Whitepaper

Stiply

STIPLY
PART OF 4CEE

Revision history			
Version	**Date**	**Author(s)**	**Description**
1.5	22-12-2025	J.W. ter Steege	Minor Lay out and typo corrections
1.4	11-11-2025	Jeffrey Bosch	Removed cloud parts except backup policy
1.3	07-07-2025	Jeffrey Bosch	Updated infrastructure pictures, development processes and updated all chapters around cloud, platform and software development (4, 6,8,9-12)
1.2	28-12-2023	Jeffrey Bosch	Updated retention period in chapter 8.4
1.1	08-12-2022	Jeffrey Bosch	Made a new version
0.7	07-11-2022	Jeffrey Bosch	Updated architecture drawing and deponent process (5.1.1 & 5.1.2)
0.6	08-03-2022	Jeffrey Bosch	Updated chapter 10.4
0.5	04-02-2022	Jeffrey Bosch	Updated backup scenario, small typography changes.
0.4	05-11-2021	Jeffrey Bosch	Updated secure software development part with new audit steps
0.4	15-09-2021	Jeffrey Bosch	Appleid 4cee template
0.3	17-08-2021	Jeffrey Bosch	The following changes: - Added chapter 5,6 - Updated chapter 7
0.2	10-08-2021	Jeffrey Bosch	The following changes: - Added Chapter 1,2,3,4 - Updated cloud architecture picture
0.1	07-07-2020	Bart Timmersman	Add cloud description

Contents

1 Introduction

1.1 Overview of the Organization

Our organization is a leader in Digital Signing solutions, known for its commitment to innovation, security, and quality. We adhere to ISO 9001, 27001, and 27017 standards, ensuring the highest levels of quality management and information security.

1.2 Purpose of the document

This document aims to provide a comprehensive overview of our cloud environment, software development practices, security measures. It is intended to inform external stakeholders about our capabilities and commitment to excellence.

1.3 Scope of the Document

The scope of this document includes detailed descriptions of our cloud infrastructure, software development lifecycle, security certifications, disaster recovery procedures.

1.4 Cloud Team

The cloud team is composed of professionals with expertise in cloud architecture, security, and administration. They manage and optimize the cloud environment to maintain high availability, scalability, and security. Stiply is part of 4CEE, an ISO certified company.

1.5 Development Team

Our development team comprises software engineers, they follow agile methodologies and best practices to deliver high-quality software solutions that meet our client's needs.

2 Basic Principles

2.1 Continuous Delivery

Continuous delivery is a key principle in our software development lifecycle. It involves the automated deployment of code changes to production environments, ensuring that new features and updates are delivered quickly and reliably.

2.2 Scalability

Scalability is essential for handling varying workloads and ensuring optimal performance. Our cloud infrastructure is designed to scale horizontally and vertically, allowing us to manage increased demand without compromising performance.

2.3 Security

Security is a top priority in our organization. We implement robust security measures, including encryption, access controls, and threat detection, to protect our clients' data and ensure compliance with industry standards.

3 Software Development and Architecture

3.1 Overview of the Software Development Lifecycle (SDLC)

Our software development lifecycle follows industry best practices and agile methodologies. It includes planning, design, development, testing, deployment, and maintenance phases to ensure the delivery of high-quality software solutions.

3.2 Agile Methodologies and Practices

We adopt agile methodologies, to enhance collaboration, flexibility, and iterative development. These practices enable us to respond quickly to changing requirements and deliver value to our clients.

3.3 Tools and Technologies Used

We utilize a wide range of tools and technologies to support our software development process. These include integrated development environments (IDEs), version control systems, continuous integration tools, and automated testing frameworks.

3.4 Quality Assurance and Testing Processes

Our quality assurance and testing processes are designed to ensure the delivery of robust and reliable software solutions. We employ various testing techniques, including unit testing, integration testing, system testing, and user acceptance testing.

3.5 Continuous Integration and Continuous Deployment (CI/CD)

Continuous integration and deployment are integral to our software development process. We use CI/CD pipelines to automate the build, test, and deployment stages, ensuring faster and more reliable software releases.

