordinating the delivery of messages to subscribing endpoints and clients. With SNS you can fan-out messages to a large number of subscribers, including distributed systems and services, and mobile devices. It is easy to set up, operate, and reliably send notifications to all your endpoints – at any scale. You can find more information here: <https://aws.amazon.com/sns>

Steps:

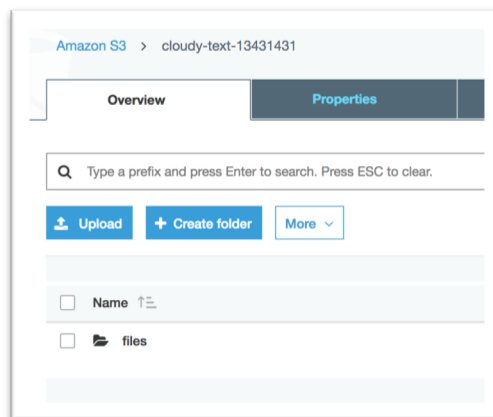
- 1) Make sure you are in the **Ireland** region.
- 2) The first step of building our lab will be to create an IAM role that will give our Lambda the necessary permissions to perform its operations. To do this, open the **IAM** service.
- 3) Select **Roles** and then the **Create role** button.
- 4) Our role will be attached to our lambda functions, so in the first step of the wizard select **Lambda** and then select the **Next: Permissions** button.
- 5) On the second step of the wizard we will chose necessary policies. We will choose the following policies:
 - AmazonSNSFullAccess
 - CloudWatchFullAccess
 - AmazonDynamoDBFullAccess
 - AmazonPollyFullAccess
 - ComprehendFullAccess
 - AmazonS3FullAccess

Note: In real life scenario, you should limit the permission to those exactly required for the API methods and resources that you require.

- 6) On the last step of the wizard provide the name: *CloudyLambdaRole* and select the **Create role** button.

The screenshot shows the 'Create role' wizard in the AWS IAM console, specifically the 'Review' step (Step 3 of 3). The role name is 'CloudyLambdaRole'. The role description is 'Allows Lambda functions to call AWS services on your behalf'. The trusted entities are 'AWS service: lambda.amazonaws.com'. The policies listed are AmazonSNSRole, AmazonSNSFullAccess, CloudWatchFullAccess, AmazonDynamoDBFullAccess, AmazonPollyFullAccess, ComprehendFullAccess, and AmazonS3FullAccess.

- 7) Next we will be creating a new S3 bucket where we will upload our files. Open the **S3** service and **create** a new S3 bucket.
- 8) In the bucket itself, create a new folder and name the folder *files*



- 9) Click the bucket name and open **Permissions** tab.
- 10) Click on **Edit** and untick **Block new public ACLs and uploading public objects** and **Remove public access granted through public ACLs**

Cloud Builders' Day Cloud Technology Workshop

Manage public access control lists (ACLs) for this bucket
Access control lists (ACLs) are used to grant basic read/write permissions to other AWS accounts.

☐ Block new public ACLs and uploading public objects (Recommended) ⓘ

☐ Remove public access granted through public ACLs (Recommended) ⓘ

Manage public bucket policies for this bucket
Bucket policies use JSON-based access policy language to manage advanced permission to your Amazon S3 resources.

☒ Block new public bucket policies (Recommended) ⓘ

☒ Block public and cross-account access if bucket has public policies (Recommended) ⓘ

⚠ Access status appears as **Only authorized users of this account** for a bucket with a public bucket policy. By choosing this option, users outside this account cannot access that bucket until the public bucket policy is removed.

- 11) Type **Confirm** in the pop-up window and click **Confirm** button
- 12) Don't close this window, in a new tab, open the **DynamoDB** service.
- 13) Select the **Create table** button.
- 14) Provide a name for your table, for example: **conversions** with a primary key value of **name**

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. 1 partition the data, and sort data within each partition.

Table name* ⓘ

Primary key* Partition key

String ⓘ

☐ Add sort key

- 15) Click **Create**.
- 16) Again, don't close this window. In a new browser tab open the **Lambda** service.
- 17) Select the **Create function** button.
- 18) In the first step of the wizard, provide a name for your function (*Cloudy_ToAudio*), choose **Runtime** = *Python 2.7* and choose the role which you have created before: *CloudyLambdaRole*.

