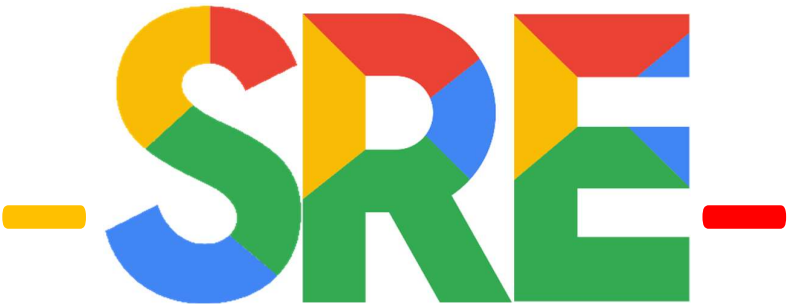

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Course: Software Requirements Engineering

Instructor: Lecturer Sara Rehmat

Submitted by:
Muhammad Rehan | 22P-9106
Hasnain Saleem | 22P-9123
Class: BSE-3A

Date: Oct 15th, 2023.

Assignment no 02

Project: Cloud Native Development Service

Vision and Scope Document

1. Business Requirements

1.1 Background

The move to cloud-native solutions happened because of a few main reasons. First, old IT systems were inflexible and costly, so companies sought cheaper and more flexible options. New technologies like containers and Kubernetes made building and managing software easier. Companies also wanted to work faster to keep up in a competitive world and needed reliable and secure systems. Many had old computer systems causing problems, so they switched to cloud-native ways to modernize. A bunch of tools and help from the community made this shift more accessible for businesses aiming to do well in the digital world.

1.2 Business Opportunity

In the ever-evolving digital landscape, businesses seek to enhance their corporate information systems' scalability, efficiency, and agility. Traditional IT infrastructures struggle to meet these demands cost-effectively. Cloud-native solutions, aligning with market trends favoring **agility** and **cost-efficiency**, offer seamless **scalability**, **cost savings**, **rapid software deployment**, and improved **system resilience**. Critical components for a complete cloud-native solution include **microservices architecture**, **CI/CD pipelines, container orchestration** (e.g., **Kubernetes**), **robust security**, **DevOps practices**, **scalable cloud infrastructure**, and skill development programs. These elements cater to customer needs such as scalability, cost-efficiency, agility, security, **competitiveness**, **modernization**, **compliance**, and **skill development**. The new cloud-native product effectively addresses these needs, facilitating seamless scalability, cost efficiency (e.g., server cost reduction during off-peak hours), rapid **software updates**, and other critical enhancements for a competitive edge.

1.3 Business Objectives

BO-1: Aim to accelerate development speed by around **50%** during the initial development phase. Once the project transitions to the maintenance phase, maintain this increased pace to handle ongoing updates and support efficiently.

BO-2: Achieve a **2x** reduction in operational costs by paying for resources based on your application's actual usage, optimizing expenditure while maintaining performance.

BO-3: This goal aims to improve system robustness and capacity, aiming to achieve a **60%** enhancement within the initial year.