3.6 Concepts

We follow key software development concepts, including modular design, code reusability, and maintainability. These concepts help us build scalable and maintainable software solutions.

3.7 Multi-Tenant

Our software architecture supports multi-tenancy, allowing multiple clients to share the same application instance while maintaining data isolation and security. This approach ensures efficient resource utilization and cost savings.

3.8 Technology Stack

We leverage a diverse technology stack, including programming languages, frameworks, and databases, to develop robust and scalable software solutions. Our technology stack includes:

- Stiply app: React / JQuery, PHP Laravel and Angular
- Admin panel: React, PHP Laravel
- Carerix: React, PHP Laravel Lumen
- Word Add: Angular, Stiply app (no backend)
- Manager portal: Angular, PHP Laravel
- Nmbrs Connector: .NET Core

3.9 Software Components

Our software solutions are composed of various components, including frontend, backend, databases, and third-party integrations. These components work together to deliver a seamless and integrated user experience.

3.10 Interaction Between Software Components

The interaction between software components is facilitated through APIs, messaging queues, and event-driven architectures. This approach ensures efficient communication and data exchange between different parts of the system.

4 Secure Software Development

4.1 Software Development Process

Our software development process adheres to secure coding practices and involves security assessments at various stages. We perform code reviews and security testing to identify and mitigate potential vulnerabilities. The objective of our build pipeline is to convert code into executable software that is ready for controlled production deployment. Risks are mitigated by employing state-of-the-art components to establish quality gates.

Regardless of which part of our software stack is modified, all code is committed to our secured GitLab environment where a peer review of the code is conducted. Additionally, code quality is monitored using the static analysis engine of SonarQube. Functional correctness is tested with several types of automated tests. Only when all tests are passed can the code be deployed to our beta environments. Upon acceptance, we merge all changes to our main branch, from which we deploy our software to the production environment.

To manage software vulnerabilities, we utilize Dependency Track, which provides notifications about vulnerable packages. This enables us to check daily or hourly for vulnerable packages. We then report the audit findings in our team's channel and create an issue if a package update is required due to a vulnerability.

We have automated this process with a Git tag creation. The pipeline will automatically deploy to acceptance and subsequently requires manual approval for production deployment. The deployment step itself is always manually approved by a developer, with a cloud engineer on standby if needed. A cloud engineer's presence is determined by the impact of the code; for example, a database change necessitates the involvement of a cloud engineer.

Furthermore, we leverage GitOps principles to manage our infrastructure and applications. This approach allows us to use Git as the single source of truth for declarative infrastructure and application definitions. Changes are tracked and versioned in Git, and the deployment process is triggered by changes to the repository, ensuring a consistent and auditable workflow.

