Sustainable Architecture Design Principles for Large Scale Research Infrastructure Projects

Yuri Demchenko

University of Amsterdam

International Conferences on High Performance Computing and Communications (HPCC2023)

Melbourne, 13-15 December 2023

Outline

- Digitalisation of Science and Large Research Infrastructure projects in Europe
  - ESFRI, EOSC, SLICES
- Project management: Use cases and scenarios
  - Importance of System and Software Engineering Skills
- Sustainable Architecture Design principles
  - Standardisation and Best Practices
  - Why Multi-Layer Architecture id essential for infrastructure projects?
  - Why API definition is not sufficient?
  - Other architecture design styles and principles
- Including System and Software Architecture Design principles into Master curricula

Motivation for this (Reflective) Research

- Factor 1: Dropping level of understanding Software and System architecture design principles by the master students: both University of Amsterdam and NTU of Ukraine
  - Consequently, quality of course projects
- Factor 2: Dropping architecture design competences in EU funded projects
  - Demos and pilots ended up with TRL2-TRL3 (Technology Readiness Level)
    - Not repeatable, not operational solutions, create “gravity” of technically not-sustainable solutions

- Observation 1: Increasing number of blogs on the efficient architecture design principles
- Observation 2: Move to sustainable IT and AI must be based on sustainable architecture design principles
- Growing understanding of competences and skills management in research and industry and to be addressed by education
  - Number of European sectoral Skills Blueprints produced in 2018-2022
  - General competence frameworks: DigComp, EntreComp, ResearchComp, GreenComp

European Research Area and Initiatives – 2017-2023: Focus on e-Science and Digital technologies

- European Research Area (ERA) is an important area of the European policy development and funding
  - Supported by main e-Infrastructure projects: GEANT, EGEE, EGI, EOSC (taking off)
- Research Infrastructure (RI) is one of pillar in supporting European science
  - More than 1800 RIs in Europe
  - Coordinated by **ESFRI (European Strategy Forum on Research Infrastructures)**
- ESFRI Roadmap 2021 - https://roadmap2021.esfri.eu/
  - Traditional domains: Social and cultural Innovation, Environment, Health and Food, Energy, Physical Science and Engineering
    - Majority are digital or e-Infrastructure enabled
  - New domain (since RM2021): **DIGIT – Digital and Information Technologies**
    - To experiment, develop and adopt modern digital and data technologies for scientific research (RIs)
    - **SLICES, EBRAINS, SoBigData, OpenAIRE, EUDAT, EGI, EuroHPC**
- **European Open Science Cloud (EOSC)** initiative since 2016 to create a common platform for European RI integration
  - Currently focused on building European Federated Data Access and Sharing Infrastructure federating institutional and sectoral Data Management Infrastructures

slicesPP

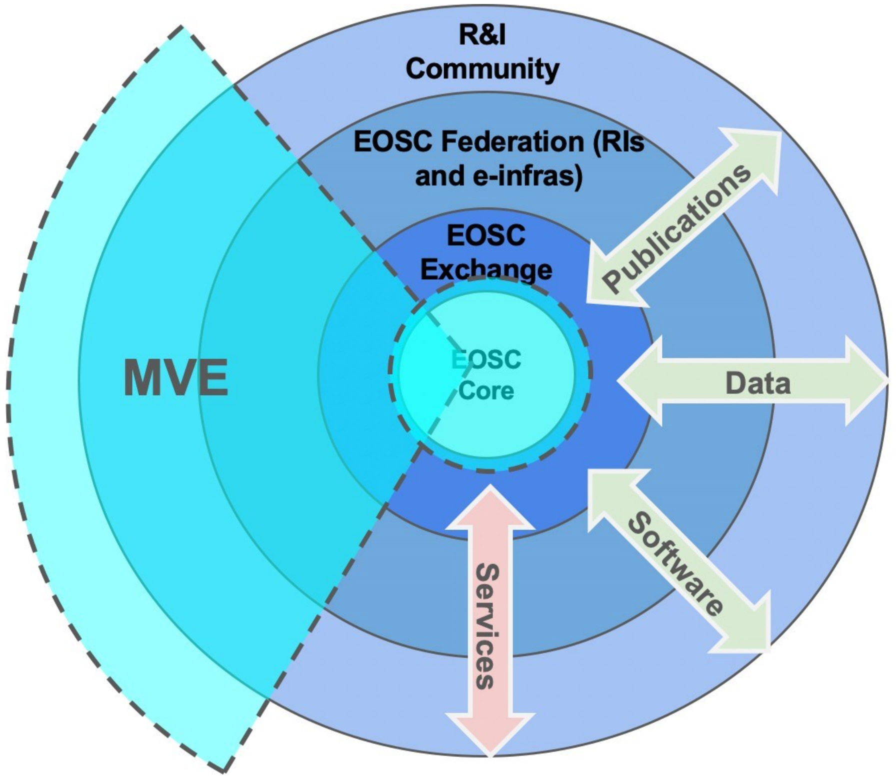
European Open Science Cloud (EOSC)