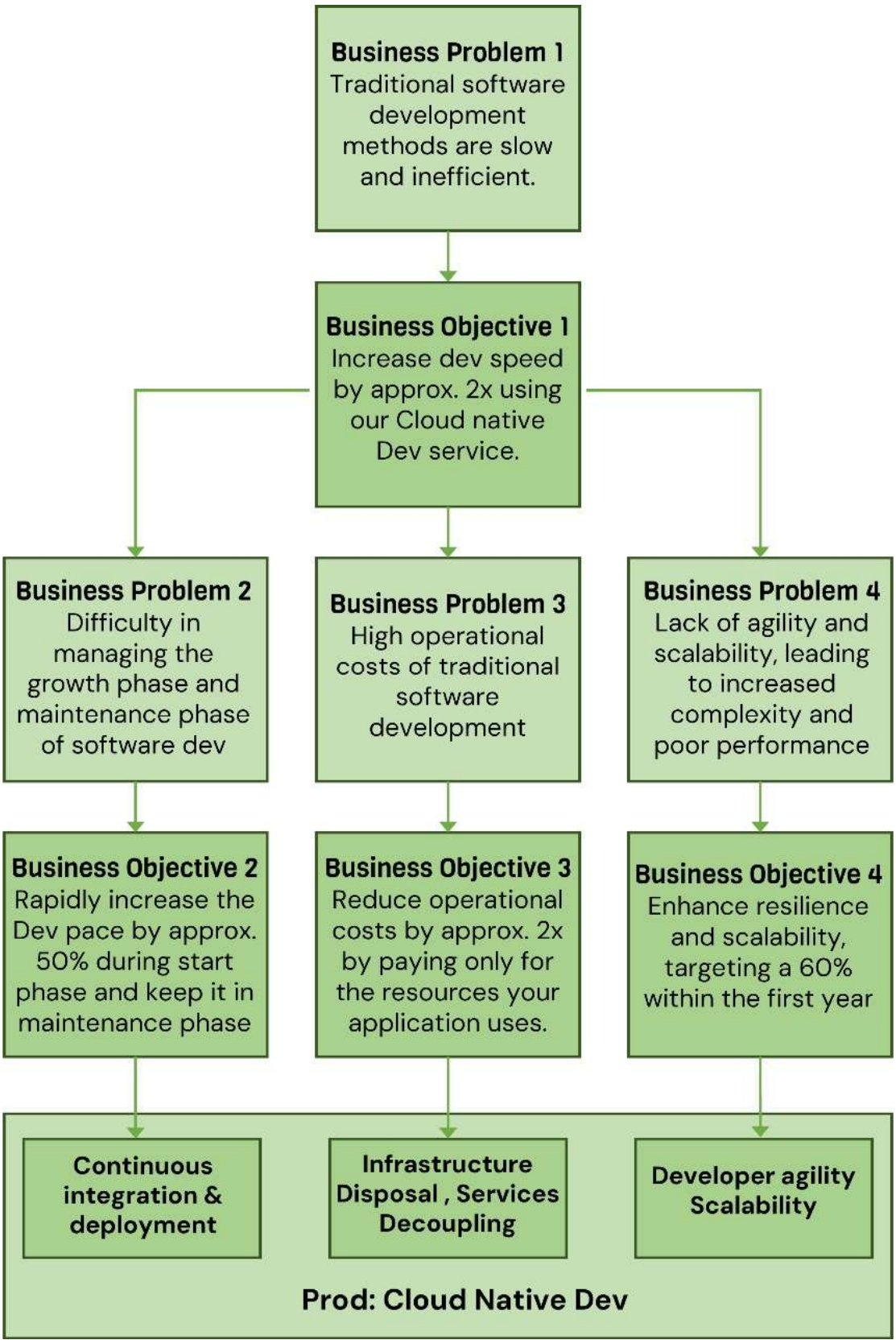

Figure 1: Business Objectives Model

1.4 Vision Statement

We offer a cloud-native development service **for** businesses and organizations seeking an edge in the competitive and rapidly evolving world of software development. This service is designed to provide a comprehensive understanding of the requirements and principles driving the shift toward cloud-based solutions. **Unlike** traditional software development methods, our approach is focused on scalability, enhanced customer experience, developer agility, continuous integration and deployment (CI/CD), disposable infrastructure, and decoupling of services.

By leveraging the expertise of the Cloud Native Computing Foundation (CNCF), we can deliver software development at approximately **twice (2x) the speed** of traditional methods. Our service thoroughly analyzes the tools and technologies used in cloud-native development, such as Kubernetes, Microservices, Containers, and Terraform. These tools help rapidly increase the development pace **~=50%** in the start phase of development, which continues even in the maintenance phase of the project.

Unlike traditional development methods, **our project** allows businesses to utilize cloud-native technologies' benefits fully. It provides a practical application of these technologies, showcasing the benefits of embracing this paradigm shift diversity.

Compared to the traditional approach, our cloud-native service ensures your applications' availability, resilience, and scalability. It also allows for **twice (2x) cost-effective** operations as you only pay for the resources your application uses. It makes it a more efficient and economical choice for businesses.

With **our project**, businesses will gain a deep understanding of the reasons behind the shift to cloud-native development. It will **enable them** to create innovative, scalable, and reliable software solutions that meet the demands of the modern digital landscape. By choosing **our service**, businesses can ensure they stay ahead of **10x times** the curve and are equipped to meet future challenges in the software development landscape.