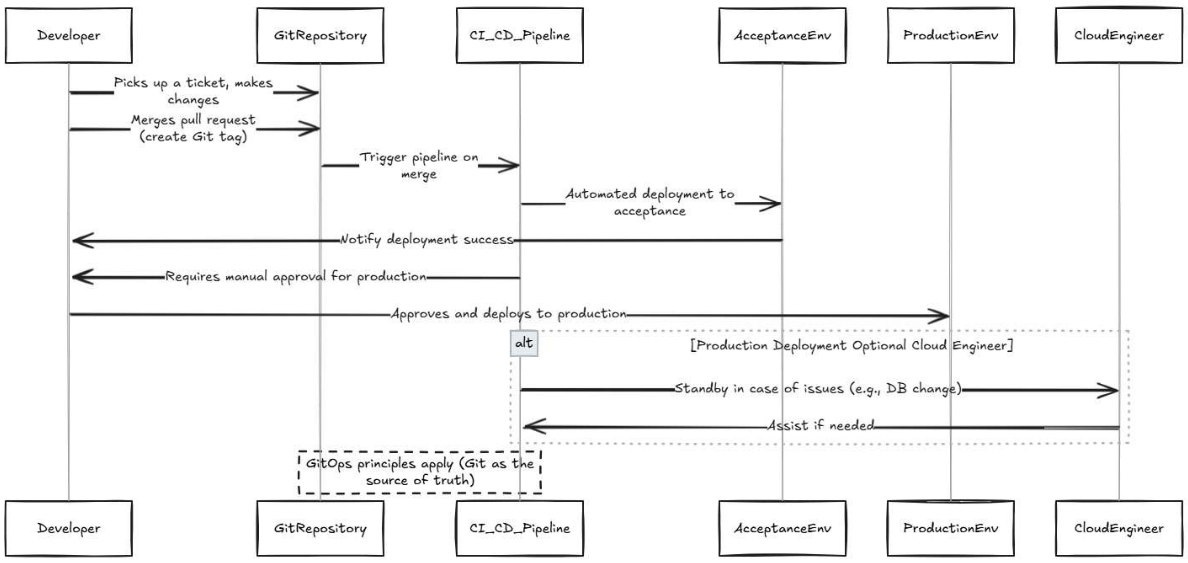

Developer
GitRepository
CI_CD_Pipeline
AcceptanceEnv
ProductionEnv
CloudEngineer
Picks up a ticket, makes changes
Merges pull request (create Git tag)
Trigger pipeline on merge
Automated deployment to acceptance
Notify deployment success
Requires manual approval for production
Approves and deploys to production
alt
[Production Deployment Optional Cloud Engineer]
Standby in case of issues (e.g., DB change)
Assist if needed
GitOps principles apply (Git as the source of truth)

Figure 1, process of development to production (see attachment of this picture in bigger format)

4.2 Software Development Tools

We use a range of tools to support secure software development, including static and dynamic analysis tools, dependency scanners, and security testing frameworks. These tools help us identify and remediate security issues early in the development process.

4.3 Software development tools

Tool	Description
.NET Core	NET Core is a free and open-source, managed computer software framework for Windows, Linux, and macOS operating systems.
Angular	Angular is a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations.
Cypress	Cypress is an open-source front-end testing tool.
GitLab	GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking, and CI/CD pipeline features, using an open-source license.
SonarQube	SonarQube is an open-source platform developed for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.
PHP Stan	An open-source tool that's focuses on findings errors based on static analyses
React	React is an open-source library that focuses on building user interfaces, it is maintained by Facebook and the community. It can work with Typescript or JavaScript
NX	NX is a smart and extensible tool that helps you scale your applications or libraries, it supports react, angular, and other typescript languages.
NPM	Node package manager, our node / javascript package manager
Dependency Track	A vulnerability scanner.
Flux	Flux is a GitOps tool that automates syncing Kubernetes cluster state with configuration stored in Git. It continuously monitors Git repositories and applies changes to ensure the cluster matches the declared desired state.

5 Security Certifications

5.1 ISO 9001: Quality Management Systems

ISO 9001 certification ensures that our quality management systems meet the highest standards. It covers various aspects of our operations, including customer satisfaction, process improvement, and regulatory compliance.

5.2 ISO 27001: Information Security Management Systems

ISO 27001 certification demonstrates our commitment to information security. It ensures that we have implemented effective security controls and risk management practices to protect sensitive data and maintain confidentiality, integrity, and availability.

