22AIE305: CLOUD COMPUTING

The top six advantages and benefits of cloud-natives solutions

Cloud native is not only a paradigm shift in how software is built; it is a cultural change that impacts the entire business

01 **Lifecycle management through DevOps**
DevOps-driven development ensures faster software delivery and better performance by enabling collaboration between teams and departments in a cycle of rapid, frequent, and reliable software delivery.

02 **Customer-centric software development**
With the single cross-functional autonomous team focused on delivering hign-quality services that run isolated through API-based integration, the business can focus on develping client features that generate revenue.

03 **Collaborative environments through contract-based interaction**
The more enterprises invest in API-based integration, the more they will improve their ability to react quickly to new challenges and deploy new solutions for their customers.

04 **Higher complexity managed through serverless platforms**
There are now over 6.5 million developers working with cloud-native technologies, four million of them using serverless architectures and cloud functions.

05 **Self-service, elastic on-cloud deployment**
At least 60% of backend developers are now using containers to deploy their solutions. By 2022, 75% of enterprises are expected to run containers in production.

06 **Enhanced architectures for improved resilience**
Whether it is a preferred base language or a diverse technological stack, enterprises should use API gateways and services meshes to guarantee security and communications throughout the microservices architecture.

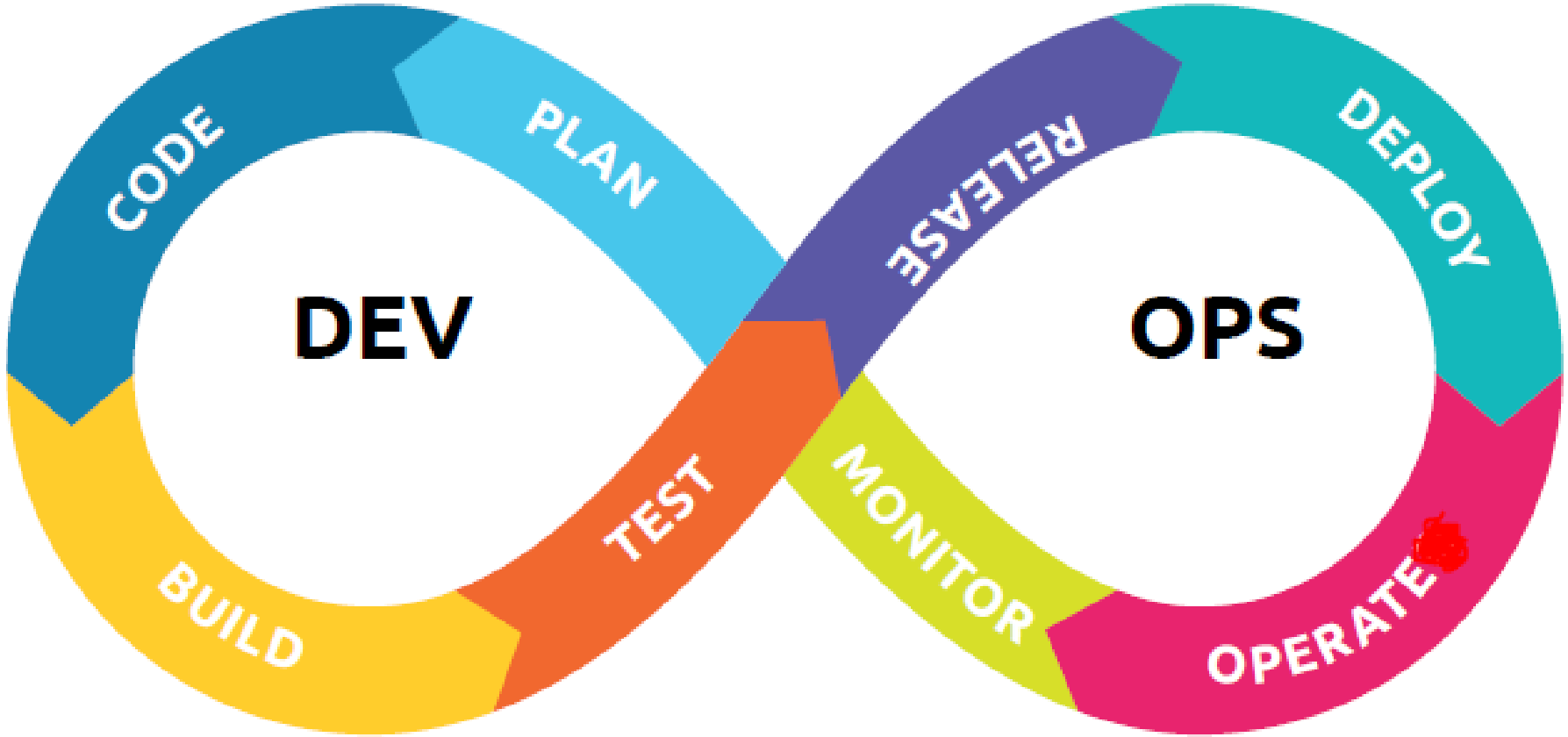


Figure 1. DevOps relies on a single team composed of cross-functional members with responsibility for the component lifecycle

CLOUD NATIVE APPLICATIONS

Cloud applications are the key to interacting with customers, partners, and employees.

Application development can make or break any business's success.

Developers build, deploy, and run applications more securely -- anywhere they are needed -- including across multiple hybrid Clouds.

CNA breaks down a monolithic applications into smaller components using microservices architecture.

Each microservice performs a specific business function

ADVANTAGES OF MICROSERVICES

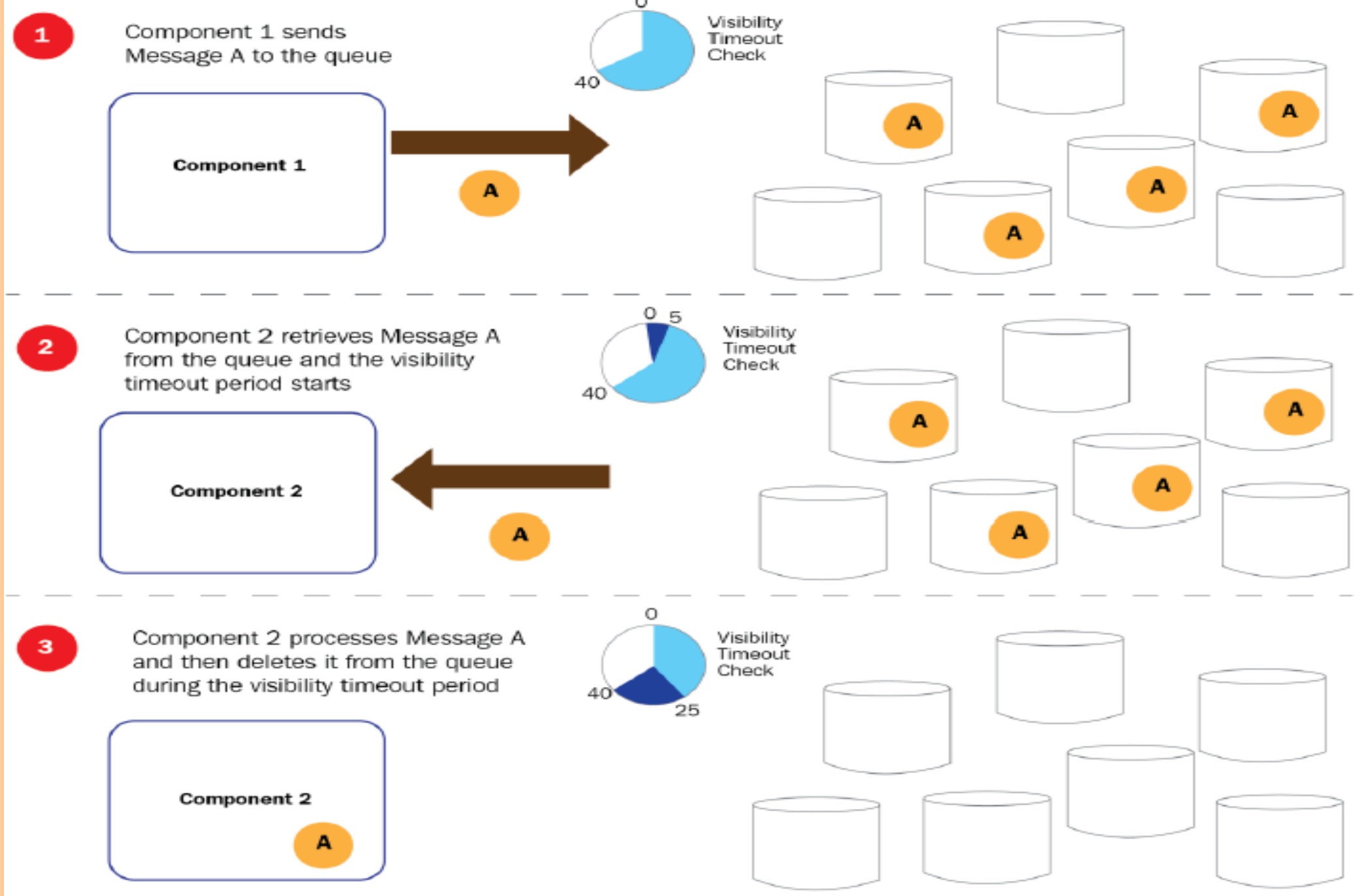
Each microservice manages its own database or data storage, leading to a decentralized approach to data management

