TUNISIAN REUBLIC
Ministry of Higher Education and Scientific Research
University of Carthage
National Institute of Applied Sciences and Technology

Graduation Project

to obtain the

National Engineering Diploma
in Applied Sciences and Technology

Specialty: Software Engineering

Title
Title

Prepared by

Name and Surname

Hosted by ……..

Company's logo

Defended on the …….. 2021 in front of a jury composed of:

……	President of the jury
……	Reviewer
……	Supervisor at the Company
……	Supervisor at INSAT

School year: 2020 / 2021

TUNISIAN REUBLIC
Ministry of Higher Education and Scientific Research
University of Carthage
National Institute of Applied Sciences and Technology

Graduation Project

to obtain the

National Engineering Diploma
in Applied Sciences and Technology

Specialty: Software Engineering

Title
Title

Prepared by

Name and Surname

Hosted by ……..

Company's logo

Supervisor at the Company ……	Supervisor at INSAT ……
Date:	Date:

School year: 2020 / 2021

Dediction

This work is dedicated to the individuals who have always been there and have always supported me with their acts, ideas, and wills. With genuine gratitude and warm regard, I dedicate this work to:

My beloved family

Your sacrifices, endless sleepless nights, and unconditional love made me who I am today. I can't emphasize enough how your support and guidance were behind every achievement I've made or am soon to reach.

My friends

You have supported me through hardships and difficulties. I will always appreciate all they have done to make me feel at home when I am far away from my family.

My teachers and professors

I've been extremely likey and privileged to have such wonderful, supporting, and caring teachers throughout my learning journey.

My colleagues at company-a

I will never forget how you pushed me to be the best engineer I can be. Your belief and trust in me have consistently raised the bar for what I can accomplish and how I can contribute to the collective.

Foulen ben Foulen

Contents

List of Figures

List of Tables

List of acronyms

- **ALB** = **A**pplication **L**oad **B**alancer
- **AMI** = **A**mazon **M**achine **I**mage
- **ARN** = **A**mazon **R**esource **N**ame
- **AWS** = **A**mazon **W**eb **S**ervices
- **CD** = **C**ontinuous **D**elivery
- **CI** = **C**ontinuous **I**ntegration
- **CLI** = **C**ommand **L**ine **I**nterface
- **DB** = **D**ataBase
- **E2E** = **E**nd **T**o **E**nd
- **ECR** = **E**lastic **C**ontainer **R**egistry
- **ECS** = **E**lastic **C**ontainer **S**ervice
- **EKS** = **E**lastic **K**ubernetes **S**ervice
- **HCL** = **H**ashiCorp **C**onfiguration **L**anguage
- **IAM** = **I**dentity **a**nd **A**ccess **M**anagement
- **MFA** = **M**ulti **F**actor **A**uthentication
- **NLB** = **N**etwork **L**oad **B**alancer
- **POC** = **P**roof **o**f **C**oncept
- **PROD** = **P**roduction
- **RDS** = **R**elational **D**ata**b**ase **S**ervice
- **RPO** = **R**ecovery **P**oint **O**bjective
- **RTO** = **R**ecovery **T**ime **O**bjective

- **SNS** = **S**imple **N**otification **S**ervice
- **SSM** = **A**WS **S**ystems **M**anager
- **STS** = **S**ecurity **T**oken **S**ervice
- **TLS** = **T**ransport **L**ayer **S**ecurity
- **VPC** = **V**irtual **P**rivate **C**loud
- **WAF** = **W**eb **A**pplication **F**irewall

Introduction

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1

General context

Introduction

The first parts of this chapter will be devoted to introducing the host company where this project was conducted, the partners we worked with, and the data-exchange solution we're going to deploy. We will then describe the motivation behind this project and the proposed solution. Finally, we will conclude by identifying the work methodology and the project timeline.

1 Host organization

The work presented in this report was conducted as part of the DevOps team in company-a. This section will be dedicated to describing the company and its field of expertise and distinguished partners.