1.5 Business Risks

RI-1: This risk pertains to the potential entry of new competitors or intensified competition in the market, which could lead to a loss of market share and reduced profitability. (Probability = 0.4; Impact = 4)

RI-2: This risk involves the possibility of project delays, resulting in missed market opportunities and potential revenue losses.
(Probability = 0.3; Impact = 3)

RI-3: This risk relates to the product not meeting user expectations, leading to decreased sales and potential damage to the brand's reputation. (Probability = 0.2; Impact = 4)

RI-4: This risk concerns the potential occurrence of a data security breach, resulting in data loss, legal consequences, and damage to the company's reputation. (Probability = 0.5; Impact = 5)

RI-5: This risk involves failing to comply with relevant regulations and facing legal penalties, fines, and reputational harm. (Probability = 0.3; Impact = 4)

2. Scope and Limitations

2.1 Major Features

FE-1: Encapsulate applications and their dependencies in portable units, ensuring consistent performance and easy deployment across various environments, enhancing application reliability and scalability.

FE-2: Streamlines repetitive tasks and processes, making software development and operations more efficient and error-free, resulting in faster and more reliable services.

FE-3: Breaks down complex applications into more minor, independently deployable services, enabling quicker development, easier maintenance, and more flexible scalability.

FE-4: Provides standardized interfaces for software components to interact, allowing seamless integration between different services, applications, and systems.

FE-5: Fosters collaboration between development and operations teams, automating software delivery pipelines for faster feature releases and improved product quality.

FE-6: Efficiently manages and scales containers and services, ensuring applications run smoothly in dynamic, cloud-native environments, enhancing performance and resource utilization.

