LCLOUD

Case study:

LCloud Ltd for **Tallyfy Inc.**

CLIENT:

Tallyfy Inc.

FIELD:

CRM, ERP, real-time analytics

SERVICE / APPLICATION:

https://tallyfy.com/

TAGS:

Workflow, organization, application, real-time analytics

DESCRIPTION:

Web application for workflow management in the organization

TIME FRAME OF THE PROJECT:

October - December 2019

AWS SERVICES USED FOR IMPLEMENTATION:

- AWS CLOUDFORMATION
- AWS LAMBDA
- AWS ELASTIC BEANSTALK

SOLUTIONS USED FOR THE IMPLEMENTATION:

- AUTOMATION
- CONTAINERIZATION
- 24/7 MONITORING & SUPPORT

Client

Tallyfy Inc. is an American IT services company that develops software to manage business processes, workflows, support procedure creation and service management. The company develops software to support almost all processes in the organization.

Their services improve workflow and information management in many companies, regardless of their size or scale. The solutions are used by such companies as: Oracle, Nestle, Emerson and healthcare industry entities or governmental institutions.

Challenges

The client came to us with architecture maintained in Rackspace, which was not prepared for fast and flexible development. *Providing infrastructure* ***for new services took a long time and hampered the application development processes.*** *The client involved a large team in the processes of implementation of new versions of the system, so it could not focus enough on the main development of its solution.* ***Tallyfy was interested in the migration of infrastructure to AWS****, automation of the process of delivering solutions for new services and the deployment process.*

An important aspect of the project was also ensuring integration with publicly available tools*, such as the GSuite platform, and* ***simplifying the integration and synchronization procedure for the user****.*

The challenge was **to design an environment that would allow automatic database launch and scaling, ensuring high availability of infrastructure and high level of security and monitoring of resources.**

Challenges – continuation

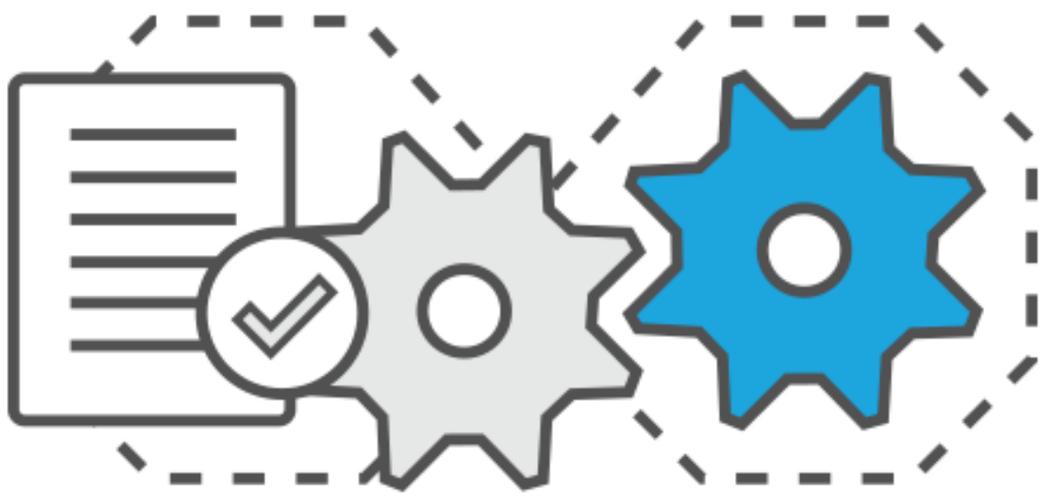
In **the previous solution single instances were used, which was not ready for autoscaling and the Client's application was a monolith**. Moreover, **it was not possible to perform "live" migration**. We could not replicate data between two clouds and perform a simple traffic surge in a fixed time. Therefore, we had to carefully plan scenarios of data migration and domain overvoltage so that the interruption in access to the application was as short as possible.

An important aspect, clearly emphasized by the client, was the goal of lowering infrastructure maintenance costs.

Proposed Solution & Architecture

For Tallyfy client, we **prepared an architecture based on AWS services, providing automation of configuration (integrated with Deploybot service) and application deployments and helped in its migration from the Rackspace cloud**.

In the first stage, the configuration assumed running the existing application in one frontal autoscaling group and preparing a separate dedicated autoscaling group. For example, it was responsible for handling tasks such as sending e-mails. **We used the Amazon Elastic Beanstalk service to quickly deliver a solution for this layer, in the form of a single EB application with two EB environments.**



Proposed Solution & Architecture – continuation

When the basic configuration was ready and the initial migration to AWS was completed, the programmers from the Customer's team started working on improving the application.