1.1 Description of the company

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, [1]

1.2 Fields of expertise

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

- **Cloud and DevOps:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,
- **Data Science and Machine Learning:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,
- **Full Stack Web Development:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.
- **Mobile development:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

1.3 Partners

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, :

- **abc1:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.
- **abc2:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.
- **abc3:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.
- **abc4:** placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

2 Project actors

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

2.1 Partnar-a

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.2 Partnar-b

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.3 Partnar-c

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

3 Motivations and problem statement

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

3.1 Motivation

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

3.2 Problem statement

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

4 Proposed solution

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

5 Project framework and methodology

This section will introduce the Agile scrum methodology and illustrate why our team chose it to implement our project!

5.1 Agile methodology

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.

3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

Our team uses a combination of two agile frameworks in a way that matches our requirements and team size:

- **SCRUM framework:** Is a sort of agile approach that breaks projects down into manageable parts known as "sprints". It also specifies roles and responsibilities within the SCRUM team.
- **Extreme programming framework:** It creates a minimal, yet effective environment that allows teams to become highly productive. XP emphasizes the importance of pair-programming sessions which is something that influenced a lot our daily work routine.

5.2 Team and roles

Our team structure contains the following roles as it's recommended by SCRUM:

- **Developers:** A developer on a scrum team is anyone on the team who is delivering work, including team members who do not work in software development.
- **Product Owner:** Maintains the product's vision and prioritizes the product backlog.
- **Scrum Master:** Assists the team in making the best use of scrum to produce the product.

The following table1.1 shows the structure of our scrum team:

Role	Members
Product Owner	desciption
SCRUM Master	descrition
Developers	8 Software Engineers from company-a: - 3 DevOps Engineers - 5 Web Development Engineers

Table 1.1: Scrum team structure

5.3 The SCRUM events

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

6 Project timeline

My contribution to the project will cover the first eight sprints as follows1.2:

Sprint	Duration	Working days	Tasks
0	26/02 - 02/03	5	- On-boarding - aaa
1	02/03 - 13/03	10	- aaa - aaa
2	16/03 - 27/03	10	- aaa - aaa
3	30/03 - 10/04	10	- aaaa - aaa - aaa
4	13/04 - 24/04	10	- aaa - aaa - aaa
5	27/04 - 08/05	10	- aaa - aaaa - aaaa
6	11/05 - 29/05	15	- aaa - aaa - aaa
7	01/06 - 11/06	10	- aaaa - aaa

Table 1.2: Sprints

Conclusion

This chapter was dedicated to introducing the host organization and its partners in the first place. Then we presented the problem statement and the proposed solution. It also allowed us to demonstrate the work approach we will use throughout our project.

The following chapter will go over some of the theoretical aspects of DevOps and Cloud Engineering that are required for the project's implementation.

2

State of The Art

Introduction

Analyzing the theoretical concepts essential for our project and the various instruments at our disposal is an unavoidable stage in achieving the project's objectives. As a result, we'll start with a description of Cloud Computing. Following that, we'll go through DevOps methods, followed by a talk on Multi-Account Strategy in the Cloud, which will help us create a better solution.

1 Cloud Computing

First, we'll go through the basics of Cloud Computing, including its definition, characteristics, service models, and deployment models. Then we'll look at some of the various cloud providers before diving into AWS compute offerings.

1.1 Definition

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.2 Characteristics of Cloud Computing

These five characteristics of cloud computing are what makes the technology so in-demand today:

- **On-demand Self Service:** Cloud computing allows users themselves to provision, manage, and monitor resources in real-time without the need for human interaction. This is usually done through a web-based self-service management console, an API, or a command-line tool.
- **Broad Network Access:** Cloud computing is accessed over a network, which is usually the internet. Meanwhile, private cloud services might be accessed from anywhere within the company.
- **Resource Pooling and Multi-tenancy:** Computing resources like networks, servers, storage, applications, and services can be pooled to serve numerous customers by logically separating them. This is accomplished through the use of a multi-tenant paradigm, which enables several clients to utilize the same application or physical infrastructure while maintaining data confidentiality and privacy.