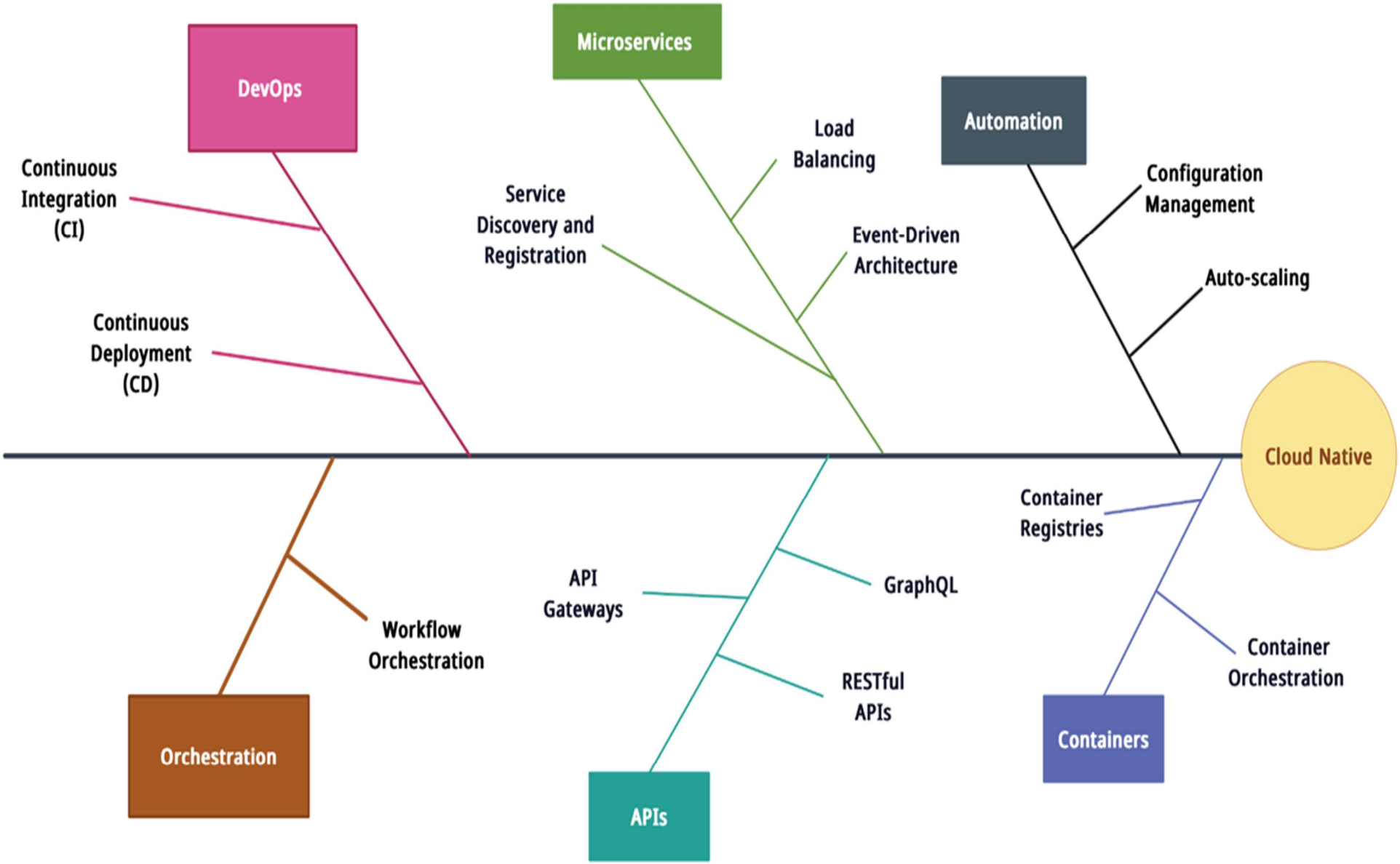

Figure 2 : Feature tree for Cloud Native

2.2 Scope of initial and subsequent releases

Feature	Release 1	Release 2	Release 3
FE-1, Scalability	Basic horizontal scaling using load balancers	Advanced horizontal scaling with auto-scaling groups	Full implementation of horizontal and vertical scaling with auto-scaling, load balancing, and caching
FE-2, Enhanced Customer Experience	Basic user interface improvements	Advanced user interface enhancements with responsive design and accessibility features	Full implementation of user interface improvements with optimized performance and personalized experiences
FE-3, Developer Agility	Essential integration with version control systems	Advanced integration with CI/CD pipelines and automated testing	Full implementation of developer agility with continuous deployment, feature flagging, and automatic rollback mechanisms
FE-4, CI/CD	Basic setup of CI/CD pipelines for build and deployment	Advanced CI/CD pipelines with automated testing and release management	Full implementation of CI/CD with continuous integration, automated testing, deployment pipelines, and release orchestration
FE-5, Disposable Infrastructure	Not Implemented	Basic implementation of infrastructure as code using Terraform	Full implementation of disposable infrastructure with automated provisioning, scaling, and teardown using Terraform and Kubernetes

FE-6, Decoupling of Services	Essential service decoupling using API gateways	Advanced service decoupling with event-driven architecture and microservices	Full implementation of service decoupling with microservices, event-driven communication, and service mesh
FE-7, Cloud Native Technologies (Kubernetes, Microservices, Containers, Terraform)	Basic containerization using Docker	Advanced container orchestration with Kubernetes	Full implementation of cloud-native technologies with containerization, orchestration, and infrastructure as code using Kubernetes, Docker, and Terraform

2.3 Limitations

LI-1: The reliance on a stable internet connection can be a critical limitation, as it affects the accessibility and usability of cloud-native applications. Users may experience disruptions or loss of functionality without a reliable internet connection.

LI-2: Users place a high value on the privacy and security of their data. Any perceived or actual data privacy breaches can have serious consequences, making this a top concern.

LI-3: Users expect cloud services to be highly reliable. Downtime or service outages can significantly impact their ability to access and use cloud-native applications, making service reliability a top priority.

LI-4: Security is paramount for users, as they want assurance that their data and transactions are secure. Any perceived or actual security vulnerabilities can erode trust and confidence in cloud-native solutions.

LI-5: Users are concerned about the financial aspect of cloud-native solutions. The uncertainty of variable costs and potential budgetary concerns can be a significant limitation, especially for businesses and organizations.