- EOSC is an overarching concept and framework to integrate existing RIs and facilitate information and data exchange between RI, organisations and researchers
  - First phase 2016-2020 with funded projects 2016-2022
    - 53 projects in total
- EOSC main projects and co-creation activities
  - EOSCpilot – Initial EOSC architecture and requirements (10 Mln, ended 2018)
  - EOSChub – FAIR data principles implementation, Technical integration platform, RI marketplace and API/services directory (33 Mln, ended 31 March 2021)
  - EOSCsecretariat https://www.eoscsecretariat.eu/ (10 Mln, ends 31 October 2021)
    - Establishment of the Governance structure and EU EOSC association
    - SRIA (Strategic Research and Innovation Agenda) document (195 pages, Final version dated 15 Feb 2021) – Provided long term vision for EOSC services but w/o technically consistent architecture definition
  - EOSC-Future (2021-2023) – To develop general architecture and key infrastructure services (30 Mln, Ended 30 Oct 2023)
  - EOSC-Beyond (2024-2027) – Future develop and implement EOSC in operation with user centricity
- Built on experience of the past successful initiatives and project
  - EGEE and WLCG, EGI, RDA (co-founded by EC and NSF), GEANT/TERENA
- Provides a model experience for other EU initiatives, such as GAIA-X European Federated Data Cloud but leadership faded w/o clear architecture definition

slicesPP

EOSC Conceptual Architecture

- EOSC-Core
  - Minimum architecture elements to enable the Federation
- EOSC-Exchange
  - Evolving Federation to serve the needs of research communities
  - Widening to the general public and the private sector
- EOSC Federation (RIs and e-Infra)
- Research and Innovation Community
- Minimal Viable EOSC (aka MVE)
  - Minimum Federation to bring value to users



Projects Setup: Teams composition and expertise

- **Case 1 (reference or successful). Development of cloud based applications for research and industry (Innovation Action).**
  The team is composed of specialists from all relevant domains (infrastructure, network, DevOps/Agile, security, software development, use cases domain) with sufficient experience in system and software engineering at different levels (architecture, requirement engineering, integration).
- **Case 2** (*not sufficient* system engineering expertise). Development of the complex research infrastructure to support experimentation on digital infrastructure technologies.
- **Case 3** (*lack of* system engineering and infrastructure expertise). This is the often case when the project requires delivering digital infrastructure to support specific domain research, but the project consortium partnership lacks partners in computer or infrastructure technologies, system and software engineering.
- **Case 4** (*not sufficient knowledge of background technologies*). This scenario can be a variation of scenarios 2 or 3. The project architect (or person responsible for the architecture development) has experience from the previous successful RI or e-Infrastructure projects of 5-8 years ago when performing in a junior position or specific task leader.