- **Rapid Elasticity:** Cloud computing facilitates the rapid and automated provisioning and disposal of resources. It also allows users to quickly and easily scale in response to demand.
- **Measured Service:** Each occupant's resource usage is tracked, monitored, managed, and reported on. This provides both the service provider and the customer with transparency.

1.3 Service models

Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) are the three core concepts of Cloud architecture. As shown in figure 2.1, each service model reduces the ratio of client intervention and control over the end service as it progresses from left to right. In IaaS, the cloud provider manages the operating system and underpinning infrastructure, while the user manages the layers above the operating system. When moving to PaaS, the customer relinquishes management of the Middleware and Runtime layers, focusing instead on the App and its data. The cloud provider oversees everything and offers the final software as a service that runs on the cloud in the case of SaaS.

Figure 2.1: Cloud Service Models

Every Cloud Service Model is best suited to a specific use case or user group. The most popular is SaaS, which anyone can use without needing to know anything about programming or cloud engineering. In fact, most of us use SaaS in our daily lives, such as Google Docs, Slack, Dropbox, and others. Because you only pay for the service/storage you use, SaaS pricing structures are simple, making it an excellent alternative for small enterprises and startups.

Paas, on the other hand, is better for developers because it provides them with ready-to-use platforms to deploy their services. Because it provides a high level of abstraction and delegates infrastructure, security, scaling, as well as patching and upgrading operating systems to the cloud provider, this service model can significantly speed up development. PaaS also has the benefit of a pay-as-you-go pricing model, which is comparable to SaaS and can be very cost-effective.

IaaS is the most sophisticated because it gives engineers full control over the provisioning and management of a wide range of computing infrastructures, including storage, servers, and networking hardware. IaaS is by far the most difficult to manage because it necessitates expertise in a wide range of engineering fields, including network, security, and operating systems. However, An IaaS solution can save a great deal of money and boost efficiency if it's well-designed and implemented. The IaaS pricing model is also complicated because it includes paying for provisioned resources, whether or not they are used.

1.4 Deployment models

There are five cloud deployment models:

- **Public Cloud:** The public cloud is one in which cloud infrastructure services are made available to the general public or major industry groups via the internet. In this cloud model, the infrastructure is owned by the entity that provides the cloud services, not by the consumer.
- **Private Cloud:** The public cloud deployment model is diametrically opposed to the private one. This technology could be housed on a physical infrastructure owned and controlled by the customer.
- **Hybrid cloud:** Hybrid cloud computing provides the best of both worlds by bridging the public and private worlds with a layer of proprietary software.

- **Community cloud:** As the title suggests, it is utilized by many companies with comparable needs. It hosts a highly specialized business application that several enterprises use.
- **Multi-cloud:** It is comparable to the hybrid cloud deployment strategy, which blends public and private cloud resources. Instead of combining private and public clouds, multi-cloud uses an extensive range of public clouds.

1.5 Cloud providers

Four cloud service providers dominate the global cloud market: Alibaba in China and Asia-Pacific, and AWS, Microsoft, and Google in the rest of the globe.

We can see by examining the cloud market using the Gartner Magic Quadrant2.2, that Amazon AWS, Microsoft Azure, and Google Cloud are assuming market leadership:

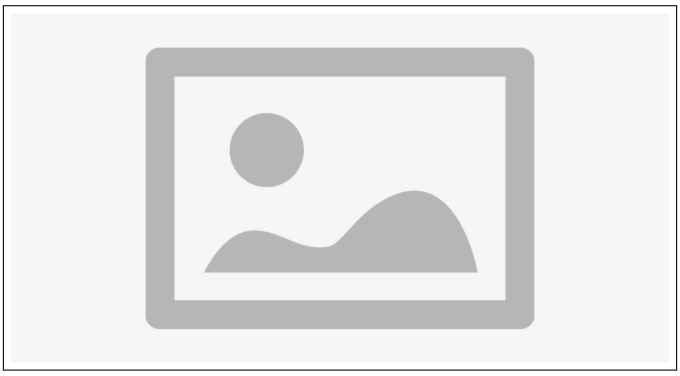

Figure 2.2: Top Cloud Service Provider
[?]