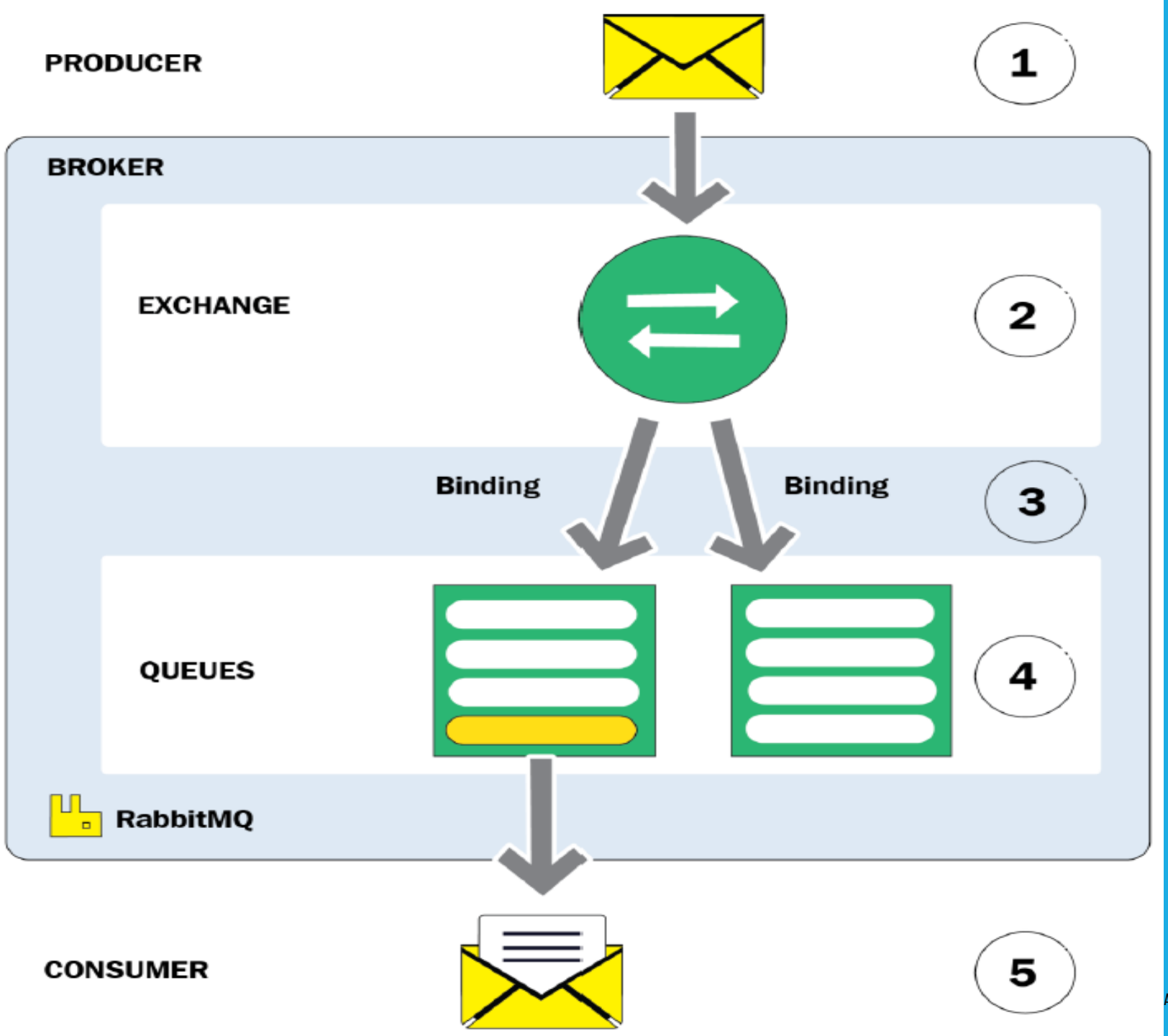
Microservices can communicate with each other using lightweight protocols (sockets, pipes, namespaces, etc. or shared memory).