3. Business Context

3.1 Stakeholder Profiles

Stakeholders	Major Benefit	Likely Attitudes	Major Interests	Known Constraints
CNCF Member Organizations	Industry collaboration and innovation; aims to participate in collaborative projects and stay at the forefront of cloud-native development. Access to CNCF projects and resources; expect to leverage CNCF's cloud-native technologies for their own infrastructure and applications.	Positive and collaborative, focusing on the opportunities for innovation and technology adoption.	Access to CNCF-hosted projects like Kubernetes, Prometheus, and Envoy. Participation in CNCF working groups, special interest groups, and events. Networking and knowledge-sharing opportunities within the CNCF community.	Active involvement in CNCF activities and collaboration.
Cloud-Native Software Developers and Engineers	Access to cutting-edge tools and technologies; expects to work with cloud-native software and infrastructure technologies. Career development and learning opportunities; aims to enhance skills and knowledge in cloud-native practices.	Enthusiastic and eager to learn and contribute to cloud-native projects.	Access to CNCF projects for development and contributions. Educational resources, tutorials, and documentation. Participation in open-source communities and events.	Staying updated with rapidly evolving technologies. Balancing contributions to CNCF projects with other work commitments.
End-Users and Enterprises Adopting Cloud-Native Technologies	Efficiency and scalability: Expects to leverage cloud-native technologies to improve operational efficiency and scalability. Community and vendor support: Aims to tap into a wide range of cloud-native tools and expertise.	Pragmatic, focusing on business benefits and technology adoption.	Vendor-neutral ecosystem with a variety of solutions. Community forums and support channels for problem-solving.	Ensuring security and compliance in cloud-native environments. Evaluating and selecting the most suitable cloud-native solutions for their specific needs.
Cloud Service Providers (e.g., AWS, Google Cloud, Azure)	Expect to offer cloud-native services and solutions that align with CNCF projects. Aim to stay at the forefront of cloud-native technologies.	Positive, focusing on providing cloud-native solutions and contributing to CNCF projects.	Integration with CNCF-hosted projects for enhanced cloud-native capabilities. Certification and compatibility with CNCF standards. Participation in CNCF working groups and events.	Ensuring compatibility and adherence to CNCF project standards. Balancing existing services with emerging cloud-native technologies.

3.2 Project Priorities

Dimensions	Constraints	Driver	Degree of freedom
Features		CNCF projects should aim to deliver a rich set of features that meet the diverse needs of the cloud-native community. While it's a driver for project success, the priority should be balanced with other dimensions to ensure that quality, schedule, cost, and staff constraints are not compromised.	
Quality		Maintaining high-quality code, security, and performance is a critical driver. CNCF projects should prioritize rigorous testing, security reviews, and adherence to best practices to ensure the quality and reliability of cloud-native technologies.	
Schedule	Meeting project deadlines is often a non-negotiable constraint. CNCF should manage schedules effectively to ensure that projects align with industry expectations and can be integrated into cloud-native environments.		
Cost			Cost considerations should be a factor but have some flexibility. CNCF should manage costs efficiently, but it's important to have some latitude for adjusting resources when necessary to support project success.
Staff			Having the right staff expertise and resources is crucial for project success. CNCF should ensure it has the flexibility to allocate and manage staff resources effectively to support its projects, especially when facing unexpected challenges.

3.3 Deployment Considerations

To ensure effective deployment of cloud-native solutions using the Cloud Native Computing Foundation (CNCF) ecosystem, several considerations are essential. CNCF projects typically serve a global and diverse user base. Access requirements can vary, with users distributed across multiple time zones, and they may need to access the system at various times. It's crucial to accommodate this distributed user base effectively. Infrastructure changes might be necessary to support the software's capacity, network access, data storage, and data migration requirements. These changes should align with CNCF's scalability, interoperability, and security principles. Additionally, comprehensive documentation is vital to assist those preparing training materials or adapting business processes in coordination with deploying new solutions, ensuring a seamless transition for users and stakeholders.

3.4 Scope representation techniques:

Context Diagram

Businesses & Organizations (Clients) <-> Cloud-Native Development Service

Clients provide the requirements for the software projects they want to develop. The service, in turn, delivers the developed software applications back to the clients.