Amazon AWS will be discussed in further detail in the following section since we'll be using it as our primary cloud provider and as one of our partners.

1.6 AWS compute services

Today, Amazon Web Services (AWS) is the world's leading public Cloud computing services provider. It owns and administers the network-connected hardware required for application services, while users configure and employ what they need using a web application or command lines. It is the most comprehensive and commonly used Cloud platform globally, with over 200 fully-featured services available from data centers around

the world. Millions of customers, including the fastest-growing startups, most prominent corporations, and top government agencies, rely on AWS to save costs, become more agile, and innovate more quickly.

The Amazon Web Services portfolio includes more than 100 services, including computation, databases, infrastructure management, application development, and security. These services are classified as follows:

- Compute
- Storage databases
- Data management
- Migration
- Hybrid cloud
- Networking
- Development tools
- Management
- Monitoring
- Security
- Governance
- Big data management
- Analytics
- Artificial intelligence (AI)
- Mobile development
- Messages and notification

2 DevOps

This section will provide a basic description of DevOps before discussing its benefits. Then, we'll go through certain aspects of CI/CD pipelines in which DevOps plays an important part.

2.1 Definition

DevOps, an acronym for "development and operations," is a collection of techniques, technologies, and cultural philosophies that enable a company to produce applications and services fast. It also allows products to move quicker from the drawing board to the market than traditional software development since operations and development engineers collaborate closely throughout the lifecycle, from design to development to production support. Indeed, operations workers and developers frequently use many of the same technologies in tandem, helping the task move much more smoothly and quickly.

As seen in the figure below2.3, DevOps team members carry out duties that are often divided among three teams: development(plan, code, and build), operations(release, deploy, and operate), and quality assurance(test and monitor).

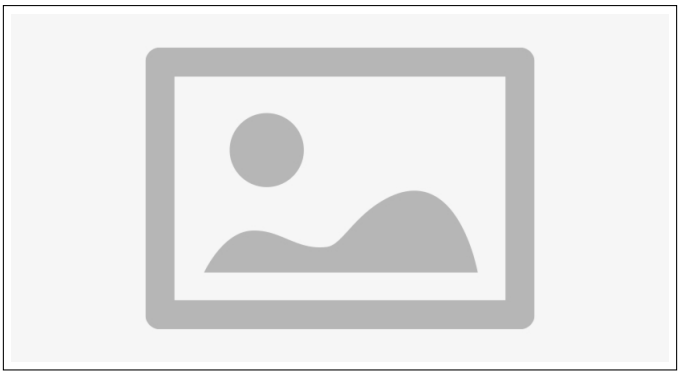

Figure 2.3: DevOps process

2.2 Benefits

Adopting the DevOps approach has various advantages:

- **Speed:** You may boost customer inventiveness, enhance consumer responsiveness, and accomplish productivity and development by moving quicker. The DevOps approach enables development and operations teams to meet these goals. It provides teams with the resources and methods to govern and upgrade applications more quickly.
- **Fast deliver:** To accelerate the innovation and refinement of goods and enhance the pace and frequency of launches, we can meet consumer needs and gain a competitive advantage by introducing new features and correcting issues as soon as feasible.
- **Scalability:** Infrastructure and construction processes may now be handled and managed at scale, thanks to DevOps. Automation and precision help handle complex systems in a practical and risk-free manner.
- **Improved collaboration:** The DevOps strategy stresses topics such as accepting responsibility for building more successful teams. Development and operations teams collaborate extensively, exchanging responsibilities and merging procedures. As a consequence of this, they may eliminate inefficiencies and save time.
- **Security:** Moving ahead with the DevOps strategy has no impact on security because of the automated compliance regulations, stricter controls, and configuration management approaches deployed. For example, we can detect and monitor enforcement at any scale by leveraging infrastructure as code and policy as code.
- **Reliability:** Adoption of DevOps enhances service reliability by increasing the availability and connectivity of services required for the efficient operation of a business.