Synchronous communication uses RESTful APIs, P2P protocol

Asynchronous communication uses messaging systems like RabbitMQ or Kafka.

Microservices are designed to handle failures gracefully.


1
Component 1 sends Message A to the queue
0
40
Visibility Timeout Check
Component 1
A
A
A
A
A
2
Component 2 retrieves Message A from the queue and the visibility timeout period starts
0 5
40
Visibility Timeout Check
Component 2
A
A
A
A
A
3
Component 2 processes Message A and then deletes it from the queue during the visibility timeout period
0
40
25
Visibility Timeout Check
Component 2
A


PRODUCER
1
BROKER
EXCHANGE
2
Binding
Binding
3
QUEUES
4
RabbitMQ
CONSUMER
5

ADVANTAGES OF CNA

Faster development that breaks applications into portable reusable components, speeds up the development life cycle, and reduces time to market.

Greater scalability in a cloud environment that helps you scale applications more easily and cost-effectively.

Improved efficiency using automated processes that makes the IT team more efficient and reduces costs.

Better product quality with DevOps and continuous delivery that helps reduce and eliminate software bugs and improve application quality.

ADVANTAGES OF CNA CONT'D

Enhanced innovation that gives you easier access to on-demand infrastructure and developers more freedom to focus on innovation.

Cloud-agnostic cost-efficiency with microservices and containers that creates portable applications that you can deploy across multiple cloud vendors, helping avoid cloud vendor lock-in.

Future-ready applications with cloud-native development allow users to update applications more quickly and easily to meet changing market demands.

Higher return on investment (ROI) by using cloud-native development allows you to integrate monolithic applications with modern applications and retain the value of your legacy applications.

Cloud-native patterns and best practices

- While the tools and architectural standards of a cloud-native environment are important, using them effectively requires following commonly accepted best practices and patterns.
- Many of these practices fall under the umbrella of DevOps, a software development approach that emphasizes continuous improvement, automation, collaboration and shared ownership in order to improve the speed, quality and reliability of resulting products.

DevOps

- DevOps is a software culture that improves the collaboration of development and operations teams. It is a design philosophy that aligns with the cloud-native model.
- DevOps practices allow organizations to speed up the software development lifecycle.
- Developers and operation engineers use DevOps tools to automate cloud-native development.

Microservices advantage

- The developers of Cloud applications can roll-out smaller patches to existing systems and can add new features without disturbing the users with requests to install software-updates or service-packs.
- Developers have the ability to integrate other services to their Cloud applications on demand.

Cloud-native applications on AWS

- AWS offers the necessary technologies, tools, and services to develop functional CNA. You can focus on building software products instead of worrying about the underlying infrastructure.
- Build new applications or features using serverless technologies with AWS Lambda and purpose-built databases with Amazon DynamoDB
- Use tools like AWS Amplify and AWS CDK to maximize agility and accelerate development
- Choose from 15 relational and nonrelational purpose-built AWS databases to support microservices architecture and modern application needs, such as storing documents and key-value pairs
- Use our portfolio of DevOps services and our vast Partner Network to help develop and run applications faster and build applications at scale