The application layer had to be divided into two parts:

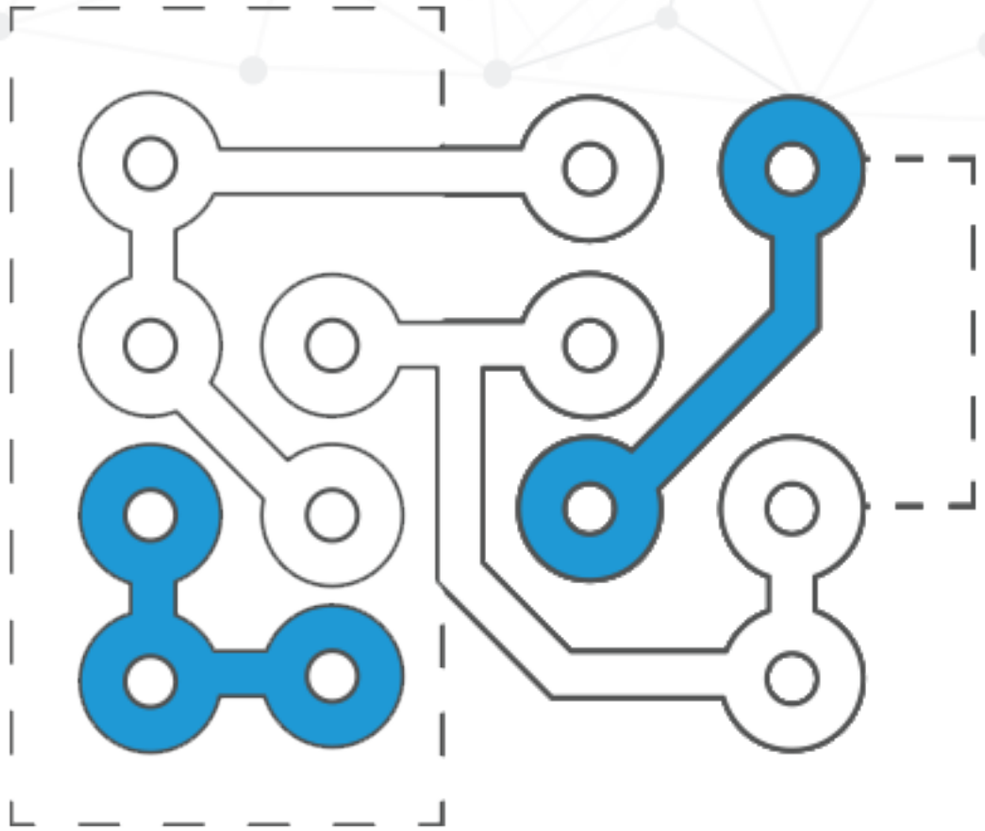
- The application on the client side was based on Node.js and operated from the Amazon S3 bucket through the Amazon CloudFront distribution.
- On the application side, it still runs on instances run by Amazon Elastic Beanstalk environments using the Application Load Balancer and communicates with databases and Amazon SQS queues.

Proposed Solution & Architecture – continuation

The whole configuration is managed by several Amazon CloudFormation stacks.

There is one main stack through which we run several nested stacks to simplify the provisioning of new infrastructure, in the same or another AWS region.

A copy of the templates is stored in our internal repository, where they are approved after any changes are made.



Proposed Solution & Architecture - continuation

The **client decided to use the DeployBot service** to automatically implement their applications to AWS. DeployBot is integrated with GitHub repositories, which contain the application code.

Two DeployBot environments were created for both repositories, which correspond to the versions of AWS environments - staging and production.

Within the environment, we have access to the development and deployment history, define industry and target source servers - in our case Amazon S3 buckets of Amazon Elastic Beanstalk environments - and create additional configuration files and scripts that can be added to the deployment artifacts.

Single DeployBot environment is configured to deploy changes only from a selected branch, such as master.

When the desired changes are combined in the tracked branch, DeployBot automatically starts the building phase.

Proposed Solution & Architecture - continuation

Builds were made in temporary containers launched from a template prepared by us.

During this phase additional dependencies required to run the application were installed and the previously mentioned scripts used during deployment are added.

When the deployment archive was ready, DeployBot automatically started deployment on the target environment.

Proposed Solution & Architecture – continuation

In the case of a client's frontend application, DeployBot simply retrieved data from the deployment archive to the target Amazon S3 bucket and then cancelled the entire Amazon CloudFront distribution.