slicesPP

Project Development Scenarios for Success

- **Scenario 1 (Structural architecture driven design).** The project team has sufficient experience in domain specific systems from the previous work or projects, but the new project requires wider expertise.
  - **Scenario 2 (Designing Up, Dow**The team starts with implementing available solutions that address the main goals of the system or infrastructure, defines the overall architecture, and requirements to the core components
  - To solve the problem of the expertise gap, the team (actually the team leader) looks for cooperation with other projects and invites experts to share expertise.
  - A dedicated team member is assigned to gain new expertise and take over necessary components implementation in the future.
- **Scenario 2 (Designing Up, Down, and Out):** System implementation is started without consistent and grounded architecture definition but with a large availability of legacy components and sufficient expertise in ICT systems design from the previous project.
  - The project or team starts with implementing some infrastructure and services islands, often siloed services, based on available expertise and already existing components.
  - To proceed with the development and implementation, start the structural analysis of the services being developed and put them in the context of the defined system architecture following best practices
- **Scenario 3 (ad-hoc services piloting).** The project is started by the community with identified needs for information digital infrastructure services but without prior experience in building such services, for example, building research infrastructure for social or environmental sciences, humanities.
  - The project may succeed in user needs studies and defining user requirements, but it may fall short of transforming user requirements to technical system requirements and corresponding architecture and functional design.
  - The project may end up with the pilot services for demonstration of the proof-of-concept and stand-alone tools but further successful development will require a professional approach in infrastructure design, what in its turn may also require services re-design.

slicesPP

Why Architecture is Important?

- Long term vision and coordination of cooperating teams
- Blueprint for designing, building, and maintaining the complex and interconnected systems that underpin research infrastructure

- Architecture design thinking – What does it mean?
  - To be cultivated starting from university education
  - Provided as professional training

- Lack of common background may create confusion
  - Lack of well defined architecture may result in divergent development
  - Divergence may turn to convergence in technical development
    - If a critical point not missed – due to gravity of wrong solutions

slicesPP

Important Overloaded Terms that may create confusion and divergence in project development

- **Architecture concept** is itself often understood in different ways by researchers and developers with different backgrounds and experiences.
  - Way to achieve homogeneity is learning by examples
- **Architecture, reference architecture, framework, reference model**
  - All these terms are in many cases interchangeable, often used together and in combination, but have their own meanings in specific contexts.
- **Architecture diagram, architecture model**, functional diagram, process or sequence diagram
- **Data driven** and **data centric** architecture, models and applications vs storage architectures
  - Must ensure continuity in data handling and processes management
  - Relate to workflow driven models
- **Multi-layer and multi-tier** systems in system and applications engineering, and **multi-level systems** in security engineering
- **Blueprint and Bill of Materials (BOM)** as general concepts and similarly named concepts in DevOps and CI/CD in software engineering.
- **Metadata and data modeling** definitions as they are defined in industry and research data management domains against information models and metadata in telecom and Internet service management.
- **Security as a general concept and those related to different security domains** such as computer security, network/ Internet security, application security, cryptography, hardware security, operational security, access control, identity management and trust management.

Why Layered/Multi-Layer Architecture is important?

- Services (resources) separation
  - Independent development and update/evolution: changes in one layer doesn’t affect others layers
  - Re-use of services
  - Multi-provider integration
  - Inter-layer interfaces (also specialized, not only WSDL/SOAP or REST/JSON based)
- Services Composition/integration and Orchestration
- Simplifies distributed systems integration
- Simplifies/structure services management
  - Service Information model: namespaces, semantics, metadata
- Sustainable system evolution
  - Similar to LTE in 3GPP and 5G/6G