Amazon S3

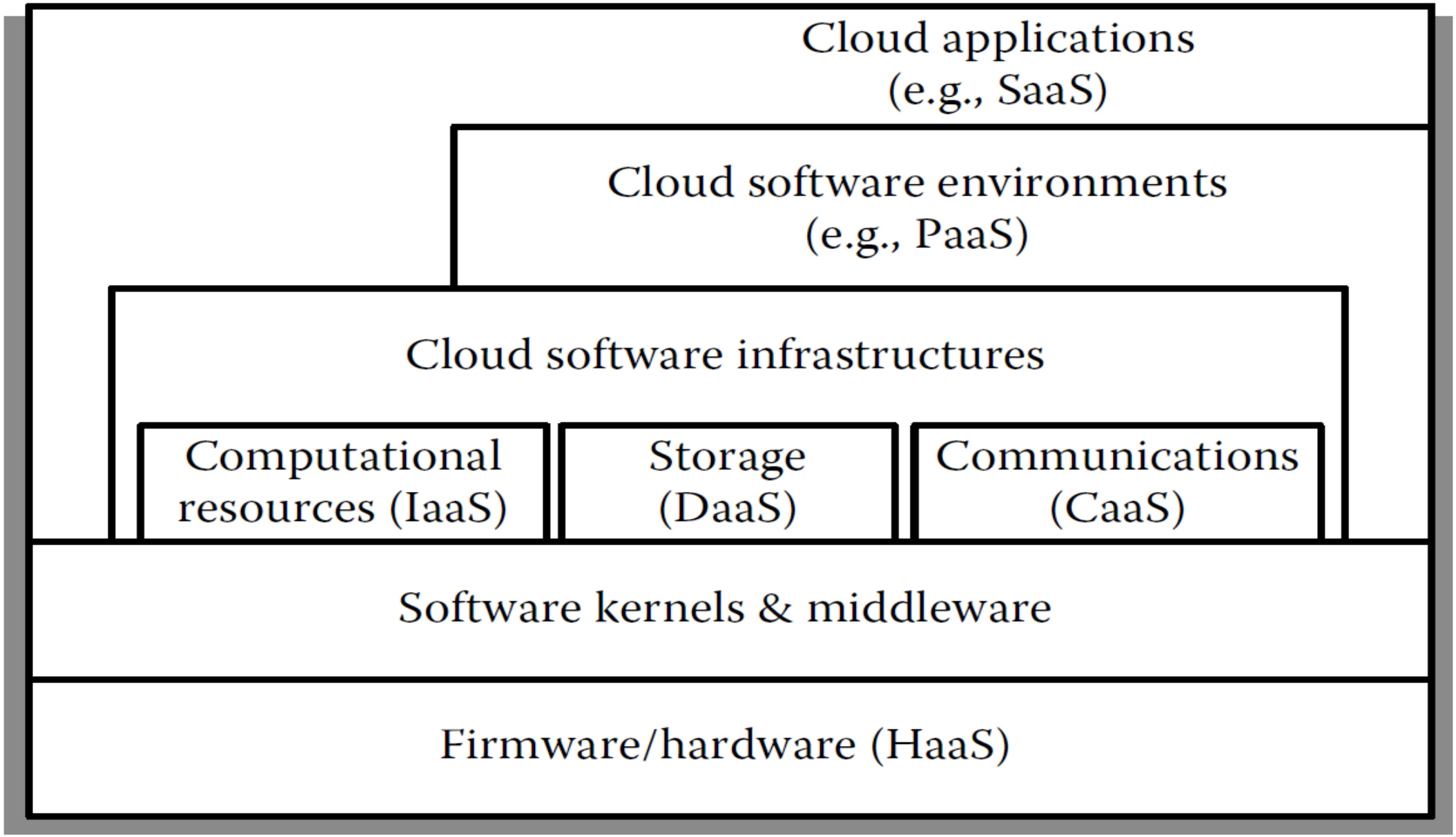
- Amazon S3 (Simple Storage Service) is an object storage service that allows you to store and retrieve data of any size.
- In data engineering, S3 is commonly used as a scalable and durable storage solution for storing raw or processed data.
- Its integration with other AWS services makes it a popular choice for building data lakes, storing backups, and serving as a data source for analytics

AWS Data Pipeline

- AWS Data Pipeline is a web service for orchestrating and automating the movement and transformation of data among AWS services and on-premise data sources.
- It allows you to define and schedule data-driven workflows, making it easier to manage complex data processing tasks.
- Data Pipeline is particularly useful in data engineering for coordinating activities such as data extraction, transformation, and loading (ETL).