- In the casesends the deployment image archive to the intermediate Amazon S3 bucket,
  - of API applications, DeployBot,
  - registers a new version of Amazon Elastic Beanstalk application,
  - starts to implement Amazon Elastic Beanstalk into the desired environment.
- Amazon Elastic Beanstalk independently manages the implementation for available instances.
  - A single instance worker environment updates it on site.
  - An environment with multiple API instances carries out the deployment on the fly.

DeployBot is easy to set up. However, it does not provide many configuration options. Therefore, the client is considering switching to GitHub Actions.

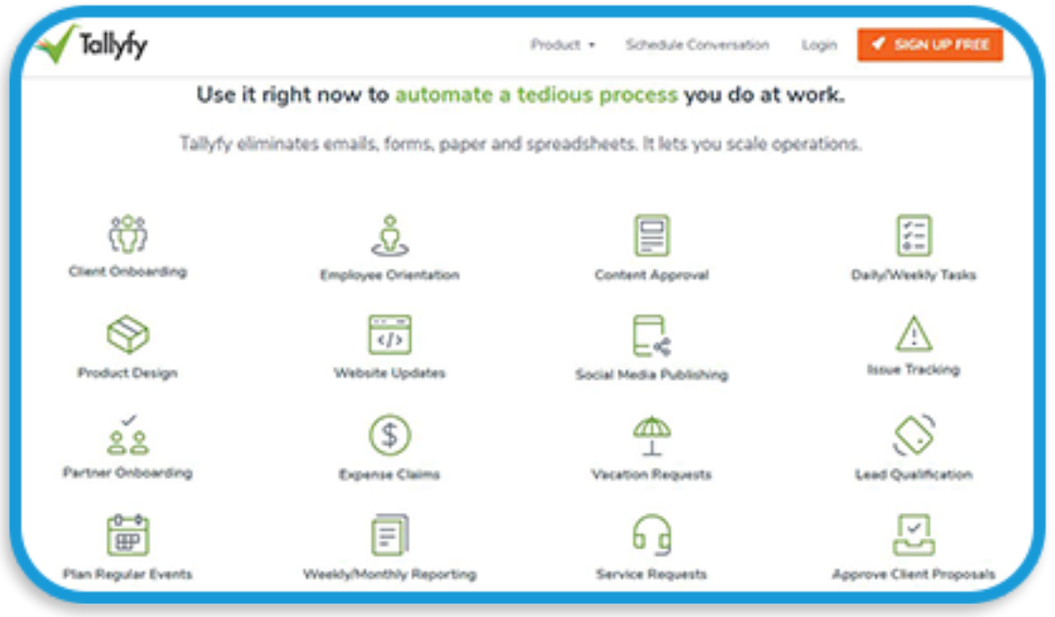
Proposed Solution & Architecture - continuation

The Client's wish was **also to obtain detailed information about the state of the environment, events occurring in it, the possibility to predict the necessity of its development**.

Moreover, the Client **wanted to obtain many metrics concerning the realized processes, including the deployment process**.

As a tool to comprehensively implement these tasks we used Amazon CloudWatch, where we configured the necessary metrics and thanks to AWS Lambda, we automated the alert process.

The client also uses an external system to monitor API availability: https://api-status.tallyfy.com/.



Proposed Solution & Architecture – continuation

On the client's side during the implementation there was a team consisting of 3 programmers, whose work was supervised by **CTO**, who also acted as **Project Manager**.

Our team consisted of a dedicated **Project Manager, certified Solutions Architect, 2 DevOpses and one SysOps**.

To achieve our goals we used the services: Amazon CloudFront, Amazon EC2, AWS Lambda, Amazon RDS, Amazon S3, Application Load Balancer.