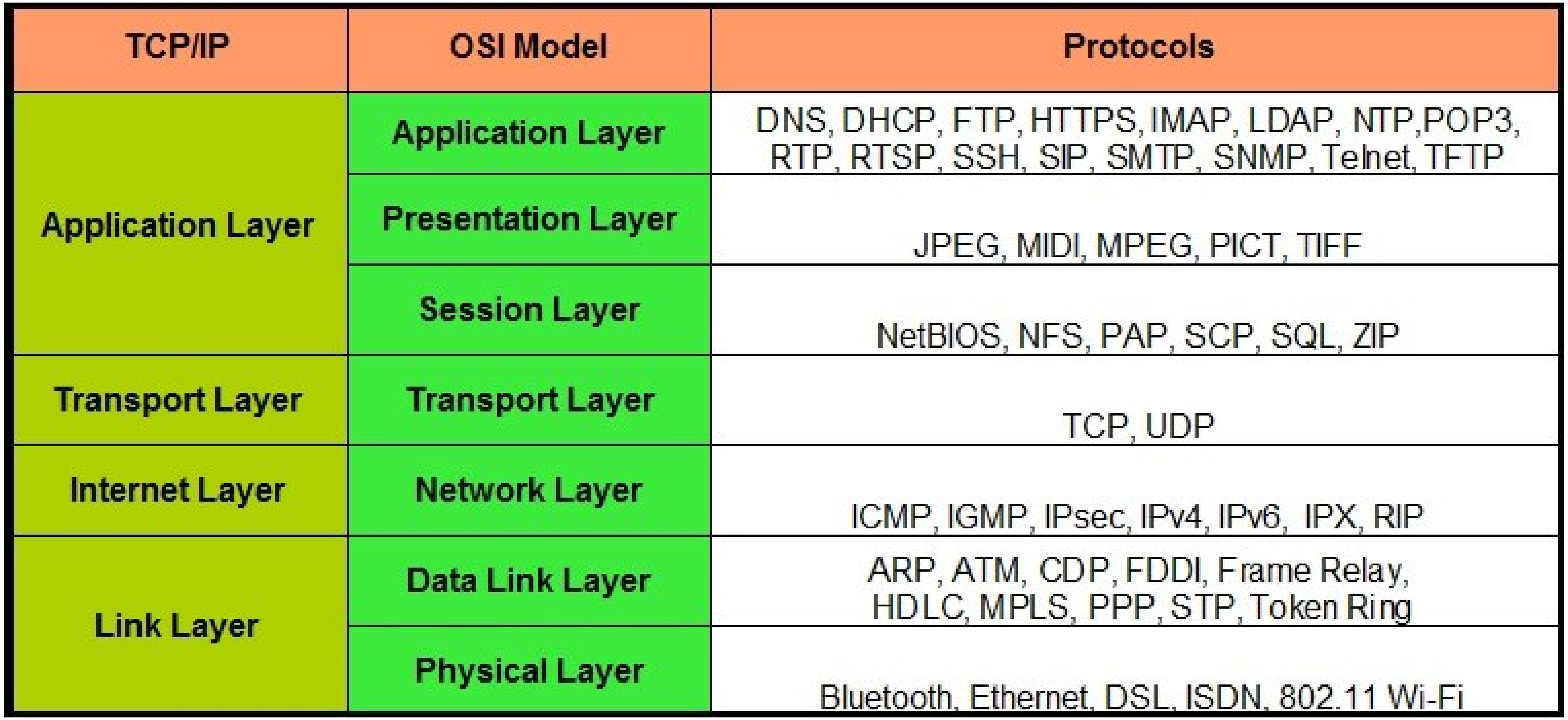
slicesPP

Standardisation: Past based and Forward looking Best Practices

- Internet Architecture: TCP/IP stack
- Telecom OSI (Open System Interconnection)
- TeleManagement Forum
- ITU-T
    - ITU Q.4068 : Open application program interfaces (APIs) for interoperable testbed federations
- ISO and IEEE
- Service Oriented Architecture (SOA)
- NIST Cloud Computing Reference Architecture
    - NIST Big Data Architecture