5.3 ISO 27017: Cloud Security

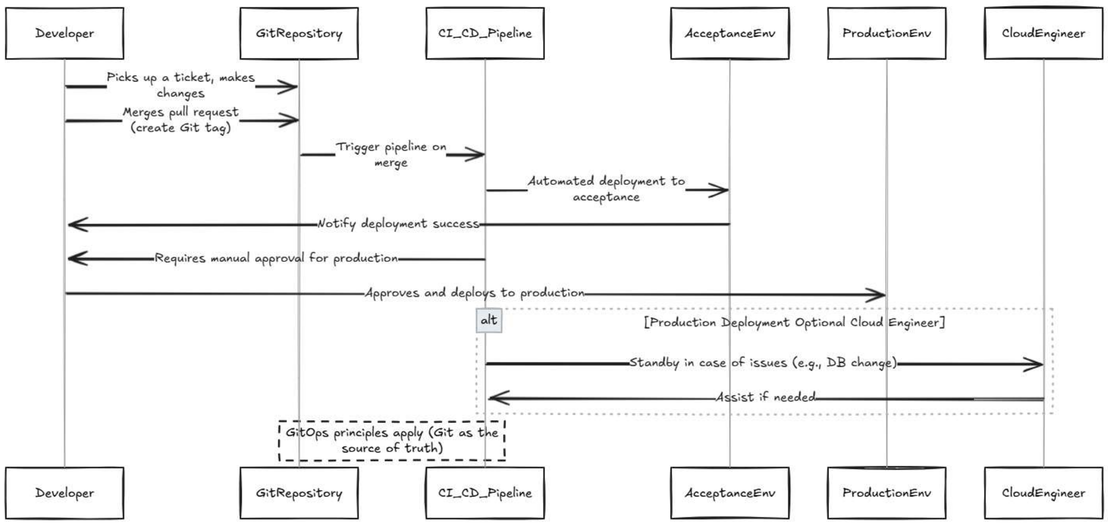
ISO 27017 certification focuses on cloud security, providing guidelines for protecting cloud environments and data. It ensures that we follow best practices for managing cloud security risks and implementing appropriate security controls.