Author from scratch ⓘ

Name

Runtime
 ▼

Role
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.
 ▼

Existing role
You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.
 ▼

- 19) In the next step of the wizard, we will first configure a trigger. Select **S3** from left side menu.

20) In the **Configure triggers** section, configure the window based on the following instructions:

- Bucket: The S3 Bucket which you have created previously.
- Event Type: PUT
- Prefix: files
- Suffix: .txt

21) Select the **Add** button and then the **Save** button.

22) Download the python file which is provided by this link:

<https://s3-eu-west-1.amazonaws.com/cloudbuildersday/lab-serverless/cloudy-ToAudio.py.zip>

23) **Paste** the code text in the python file into the form in the Lambda wizard.

24) Create two environment variables, one **PHONE** – with your phone number, and a second **DB_NAME** with the name of the DynamoDB table that we have created previously.

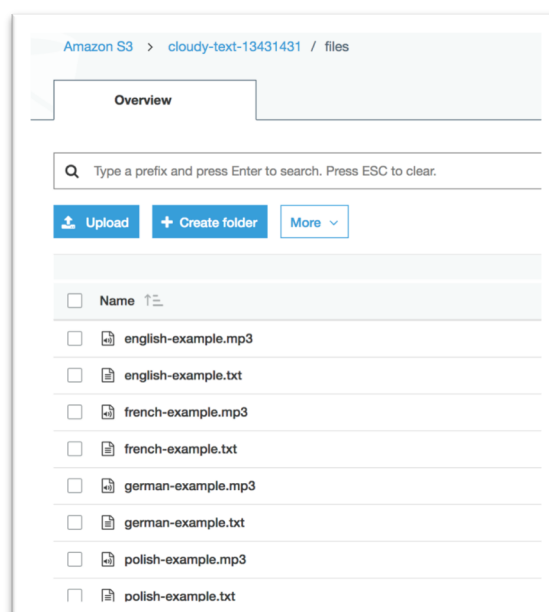
25) While still in the wizard window, increase the **Timeout** of the function to 30 seconds.

26) Select the **Save** button.

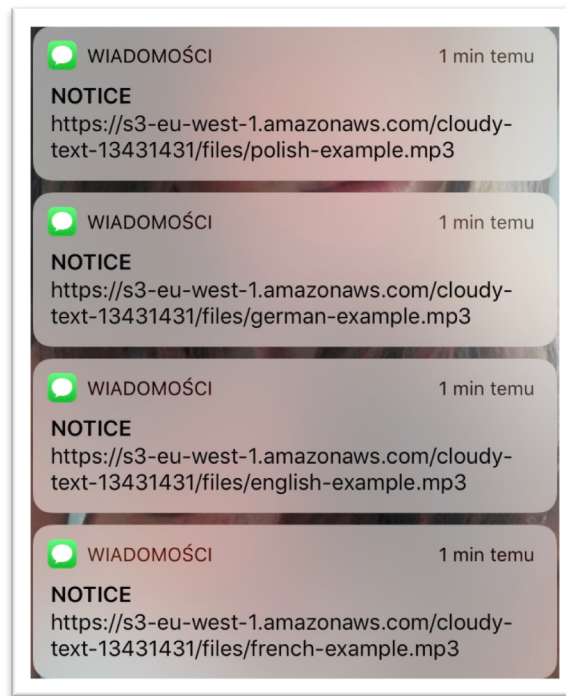
27) Now it is time to test our application. You can find some example files in different languages under this link:

<https://s3-eu-west-1.amazonaws.com/cloudbuildersday/lab-serverless/files.zip>

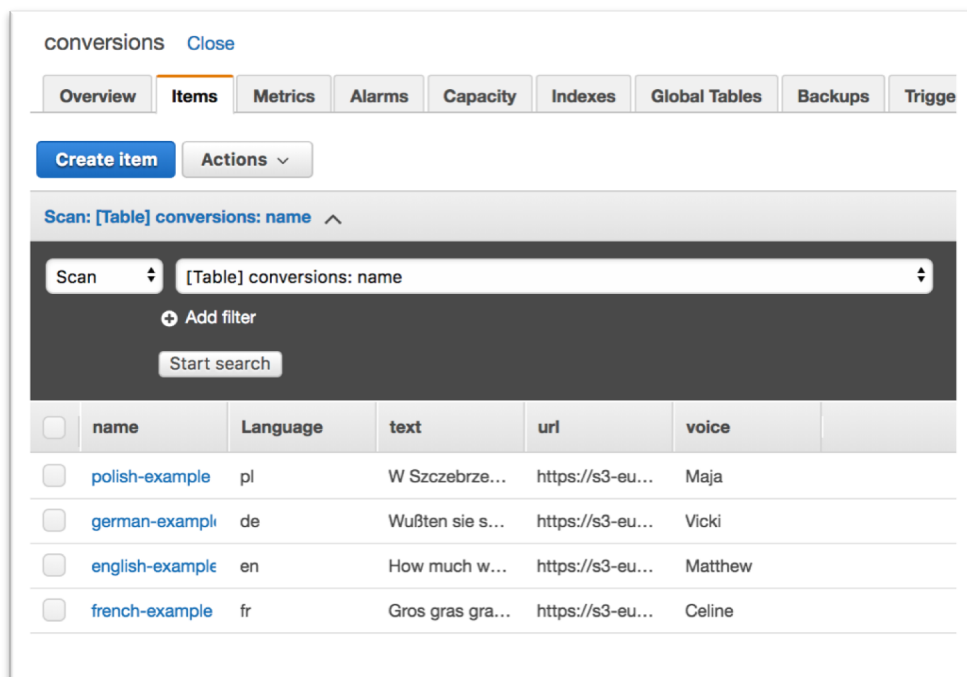
28) After uploading text files to your S3 bucket, you should quickly see new mp3 files alongside the original text files.



29) You should also get SMS(s) messages with links to your mp3 files.

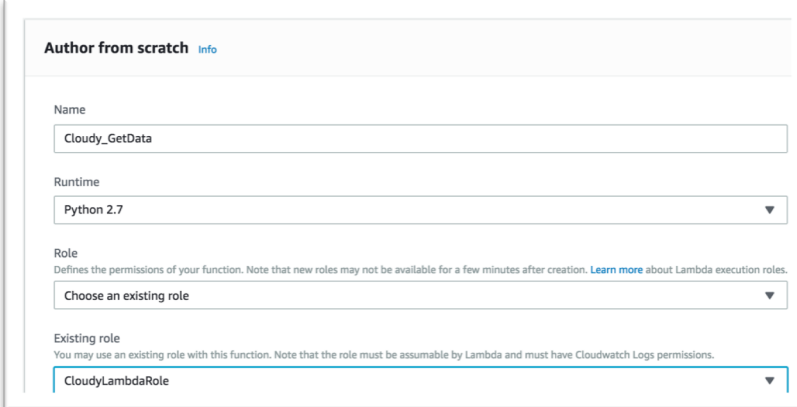


30) Now check the content of your DynamoDB table. You should see that you have new items in your database.



31) Now we will build the second part of our application – a RESTful web service. Thanks to the RESTful web service we will be able to retrieve audio files. Again open the **Lambda** service and select the **Create function** button.

32) In the first step of the wizard, provide the name of your function (*Cloudy_GetData*), choose **Runtime** = *Python 2.7* and then choose the role which you have created previously: *CloudyLambdaRole*.



The screenshot shows the 'Author from scratch' wizard in the AWS Lambda console. The 'Name' field contains 'Cloudy_GetData'. The 'Runtime' dropdown is set to 'Python 2.7'. The 'Role' dropdown is set to 'Choose an existing role'. Below this, the 'Existing role' section shows a dropdown with 'CloudyLambdaRole' selected. The wizard is titled 'Author from scratch' with an 'Info' link.

33) On the second step of the wizard add a new trigger. Select the **API Gateway** button on the left.

34) In the **Configure triggers** section choose **Create a new API**, and then fill out the wizard based on following configurations in the **Additional settings** section:

- API name = CloudyAPI
- Deployoment stage = PRD
- Security = Open

35) Select the **Add** button and then the **Save** button.