2.3 Infrastructure as Code - Terraform

Infrastructure as Code (IaC) refers to managing infrastructure (networks, virtual machines, load balancers, and connection architecture) in a descriptive model, utilizing the same versioning that the DevOps team does for source code.

Among the advantages of the IaC, we distinguish:

- **Speed:** It enables you to automate all infrastructure processes and adjustments to save time and money while reducing the chance of human mistakes.
- **Source control:** The code can be verified against a shared repository for increased transparency and accountability.
- **Consistency:** It emphasizes predictable, repeatable methods for provisioning and modifying systems and their configuration.
- **Reusability:** It makes it easier to create reusable modules, such as those that mimic development and production environments.

IaC is supported by popular third-party platforms, such as Terraform, Ansible, Chef, and Pulumi, to manage automated infrastructure.

In our project, we will be using Terraform, an open-source IaC software tool created by HashiCorp. It uses a declarative configuration language called HCL developed by the same company.

Terraform will construct a deployment plan detailing resources to be created, updated, or destroyed using the configuration code (Desired state) and the present state referred to as the backend (Current state). Terraform can then interface with the Cloud API using particular providers to make the modifications.

2.4 CI/CD

CI/CD tools are at the foundation of DevOps and the key to its success. The abbreviation CI/CD has numerous meanings. The term CI refers to continuous integration, a type of automation. Continuous integration includes making regular changes to the application code, running all necessary tests, and committing the changes to a common

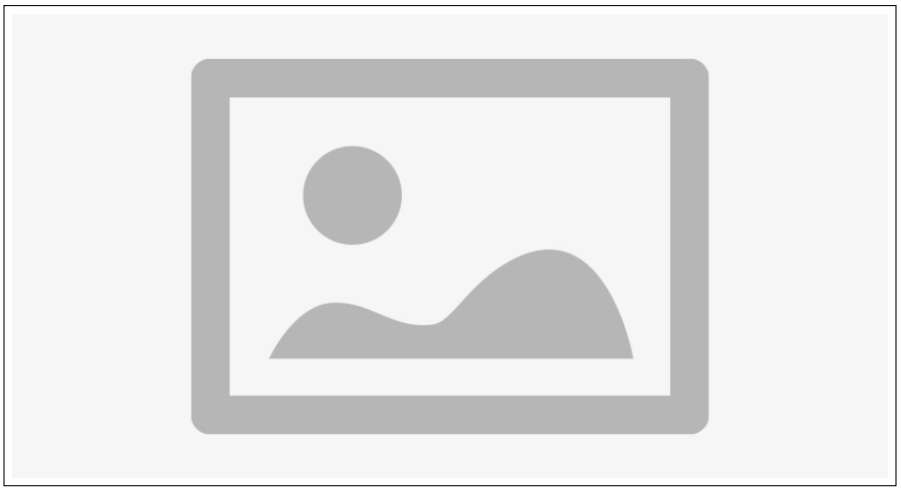

Figure 2.4: Terraform Usage
[?]

repository. This method prevents working on a large block of code for a lengthy period and multiple components that may conflict with one another.

The CD in CI/CD refers to either continuous delivery or continuous deployment, which are two highly similar words. However, there is a distinction between the two. Both models reflect automation for the pipeline's advanced phases; however, they are separated to demonstrate the high level of automation in specific instances.

Typically, changes made to an application by developers are automatically vetted and uploaded to a repository (such as GitHub or a container registry), where the operations team will deploy them to an active production environment. The continuous delivery approach improves communication between the production and business teams. As a result, its primary aim is to make new code as simple as possible to implement.

On the other hand, Continuous deployment refers to the automatic movement of developer changes from the repository to the production area, where customers may use them. This strategy frees up overburdened operations personnel from manual tasks that stymie application delivery. It is based on continuous delivery and automates the next pipeline stage.

Conclusion

We offered the ideas and terminology necessary for understanding the context of our project in this chapter. In the next chapter, we will go over how we implemented the

CI/CD pipeline for our project and how we overcame the hurdles.