6 Secure Software Deployment

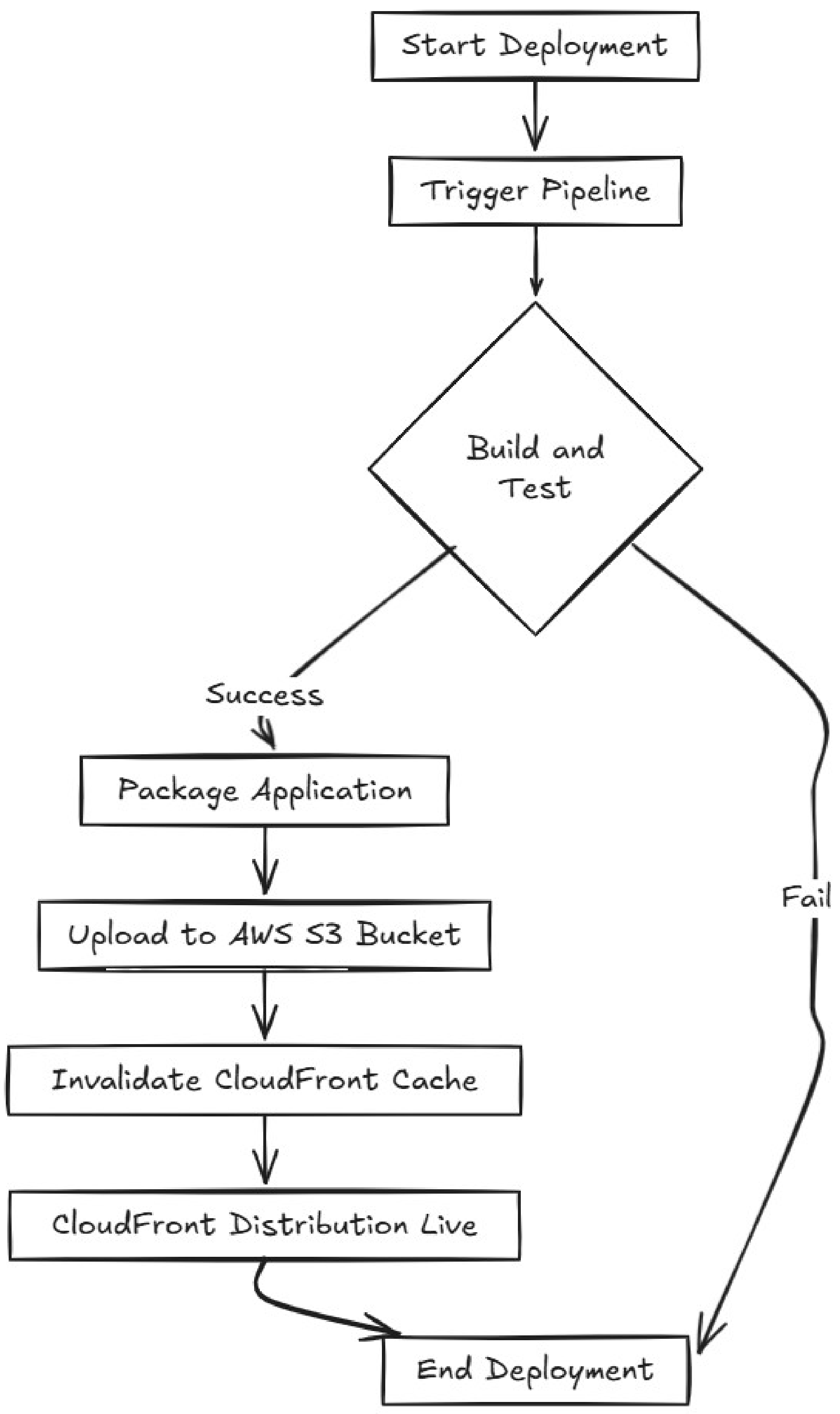
6.1 Software Deployment Process

Our software deployment process includes various security measures, such as environment isolation, automated deployment pipelines, and security testing. We use infrastructure-as-code (IaC) tools to automate and secure the deployment process.

6.1.1 Laravel applications


Developer
GitRepository
CI_CD_Pipeline
AcceptanceEnv
ProductionEnv
CloudEngineer
Picks up a ticket, makes changes
Merges pull request (create Git tag)
Trigger pipeline on merge
Automated deployment to acceptance
Notify deployment success
Requires manual approval for production
Approves and deploys to production
alt
[Production Deployment Optional Cloud Engineer]
Standby in case of issues (e.g., DB change)
Assist if needed
GitOps principles apply (Git as the source of truth)

6.1.2 Single page applications



7 Testing Software

7.1 Theory Behind Our Vision on Correct Software Testing

Our vision for software testing is based on the principles of early detection, continuous feedback, and comprehensive coverage. We aim to identify and address issues as early as possible in the development process.

7.2 Shift Left Testing vs Shift Right Testing

We employ both shift left and shift right testing approaches. Shift left testing focuses on early testing during the development phase, while shift right testing involves testing in production environments to ensure ongoing quality and reliability.

7.3 Agile Testing Quadrants

We use agile testing quadrants to categorize different types of testing and ensure comprehensive coverage. These quadrants include unit testing, integration testing, system testing, and user acceptance testing.

7.4 Applied Test Types

We apply various test types to ensure the quality and reliability of our software solutions. These include functional testing, performance testing, security testing, and usability testing.

Test type	Quadrant	Development phase	Description	Test type
Unit test	Q1	Development	We aim to cover as mthomasany units of code as possible to automatically detect errors as soon as possible.	Unit test
Prototyping	Q2	Development	One of the first things we create are prototypes of the software to be developed. These are used to elicit feedback from customers regarding functionality and usability. Usually, multiple refinements are done before the actual build starts.	Prototyping
Integration test	Q3	Acceptance	For every release of our software, we execute a fully automated regression test of multiple flows in our software. User actions are	Integration test

			simulated in the user interface in a production-like environment.	
Pentest	Q4	Production	We use checklists to check and improve specific security attributes of our software and infrastructure. Ad hoc independent penetration tests, pentests for short, are executed.	Pentest
Monitoring	Q4	Production	A multitude of monitoring tools is in use in our production environment. More	Monitoring

7.5 Test Follow-Up

Our test follow-up process includes tracking and resolving identified issues, conducting root cause analysis, and continuously improving our testing practices. We use test management tools to monitor test progress and report results.

8 Platform Architecture

At 4CEE, we maintain a comprehensive Cloud Whitepaper that covers all general aspects of our platform, including access management and other cloud-related topics. This section focuses on features that are unique to the Stiply platform, rather than general practices applied across all 4CEE services.

8.1 Backup Strategies

8.1.1 Database and Server Backups

In the Stiply environment, we automatically create backups of our databases and server snapshots every 12 hours. These backups are kept for one month, after which they are securely deleted. To ensure the safety of your information, all backups are encrypted using AES-256 encryption. Access to these files is tightly controlled and can only be gained with an encryption key, which is securely stored in the AWS administrator account.