36) Download the python file which is provided by this url:

<https://s3-eu-west-1.amazonaws.com/cloudbuildersday/lab-serverless/cloudy-GetData.py.zip>

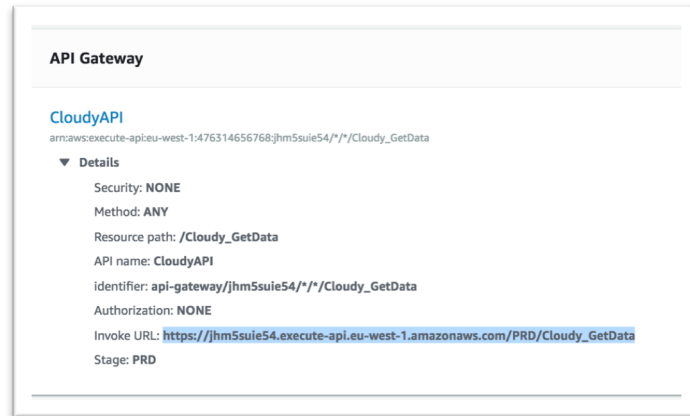
37) **Paste** the code from file into the form in the Lambda wizard.

38) Create a new environment variable: **DB_NAME** with the name of DynamoDB table that we have created previously.

39) Select the **Save** button to save the function.

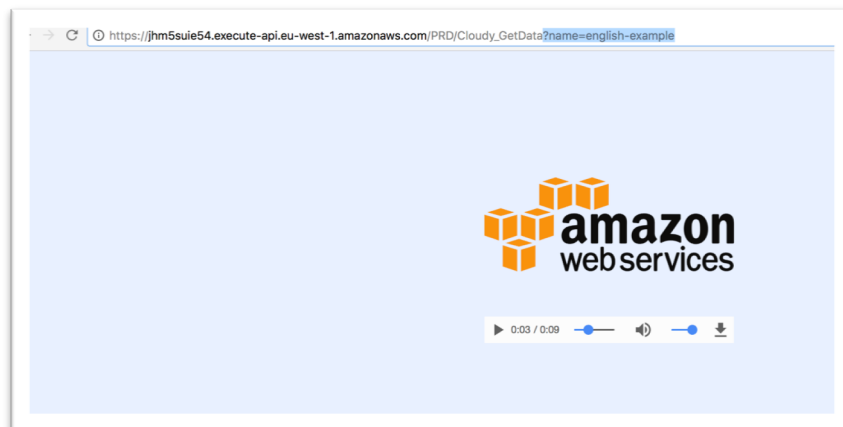
40) In the **API Gateway** section (click on your trigger), copy the **Invoke URL** attribute.

Cloud Builders' Day Cloud Technology Workshop



41) Paste this link in a browser and add the following at the end of the text:
`?name=FILE_NAME`, where `FILE_NAME` is the name of one of your files that you have previously upload to S3.

42) If everything worked correctly, you should see a webpage similar to the one below:



Summary:

Congratulations! In just 20 minutes you have manage to build real application! An A-to-Z serverless application and without provisioning any servers! The application is Highly Available (HA) and is fully scalable. In addition, your application is using AI services to make it even more interesting and powerful.

Bonus Lab – Neural network and digits recognition

In this is bonus lab we will work with neural networks to build a model that will be able to recognize hand written digits.

Services which you will be using in this lab:

- Amazon SageMaker - Amazon SageMaker is a fully-managed platform that enables developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale. Amazon SageMaker removes all the barriers that typically slow down developers who want to use machine learning. You can find more information here: <https://aws.amazon.com/sagemaker/>

Steps:

- 1) Make sure you are in the **Ireland** region.
- 2) Open the **Amazon SageMaker** service.
- 3) Select the **Create notebook instance** button.
- 4) Provide a name for your instance. For example: *CloudyAI*
- 5) In the **IAM Role** section, create a new IAM role for your notebook.
- 6) Select **Any S3 bucket** and click **Create role**
- 7) Select the **Create notebook instance** button. It will take about 15 minutes for your instances to be ready.
- 8) When your notebook is in the **InService** state, select the **Open Jupyter** button.
- 9) Open the **SageMaker Examples / sagemaker-python-sdk / mxnet_mnist_with_gluon** and click **Use**
- 10) Click **Create copy** in pop-up window
- 11) Go through each step of the document by selecting Run button.
- 12) The longest step will be the learning/teaching model [6].