3

product-x Infrastructure

Introduction

This chapter will begin by discussing the infrastructure design and the reasoning behind the different technical choices we had to make. Later, we will use the Infrastructure as Code approach to implementing it.

1 Infrastructure design

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

The following figure3.1 explains the communications between the different components:

Figure 3.1: product diagram

The following subsections will go through the service deployment, networking architecture, security, and disaster recovery strategy that we have in place.

1.1 Service deployment

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.2 Database

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder

paragraph, placeholder paragraph, placeholder paragraph,

1.3 Networking

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.4 Firewalls

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.5 Disaster recovery

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.6 Final design

You can see in the following figure3.2 a simplified version of our final infrastructure design. It important to know that we're not representing only the main components as the full diagram is much complex and contains a lot of boilerplate resources.

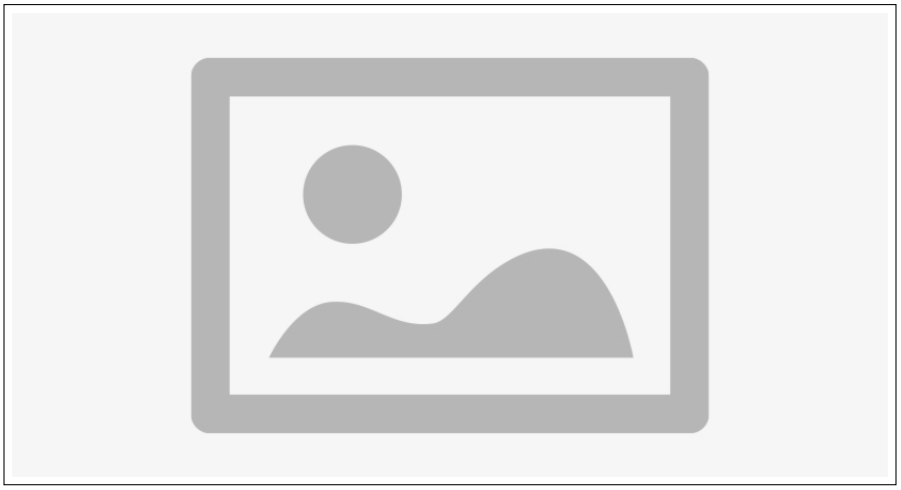

Figure 3.2: pro diagram

2 Infrastructure deployment

This section will show the actual implementation of the infrastructure using Terraform and the final result.

2.1 Using Terraform

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.2 Deployment

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

Conclusion

In this chapter, we provided a high-level overview of the product infrastructure design and the reasoning behind some of our architectural decisions. We also demonstrated how we used Terraform to build the infrastructure. In the next chapter, we will show the measures we have for quality assurance.

4

Quality Assurance

Introduction

Monitoring and alerting solutions enable IT professionals to read and analyze infrastructure data and logs to obtain insight into the performance of applications and systems. In the case of incidents, they enable system administrators to remedy the situation and avert additional problems immediately. They are also valuable for analyzing application performance and, in some situations, reducing costs.

This chapter will describe some of the monitoring services we experimented with and the end outcome. We will also demonstrate the Load-Testing project we used to assess the product performance for various setups.

1 Monitoring and Alerts

This section will begin by defining what we expect from the Monitoring and Alert system. Then we'll go through the differences between the two monitoring services we tried out and why we chose one of them.

1.1 Requirements

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.2 Tool-a

1.3 Tool-b

1.4 Comparison

The following table shows some of the differences between Tool-a and Tool-b:

2 Dashboard and alerts deployment

This section will present how the dashboard is being developed, maintained and deployed.

2.1 Infrastructure design

2.2 Usage of Terraform

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.3 Results

The following screenshot4.1 shows the end result:

Figure 4.1: Monitoring Dashboard

3 Load Testing

As it is vital to ensure high performance in our production environment, we decided to create a Load-testing project that can help us decide which would be the optimal configuration for our infrastructure. This section will list the load-test requirements, the testing framework, the load-testing infrastructure, and the final report.