Google Golang

- Go, or Golang, designed by Google, has become increasingly popular among cloud engineers for building high-performance and scalable cloud services.
- Its efficiency, simplicity, and built-in support for concurrency make it an excellent choice for developing microservices, distributed systems, and containerized applications.
- Go's compatibility with cloud platforms and its ability to handle heavy network traffic and complex processing tasks efficiently contribute to its growing adoption in cloud infrastructure projects.



UCSB-IBM Cloud Computing Classification Model depicted

DAAS, CAAS

*D*ata-Storage as a *S*ervice (*DaaS*) is an optional service provided by all Cloud service providers.

Examples of DaaS are Amazon's S3 and EMC Storage Managed Service

Communication as a Service (CaaS)

Example of CaaS is the Microsoft Connected Service Framework (CSF)

Voice over IP (VoIP) telephone systems, audio and video conferencing, and instant messaging applications can be developed using CaaS.

Node.js

- Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, event-based and scalable network applications.
- It is perfect for data-intensive real-time applications that run across distributed devices.
- Node uses asynchronous I/O so that it is highly scalable

Node.js

- JavaScript, particularly when used with Node.js, is essential for cloud engineers focused on building and deploying scalable and efficient web applications.
- Node.js allows JavaScript to be used on the server side, enabling the development of fast, non-blocking, event-driven applications suitable for the cloud.
- JavaScript's ubiquity across client-side and server-side development also facilitates full-stack development capabilities, making it invaluable for engineers working on cloud-based web services and applications.

Nodejs.org

- Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. With the advent of Node.js as a server-side JavaScript programming language, it opened up the ability to use JavaScript in the cloud.
- Node.js is built on an event-driven architecture and non-blocking I/O model, which makes it efficient and scalable for I/O-intensive operations.
- This architecture allows Node.js to handle multiple connections concurrently, making it ideal for applications that require real-time interactions and high performance.
- The non-blocking nature of Node.js ensures that operations do not block the execution of other tasks, leading to better resource utilization and faster processing.

Kubernetes YAML

- In traditional sense, Kubernetes YAML (YAML Ain't Markup Language) is essential for cloud engineers working with Kubernetes, the de facto standard for container orchestration.
- Mastery of Kubernetes YAML is crucial for defining, deploying, and managing containerized applications across cloud environments.
- Understanding the intricacies of Kubernetes resource files and configurations allows engineers to leverage the full capabilities of container orchestration, ensuring scalable, resilient, and efficient cloud-native applications.

Language	Advantages	Limitations
Python	Easy to learn, vast libraries, great for automation and data analysis	Slower performance compared to compiled languages
Java	Robust, scalable, platform-independent, strong enterprise support	Verbose syntax, longer development time
JavaScript (Node.js)	Full-stack development, event-driven, excellent for real-time apps	Single-threaded, callback complexity
Go	High performance, excellent concurrency support, ideal for microservices	Less mature ecosystem compared to older languages
Ruby	High developer productivity, great for rapid prototyping	Slower runtime performance, less suitable for high-performance tasks
C#	Seamless integration with Azure, versatile, high-performance	Primarily tied to the Microsoft ecosystem, licensing costs

JavaScript Cloud-Native Development

- Node.js:
- A widely used runtime to build scalable, server-side applications in JavaScript.
- Efficient for I/O-heavy operations, which is common in CNA
- Supported by a rich ecosystem of libraries and frameworks (e.g., Express.js, NestJS, Hapi.js).
- Express.js:
- A minimal, flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- Great for building RESTful APIs and microservices.

JavaScript Cloud-Native Development

- NestJS:
- A progressive Node.js framework built with TypeScript, used for building efficient, reliable, and scalable server-side applications.
- Based on modular architecture and encourages good design patterns like dependency injection.
- Docker:
- A tool to containerize applications, allowing them to be easily shipped and run in any environment.
- JavaScript applications (Node.js-based) can be packaged with Docker for consistency across dev, test, and production environments.