Cloud Builders' Day Cloud Technology Workshop


```
In [*]: m.fit(inputs)

[Epoch 8] Training: accuracy=0.999997
[Epoch 8] Validation: accuracy=0.973800
[Epoch 9 Batch 100] Training: accuracy=0.991188, 2871.512878 samples/s
[Epoch 9 Batch 200] Training: accuracy=0.990498, 2535.624943 samples/s
[Epoch 9 Batch 300] Training: accuracy=0.990233, 1874.172345 samples/s
[Epoch 9 Batch 400] Training: accuracy=0.989825, 4315.171967 samples/s
[Epoch 9 Batch 500] Training: accuracy=0.990040, 1810.972993 samples/s
[Epoch 9] Training: accuracy=0.989633
[Epoch 9] Validation: accuracy=0.974700
[Epoch 10 Batch 100] Training: accuracy=0.990000, 1835.364749 samples/s
[Epoch 10 Batch 200] Training: accuracy=0.991095, 2984.101597 samples/s
[Epoch 10 Batch 300] Training: accuracy=0.991794, 2203.365220 samples/s
[Epoch 10 Batch 400] Training: accuracy=0.991920, 2151.479450 samples/s
[Epoch 10 Batch 500] Training: accuracy=0.991796, 4360.527301 samples/s
[Epoch 10] Training: accuracy=0.991100
[Epoch 10] Validation: accuracy=0.977300
[Epoch 11 Batch 100] Training: accuracy=0.993564, 4370.204741 samples/s
[Epoch 11 Batch 200] Training: accuracy=0.993333, 1764.945191 samples/s
[Epoch 11 Batch 300] Training: accuracy=0.992957, 4271.707337 samples/s
[Epoch 11 Batch 400] Training: accuracy=0.993067, 2403.433555 samples/s
```

13) When the model will be ready and deployed, you can test it in the browser by writing some digits.

```
In [8]: from IPython.display import HTML
HTML(open("input.html").read())

Out[8]:
```



The predictor runs inference on our input data and returns the predic

```
In [10]: response = predictor.predict(data)
print int(response)

5
```

Cleaning

Congratulations, you have managed to complete all the labs! This was intensive, but now we need to make sure that we clean-up our cloud account. This should take approximately 5 minutes, and ensures on-going costs are not incurred. So please complete these steps!

Perform the steps below, in the order that they are written.

Bonus Lab: Neural network and digits recognition

- 1) Open the **Amazon SageMaker** service
- 2) Open the **Notebook** tab and select your notebook. Terminate it.
- 3) In the **Models** tab, delete the model which you have created in your lab.
- 4) In the **Endpoints configuration** tab, delete configurations which are present.
- 5) In the **Endpoints** tab, delete the endpoint.

Lab 8 – Building serverless app with AI capabilities

- 1) Open the **Lambda** service, select your functions and delete them.
- 2) Open the **API Gateway** service and delete the API that was created for your function.
- 3) Open the **DynamoDB** service and delete the table.
- 4) Open the **S3** service and delete the bucket that you have created.

Lab 7 – Business Intelligence in Cloud

- 1) Open the **QuickSight** service
- 2) Select the **Manage Data** button, select your data set and delete it.
- 3) Unsubscribe from the QuickSight service.

Lab 6 - Analytics – Hadoop in practice

- 1) Open the **EMR** service and select your cluster. Terminate it.
- 2) Open the **Athena** service, click on three dots next to name of your table and select the option for deleting it.

Lab 5 - Collecting data in real-time

- 1) Open the **Kinesis** service and delete the kinesis stream that you have created.
- 2) Open the **S3** bucket and delete the S3 bucket that you have created.

Lab 4 - Containers in cloud

- 1) Open the **ECS** service and delete your repository.
- 2) Open the **Clusters** tab, make sure it's empty (if not delete all running tasks), and then delete the cluster.
- 3) Open the **Task definition** tab and delete your definition.

Lab 3 – Developers Tools. CD in practice.

- 1) Open the **CodeDeploy** service and delete your application.
- 2) Open the **CodeCommit** service and delete your repository.
- 3) Open **CodePipeline** and delete your pipeline.

Lab 2 – Autoscaling and IaasC

- 1) Open **CloudFormation** and delete the stack that you have created.

Lab 1 – Preparing Cloud Environment.

- 1) Open the **Cloud9** service and delete your environment.
- 2) Open the **EC2** service and delete all remaining EC2 instances.
- 3) In the **EC2** service, go to Keys tab and delete your key.
- 4) Open the **IAM** service and then the users tab - you should delete the `ide_user` user.
- 5) Open the **Roles** tab and delete roles that you have created.