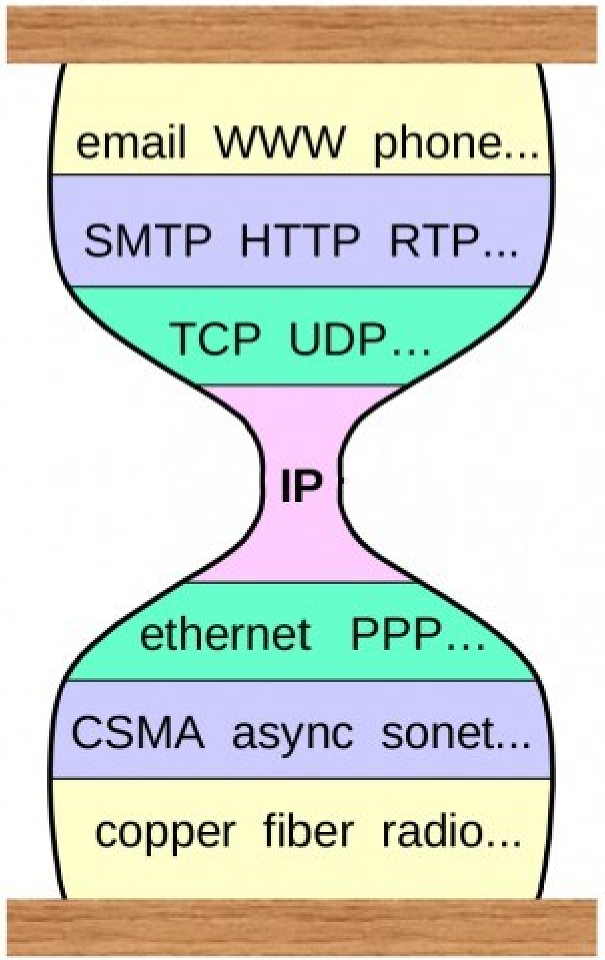
slicesPP

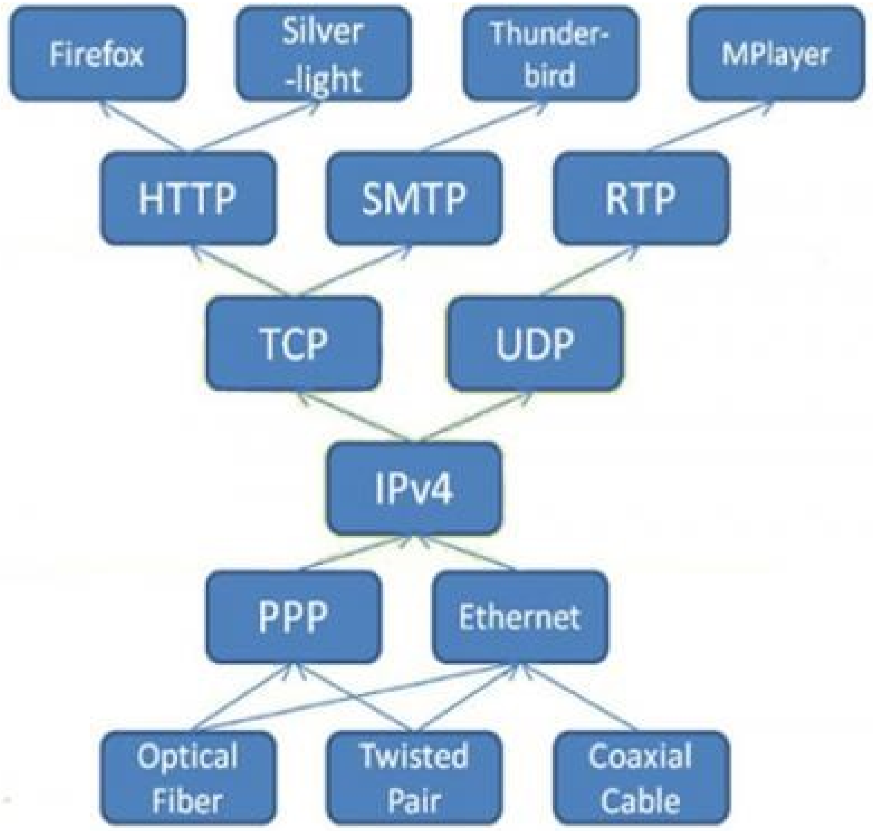
TCP/IP - Internet Protocol and Transmission Control Protocol



TCP/IP	OSI Model	Protocols
Application Layer	Application Layer	DNS, DHCP, FTP, HTTPS, IMAP, LDAP, NTP,POP3, RTP, RTSP, SSH, SIP, SMTP, SNMP, Telnet, TFTP
	Presentation Layer	JPEG, MIDI, MPEG, PICT, TIFF
	Session Layer	NetBIOS, NFS, PAP, SCP, SQL, ZIP
Transport Layer	Transport Layer	TCP, UDP
Internet Layer	Network Layer	ICMP, IGMP, IPsec, IPv4, IPv6, IPX, RIP
Link Layer	Data Link Layer	ARP, ATM, CDP, FDDI, Frame Relay, HDLC, MPLS, PPP, STP, Token Ring
	Physical Layer	Bluetooth, Ethernet, DSL, ISDN, 802.11 Wi-Fi