- **GraphQL**:
- A query language for your API that gives clients the power to ask for exactly what they need.
- Ideal for cloud-native applications with dynamic client needs, reducing over-fetching and under-fetching of data.

AWS SDK

- AWS SDK for JavaScript:
- Provides a set of APIs to interact with AWS services such as S3, DynamoDB, Lambda, and API Gateway.
- Ideal for building cloud-native services on AWS using JavaScript.

- Azure SDK for JavaScript:
- For building and deploying cloud-native applications on Microsoft Azure.
- Supports services like Azure Functions, Blob Storage, and Cosmos DB.

MongoDB

- **MongoDB Atlas**:
- A cloud-based NoSQL database that scales horizontally.
- Commonly used in JavaScript cloud-native applications for its scalability and flexibility, often paired with Node.js.

Best practices

- Modular and Decoupled Code: Split code into smaller, reusable components to avoid monolithic applications.
- Use Asynchronous Programming: Node.js is inherently asynchronous, and leveraging async/await or Promises can optimize performance.
- Implement Health Checks: Use health check endpoints and monitoring tools to ensure your microservices are functioning properly.
- Automated Testing: Use testing frameworks like Mocha, Jest, or Chai to automate tests in your CI/CD pipeline.
- Security Best Practices: Secure your APIs and services using best practices such as OAuth, API gateways, and encryption (HTTPS, TLS).
- Observability and Monitoring: Use monitoring and logging tools (e.g., Prometheus, Grafana, ELK Stack) to track the performance of your cloud-native JavaScript applications.
- Service Mesh: In large, distributed environments, use a service mesh like Istio to manage microservice communication, load balancing, and monitoring.

scalable cloud-native apps

- Developing cloud-native applications using JavaScript revolves around utilizing modern cloud technologies, modular development, and scalable deployment practices.
- Node.js, containerization (Docker, Kubernetes), and serverless architectures (AWS Lambda, Azure Functions) are essential tools for building resilient, scalable cloud-native apps.

Creating a Stack

- Every Cloud native service and all resources that it uses are packaged as a cohesive and self-contained group called a Stack.
- On Amazon AWS this is called CloudFormation stack
- You must be familiar with JSON format and xml format (.yml is used in AWS)

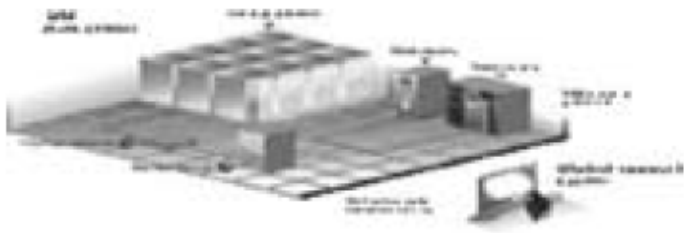
SLS

- Server-Less Service is a tool to deploy Cloud server less resources such as functions. (see https://serverless.com/framework/docs)
- It is an abstraction layer on top of Infrastructure as code tools such as AWS CLoudFormation
- It has extensibility features including plugins, dynamic variables
- New projects are created either by cloning a template (which is fast) or by starting from scratch.
- This creates a .yml file (Yaml) in which service name and providers are specified
- Service names are combined with stage that we have defined.
- Stage and region (specified using the flag –s and –r) are two required options

NPM

- Node Package Manager (NPM) is a dependency management and packaging tool for Cloud native applications
- It has scripting facility to manage various tasks
- This is available at https://docs.npmjs.com
- The “npm” is the executable which can be followed by a keyword such as install, test, deploy etc.
- Npm install
- This command will install all the declared dependencies into the project