3.1 Requirements

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

3.2 Tool-a

Figure 4.2: Tool Logo

3.3 Infrastructure for Load Testing

3.4 Load Testing execution

A normal execution of a test would be as follow:

1. aaa
2. aaa
3. aaa
4. aaa
5. aaaa
6. aaaa
7.

3.5 Reports and Insights

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

Conclusion

In this chapter, we had the opportunity to go through the different measures we have in place to ensure that our system is running smoothly and error-free. The next chapter will cover how we managed to automate the deployment of both the product and its monitoring & alerts system.

5

Continuous Integration and Continuous Delivery

Introduction

Now that we have the product and its monitoring dashboard's infrastructure, we need a mechanism to automate their deployment across numerous accounts. Performing the deployment manually can be time-consuming and prone to human mistakes. This chapter will present some AWS services that will enable us to manipulate resources across accounts. Then, we'll go through the design of our CI/CD pipeline and how it's deployed and maintained.

1 CI/CD Pipeline

1.1 Continuous Integration & Delivery

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

1.2 Requirements

Since the product is developed and entirely maintained by entity-y, our pipeline will only be responsible for releasing new versions of the product as well as deploying infrastructure changes.

The followings are the functional and non-functional requirements of our pipeline.

Functional Requirements:

- **Pulling resources:** The pipeline should have access to pull required artifacts and code source.
- **Automated Build:** The pipeline must be able to automatically build required artifacts.
- **Automated Release:** The pipeline must be able to automatically release artifacts to artifact repository.
- **Automated Deployment:** The pipeline must be able to automatically deploy the product and it's underlying infrastructure.
- **Automated E2E test:** The pipeline must be able to automatically perform end-to-end test on the latest deployed version to confirm that the product and the infrastructure are behaving as expected.

None-Functional Requirements:

- **Pipeline as code:** The pipeline should have a versioned life-cycle helping us to ensure better maintainability and easy replication of the pipeline.

- **Idempotence:** The pipeline should maintain the same predictable behavior when deploying changes.
- **Performance:** The pipeline should be able to deploy changes within a reasonable amount of time.
- **Reliability and maintainability:** The pipeline should be likely to work correctly. In case of failure, the pipeline should be easily repaired.
- **Security:** The pipeline should be protected against unwanted access. It should also protect secrets and password used during the build and deployment phases.
- **Usability:** A reasonably familiar person should be able to initiate and debug the pipeline.

1.3 Accounts setup

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

2 Pipeline design

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.1 Stages

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.2 Sourcing stage

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.3 Build Stage

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.4 Infrastructure Deployment

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.5 Dashboard Deployment

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

2.6 E2E testing

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

3 Pipeline deployment

3.1 Project structure

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

3.2 Results

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph,

After its first execution, the pipeline takes around 12 minutes to finish all its steps.

The following table 5.1 breaks downs the average execution time by stage calculated over the last six executions:

Steps	Average duration
Build	2 min 21 seconds
Infrastructure Plan	1 min 45 seconds
Infrastructure Apply	2 min 30 seconds
Dashboard Plan	1 min 53 seconds
Dashboard Apply	1 min 50 seconds
E2E testing	2 min
TOTAL	**12 min 19 seconds**

Table 5.1: Pipeline execution time.

As for now, the pipeline is fast enough to cover our needs, and it's fully automated and doesn't require any manual intervention (except for manual approval if needed). It also inherits the security provided by the IAM service making it only accessible on write to people granted access to the tooling account. We can also make custom roles to allow read-only access to the pipeline when needed.

Conclusion

In this chapter, we described the different steps of our CI/CD pipeline and the technical obstacles we encountered. We've also provided a high-level overview of deploying our pipelines across different Terraform projects. The final findings demonstrate that we satisfied our earlier requirements for our pipelines.

Conclusion and perspectives

placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph, placeholder paragraph.

Bibliography

[1] Company official website ©2021. Description of the company. [Access on 03/03/2021].

Placeholder PDF