CNCF <-> Cloud-Native Development Service

The CNCF provides principles and guidelines for cloud-native development. The service uses these guidelines to ensure the software development adheres to the best practices in cloud-native development.

Cloud-Native Tools & Technologies <-> Cloud-Native Development Service

The Cloud-native tools and technologies provide the necessary infrastructure and capabilities for the Cloud-native development service to build, deploy, and manage software applications in a cloud environment. In return, it receives Build logs, deployment logs, application logs, performance metrics, and error events that can be monitored and analyzed for insights into the development process in our service.

Applications Developed using the Service <-> Cloud-Native Dev Service

These applications provide feedback, like runtime errors and usage statistics, to the service. In return, these apps receive deployment information, configurations, monitoring management, and compliance from our service.

The Modern Digital Landscape <-> Cloud-Native Development Service

The "Modern Digital Landscape" refers to the vast collection of technologies, trends, and practices that shape the digital infrastructure and how we interact with technology. This landscape constantly evolves, giving us insights to stay up-to-date with the new standards and regulations.

Cloud Service Providers <-> Cloud-Native Development Service

These services deploy the applications on these cloud service providers. The provider gives our service Infrastructure, Platform, Services, Scalability, Security, Cost-effectiveness, and many more benefits.

Cloud-Native Software Developers <-> Cloud-Native Development Service

Developers and engineers provide inputs like code, configuration, testing, and maintenance to the service and receive feedback like build status, deployment status, Guidelines from the CNCF, and requirements from clients (as it is an indirective receiving from other external terminators that's why these are not mentioned in the context diagram.).