What AWS services were used as part of the solution

Amazon CloudFront

Amazon VPC

Application Load Balancer

Amazon S3

Amazon CloudWatch

AWS Lambda

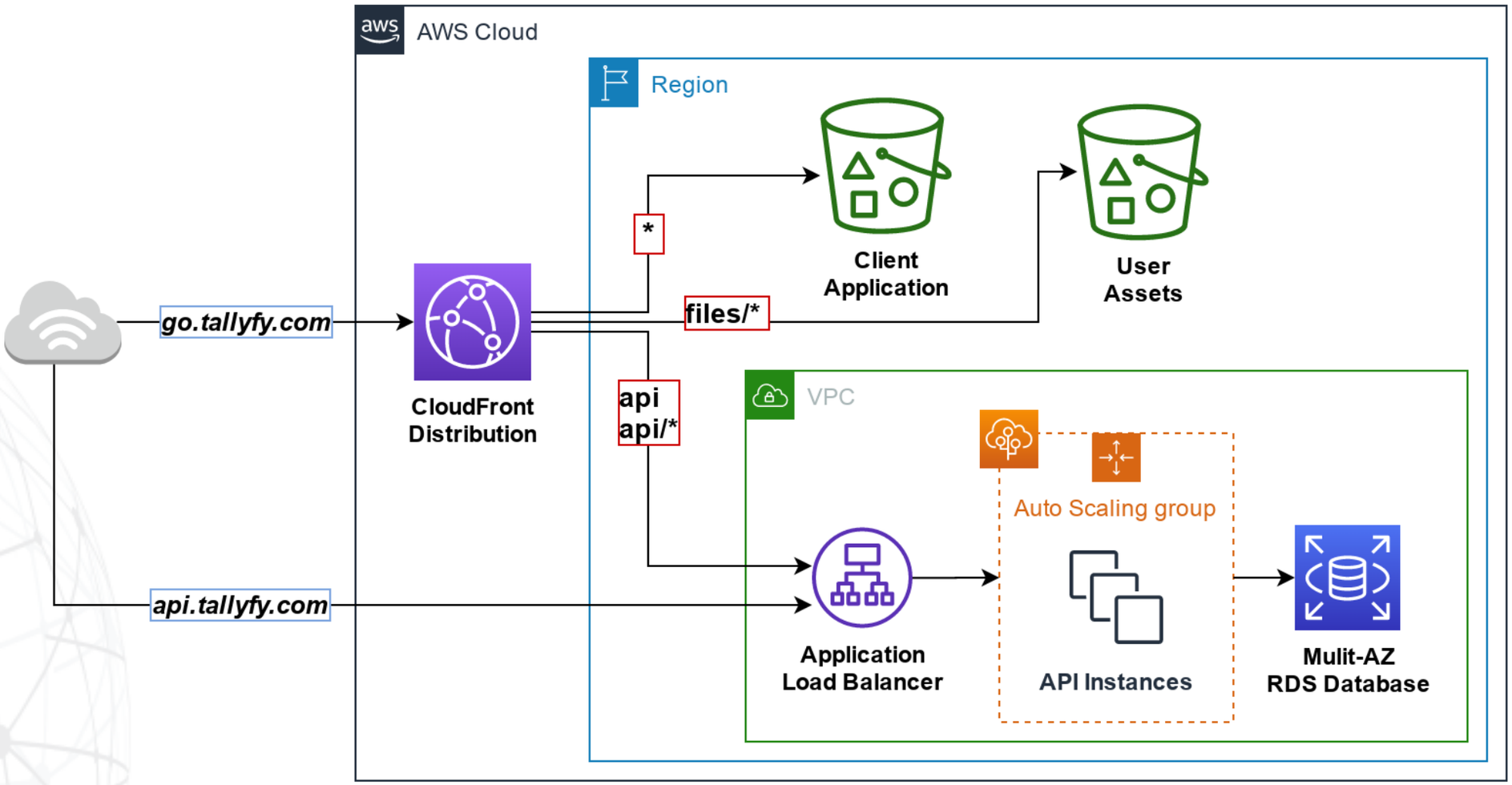
Amazon RDS

Third party applications and solutions used in the project

The project used a number of solutions and applications that enriched and improved it. **GitHub** was used as the code repository. **Redmine** was used to manage milestones.

We implemented ongoing communication in the project using **Slack**.

Outline of AWS infrastructure and services:



Outline of AWS infrastructure and services used for Tallyfy Inc.

Results from the solution

The results of the project are:

- **Migration of resources from a traditional Rackspace solution to AWS**, which helped to significantly minimize overall infrastructure costs.
- The **adaptation of the application to the possibility of serving is done from the cloud**.
- The optimal architecture of the solution, having a significant impact on the effectiveness of maintaining the environment, **increasing its efficiency and achieving high availability**.
- **Minimization of the number of Amazon EC2 instances to Amazon S3** - static application elements have been moved to Amazon S3.
- Comprehensive environmental information, thanks to detailed metrics implemented in Amazon CloudWatch.
- Automatic implementation of new application versions

Metrics for Success

Thanks to the implementation of solutions based on AWS cloud and LCloud services we have achieved our goals:

- Thanks to the migration of resources to AWS, **high application availability was achieved - 100 % availability according to monitoring statistics**.
- **Reduction of infrastructure costs by about 90% per year.**
- The infrastructure is delivered very efficiently, which **shortens the waiting time for the resources to be activated for new customer services by 7 hours** (for each new service).
- The team of programmers involved in **the deployment process** so far can focus on application development, as the overall process, **which used to take 5 hours, now takes 1 hour with a preparation phase**.