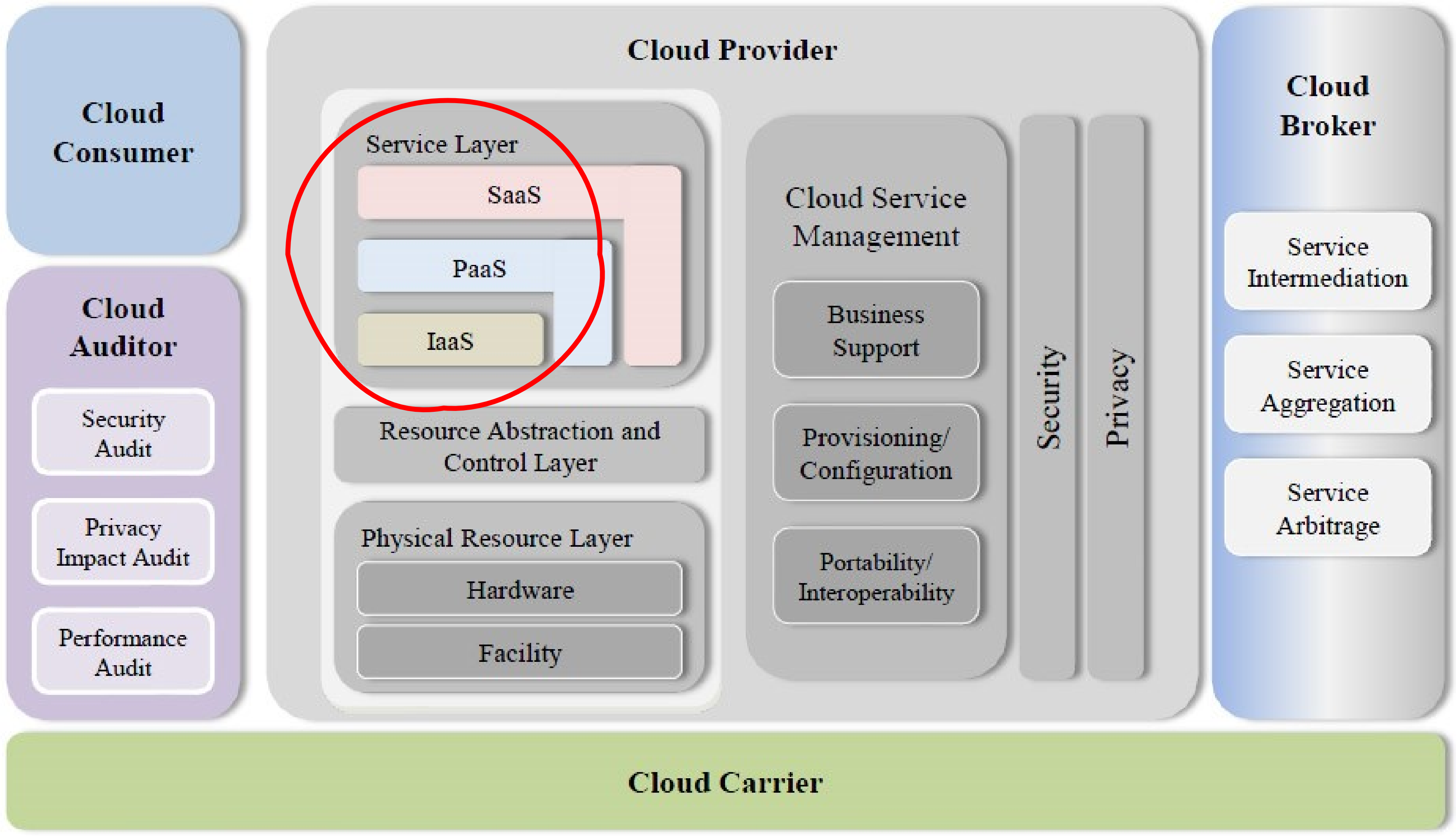
Internet Protocols “narrow waist” - Hourglass