8.1.2 File Backups

Our cloud services are designed to be flexible and secure, meeting a variety of client needs. For example, files such as images and documents (blobs) are stored in each customer's dedicated Amazon S3 bucket. To add an extra layer of protection, we keep a copy of this data in a backup region — specifically, Paris, France. The system automatically replicates files to this backup location, which is managed under a separate AWS account for enhanced security. Once the replication is complete, the backup bucket’s owner assumes ownership of the data to further safeguard it

9 AUDITS INITIATED BY CUSTOMERS

9.1 ADDITIONAL INDEPENDENT AUDIT

Customers have the right to request an additional independent audit. This audit can be carried out on the technical and organizational security measures that Stiply takes to protect the information of it and its customers. The following conditions are attached to this audit:

- The audit must be planned and approved by Stiply;
- All risks involving this audit must be evaluated and mitigated;
- Stiply will inform involved parties like AWS;
- The audit takes place against the ISO9001 & ISO27001 & ISO27017 standard;
- The audit is performed by an accredited auditor;
- The costs for the audit are for the client;
- All confidential business information remains within Stiply;
- All information about other customers remains within Stiply.
- The additional independent audit can only be requested once a year

Stiply accepts to resolve all identified shortcomings within a reasonable period.

10 Conclusion

In conclusion, this document has provided a comprehensive overview of our cloud environment and its various components. We are dedicated to maintaining the highest standards of quality, security, and innovation in our cloud services.

10.1 Summary of Key Points

This section summarizes the key points covered in the document, highlighting our commitment to excellence and continuous improvement.

10.2 Call to Action

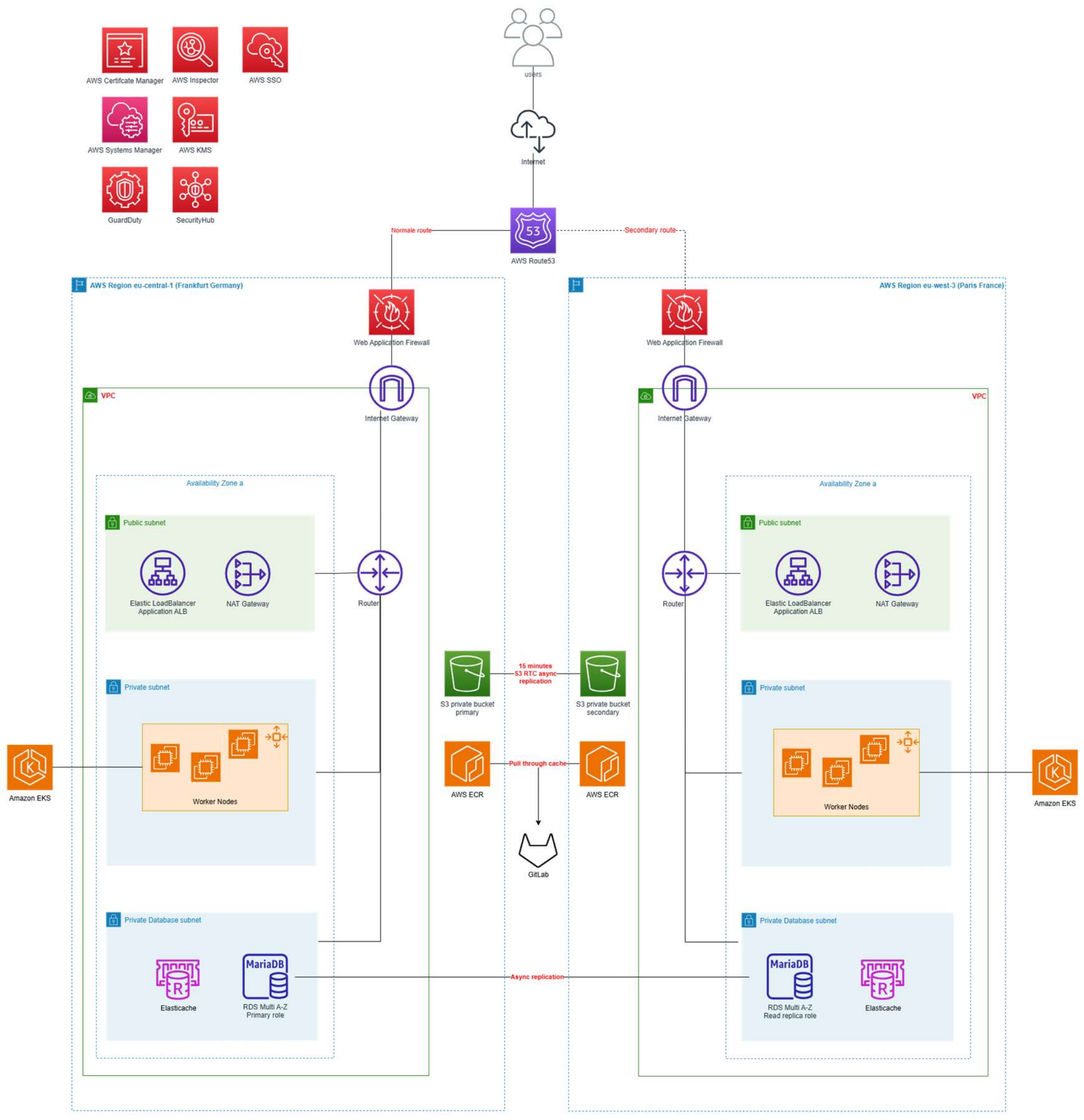
We encourage our stakeholders to engage with us and explore the full potential of our cloud services. Together, we can achieve great things and drive the future of cloud computing.