Enterprise customer



ISV and development community

Extend enterprise data center

Easy access to resource

Virtual infrastructure

Network

Server

Storage

Software

Isolation

Virtual infrastructure

Network

Server

Storage

Software

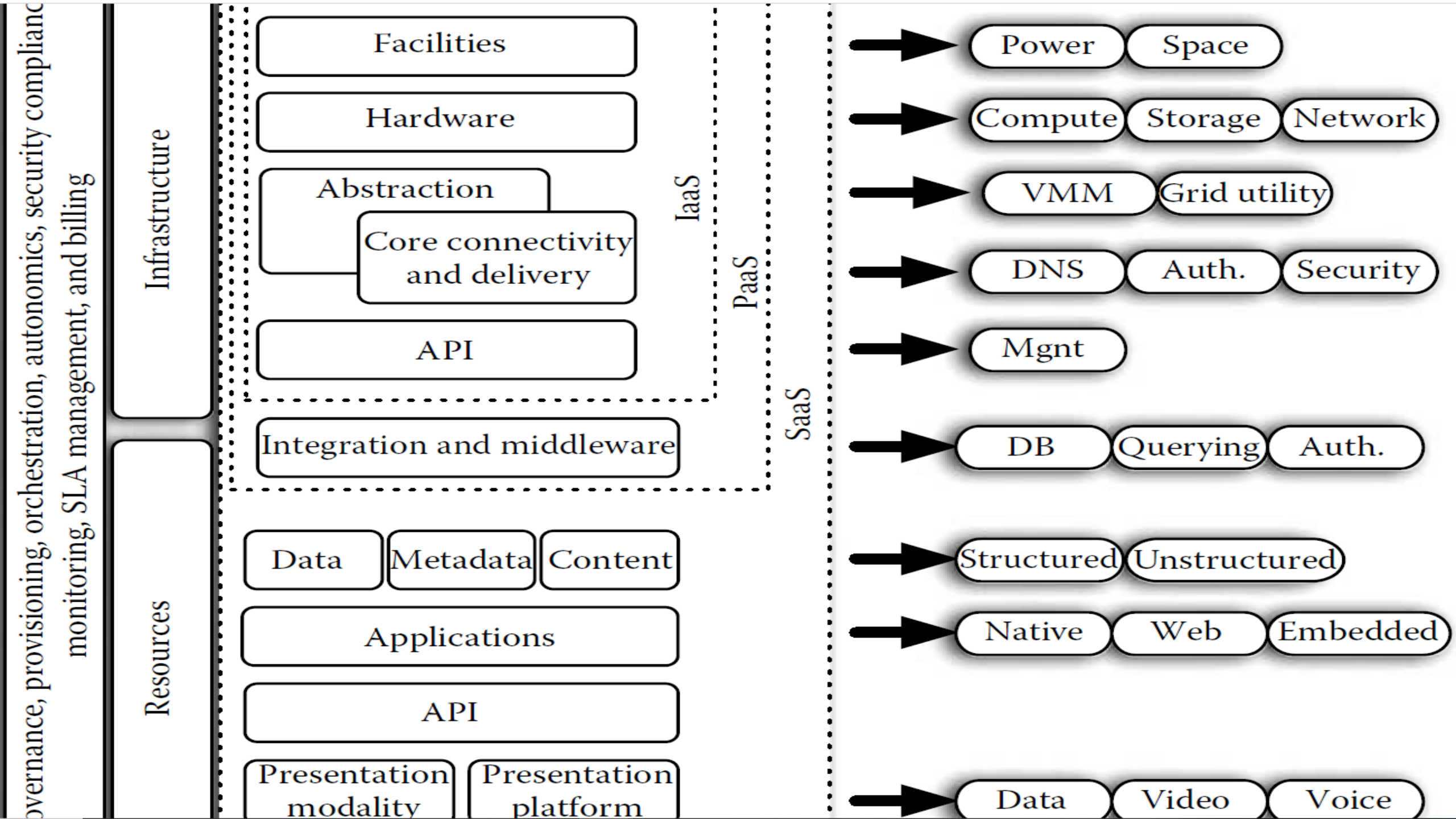
Cloud computing

Server

Network

Storage

Consolidated IDC fabric


overnance, provisioning, orchestration, autonomics, security compliance
monitoring, SLA management, and billing
Infrastructure
Resources
Facilities
Hardware
Abstraction
Core connectivity and delivery
API
IaaS
PaaS
Integration and middleware
SaaS
Data
Metadata
Content
Applications
API
Presentation modality
Presentation platform
Power
Space
Compute
Storage
Network
VMM
Grid utility
DNS
Auth.
Security
Mgnt
DB
Querying
Auth.
Structured
Unstructured
Native
Web
Embedded
Data
Video
Voice