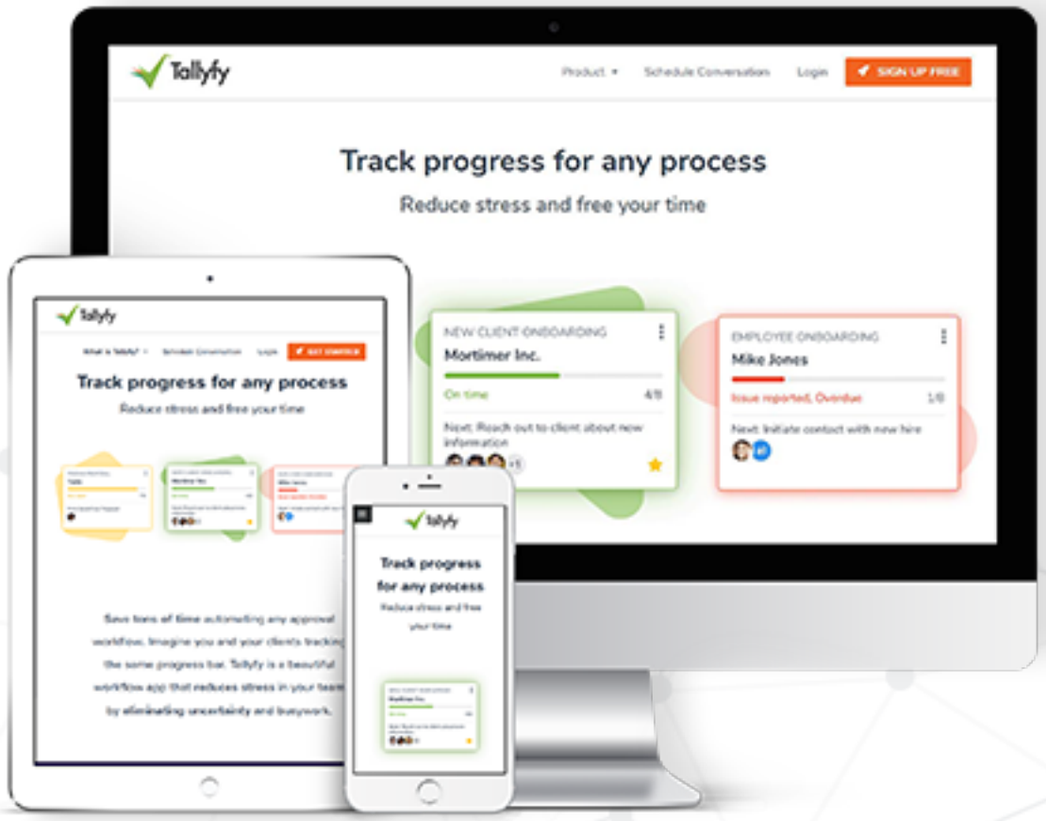
Metrics for Success

Thanks to the implementation of **AWS services, the scope of monitoring has increased**, and **the level of environmental security has been improved**. Access to all instances is limited to one Bastion instance, through which connections are tunneled or proxied.

Implemented hierarchical network topology and firewall of the application layer, based on Security Group rules and network layer based on Network Access Control List rules. CNAME records used in the private DNS zone were replaced with static IP addresses.

Lessons learned

Setting up the DeployBot service is easy. However, **it does not provide many configuration options**. Therefore, the client is considering switching to GitHub Actions.



When migrating monolithic applications to the cloud, it is important to remember that a very good understanding of its functionality, logic and dependencies and close cooperation with the developers who are the authors are the key to success. Much attention should be paid to the stage of scenario design in order to eliminate the inconvenience of migration.

Lessons learned

It is important to plan the automation of processes, such as providing infrastructure for new services, the process of implementing new versions of applications or monitoring, immediately after the migration, because only then will the benefits of using cloud computing be realistically visible.

The expected cost optimization is achieved mainly due to the change of approach to application serving and implemented automation, hence the customer's openness to such changes in processes is the basis for the success of the undertaking.

Contact us

LCloud Sp. z o.o.
ul. Złota 59
00-120 Warszawa

+48 22 355 23 55

kontakt@lcloud.pl

Quick contact:

biuro@lcloud.pl

Business:

sale@lcloud.pl
+48 22 355 23 57