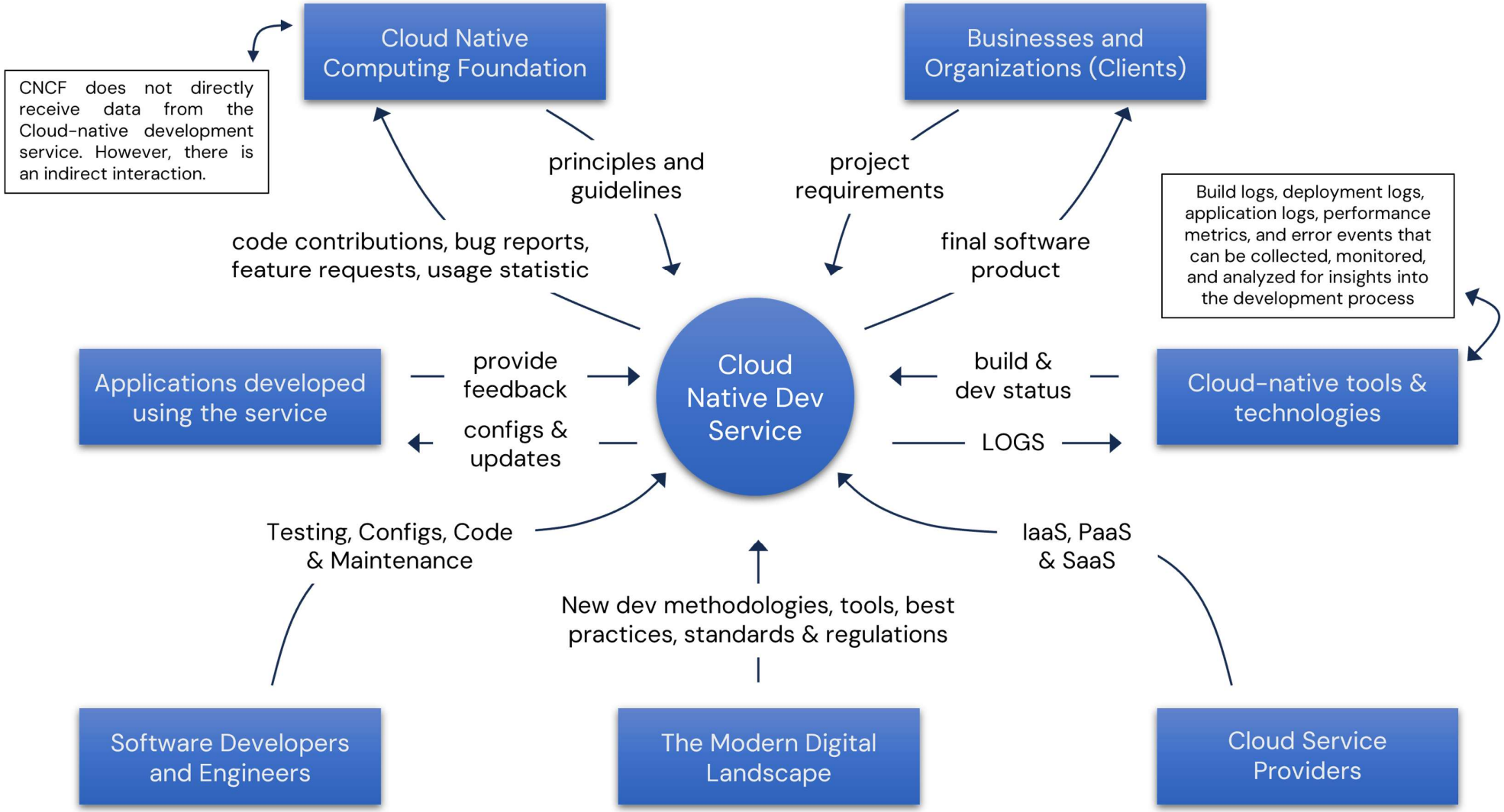

Figure 3: Context Diagram for Cloud Native

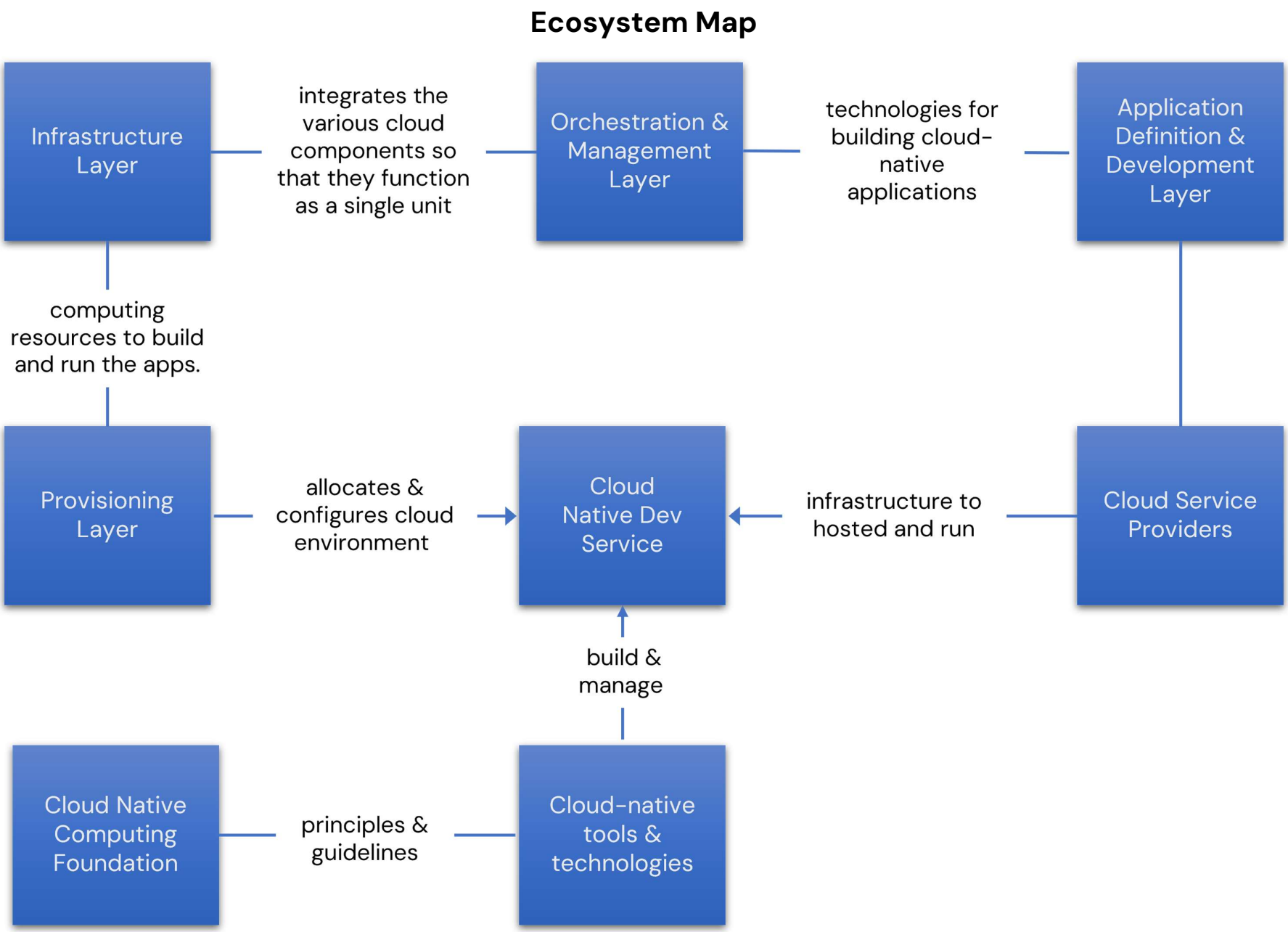

Figure 5: Ecosystem map for Cloud Native

The Infrastructure Layer, Provisioning Layer, Application Definition and Development Layer, and Cloud Service Providers work together to deliver an excellent cloud-native development service.

The Infrastructure Layer, which includes operating systems, storage, network, and other computing resources, forms the foundation of the cloud-native stack. It provides the computing resources the service uses to build and run the applications.

The Provisioning Layer, which consists of cloud services that allocate and configure the cloud environment, manages the cloud resources for the service. It ensures that the necessary resources are available when needed.

The Application Definition and Development Layer, which includes software technologies for building cloud-native applications, provides the tools and technologies the service uses to make the applications. It defines the structure and behavior of the applications.

Cloud Service Providers, such as AWS, Google Cloud, and Azure, provide the infrastructure where the applications developed by the service are hosted and run. They manage the hardware and computing resources in their own data centers, and the service interacts with these providers to deploy and manage the applications.

The CNCF and Cloud-native tools and technologies work together to provide a comprehensive platform for cloud-native development. The CNCF provides the guidelines and support, and the tools and technologies provide the capabilities for building and managing cloud-native applications.

Together, these layers and technologies work to deliver a comprehensive cloud-native development service that is scalable, flexible, and resilient, meeting the demands of the clients.

Event List

- **Client Requirements Submission:** Businesses and organizations submit their software project requirements to the Cloud-native development service.
- **Project Initiation:** The service initiates a new project based on the client's requirements.