10.3 Contact Information

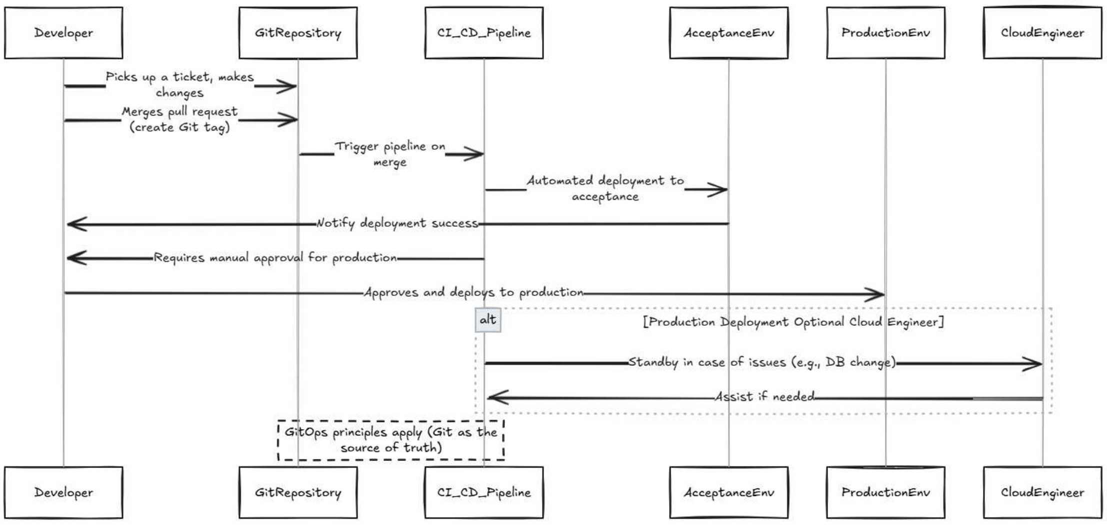
For more information or to discuss how our cloud services can benefit your organization, please contact us. We look forward to hearing from you and collaborating on innovative solutions

11 Attachments

11.1 Cloud environment



11.2 Development process


Developer
GitRepository
CI_CD_Pipeline
AcceptanceEnv
ProductionEnv
CloudEngineer
Picks up a ticket, makes changes
Merges pull request (create Git tag)
Trigger pipeline on merge
Automated deployment to acceptance
Notify deployment success
Requires manual approval for production
Approves and deploys to production
alt
[Production Deployment Optional Cloud Engineer]
Standby in case of issues (e.g., DB change)
Assist if needed
GitOps principles apply (Git as the source of truth)