- **CNCF Guidelines Application:** The service applies the principles and guidelines provided by the CNCF to the project.
- **Tool & Technology Utilization:** The service uses the Cloud-native tools and technologies (Kubernetes, Microservices, Containers, and Terraform) to develop the software project.

- **Build Logs & Deployment Logs Generation:** The Cloud-native tools and technologies generate build logs, deployment logs, application logs, performance metrics, and error events.
- **Application Development:** The service develops software applications based on the client's requirements and the guidelines provided by the CNCF.
- **Application Feedback:** The applications developed by the service provide feedback, like runtime errors and usage statistics, to the service.
- **Deployment Information & Configurations:** The service provides the applications with deployment information, configurations, monitoring management, and compliance.
- **Infrastructure Provisioning:** The service provides the necessary infrastructure for the Cloud Service Providers applications.
- **Application Deployment:** The service deploys the applications on the provisioned infrastructure.
- **Application Monitoring:** The service monitors the applications for performance, usage, and errors.
- **Application Maintenance:** The service performs application maintenance, including updates, patches, and upgrades.
- **Project Completion:** The service completes the project and delivers the developed software applications back to the clients.

This event list may seem unconventional, but it's crucial to understand that our project is a cloud-native development service. The nature of cloud-native development is fundamentally different from traditional software development methods.

*The events listed reflect a cloud-native development service's key actions and data exchanges. For example, the "**Tool & Technology Utilization**" event represents using cloud-native tools and technologies in the development process, a fundamental aspect of cloud-native development. Similarly, the "**Application Feedback**" event represents the continuous feedback loop in cloud-native development, where applications provide real-time data that can be used to improve the system.*