- Source: [2001 Presentation by Steve Deering]

Example Multi-layer NIST Cloud Computing Reference Architecture



- Formal definition of the Cloud Computing technology and core architecture in 2008 built the industry
- 5 Properties:
  - On-demand self-service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured Service
- 3 main service models: IaaS, PaaS, SaaS
- Deployment models
  - Private clouds
  - Public clouds
  - Hybrid clouds
  - Community clouds

Sustainable Architecture Design Principles (SADP)

- Infrastructure related
- Service Architecture related
- Data Management Infrastructure related
- Security and compliance design principles
- Project Management and DevOps
  - Agile development practices

Sustainable Architecture Design Principles

General architecture design principles

- Layered architecture design for services and mechanisms, including inter-layer interfaces, including cross-layer services and mechanisms definition that are typically defined as service planes, for example, management plane, security plane, data management plane.
- Multi-tier services and infrastructure design, including combined multi-layer and multi-tier systems that may use or apply different architectural and layered solutions.
- Application Programming Interfaces (API) for composable services that must be supported by consistent (and fully qualified API metadata and namespaces definition).

Service architecture related

- Service Oriented Architecture (SOA) and Microservices Architecture (MSA) that is supported with the different VM and container solutions and/or platforms.
- Cloud powered, cloud based and cloud native design principles that require knowledge of the modern cloud architecture and cloud platform, both Open Source and public clouds (at least Amazon Web Services, Microsoft Azure, and Google Cloud Platform). This also includes such powerful cloud based mechanism as Virtual Private Cloud (VPC) that provide VPN based secure environment for multi-tenant customer applications.
- Service lifecycle management model that should include all necessary services to support lifecycle stages in the context of specific services. This also includes services composition and orchestration for services deployment and operation.

Data infrastructure and services related

- Big Data computation models and supporting platform, distributed and highly scalable systems, in particular Hadoop ecosystem and NoSQL databases.
- Data management infrastructure and services that should cover two domains: services data (mostly related to management plane) and business or research data produced as a result of business operation or scientific research.
- Services and data management continuity in IoT/sensor networks, edge, cloud, data-driven applications that also include 5G/6G Radio Access Network (RAN), edge and cloud convergence.

Security and compliance design principles

- Security architecture and security services lifecycle management which are well defined by numerous standards and supported by the major infrastructure development frameworks; also security services have their own multi-layer architecture (can also be referred as security plane), their integration with the main infrastructure services, including data infrastructure) is realized via API calls and consistent definitions of the security roles, access control policies and credentials and secrets management.
- Compliance frameworks that define requirements to and recommendations for secure services and infrastructure design and operation. Cloud Security Alliance (CSA) and Compliance Assessment Initiative Questionnaire (CAIQ) provide the best overview of all important standards and regulations to ensure systems security and compliance, and data protection.

Project Management and DevOps

- DevOps and SRE (Site Reliability Engineering) practices applied to system and services engineering and operation. This should also include continuous monitoring and optimisation on multiple user -centric and business-centric SLI/KPI (Service Level/Key Performance Indicators).
- DevSecOps that extended the DevOps model and practices by addressing security aspects during the whole system/services lifecycle, intending to address "Security by Design" concept (however not yet fully developed)
- General compliance with the project management principles, models and procedures applied to infrastructure, services, and data handling and analytics.

slicesPP

SADP: General architecture design principles

General architecture design principles

- **Layered architecture design** for services and mechanisms, including inter-layer interfaces. Including cross-layer services and mechanisms definition that are typically defined as service planes, for example, management plane, security plane, data management plane.
- **Multi-tier services and infrastructure design**, including combined multi-layer and multi-tier systems that may use or apply different architectural and layered solutions.
- **Application Programming Interfaces (API)** for composable services that must be supported by consistent (and fully qualified API metadata and namespaces definition).

SADP: Service architecture related

Service architecture related

- **Service Oriented Architecture (SOA) and Microservices Architecture (MSA)** that is supported with the different VM and container solutions and/or platforms.
- **Cloud powered, cloud based and cloud native design principles** that leverages (require knowledge) of the modern cloud architecture and cloud platform, both Open Source and public clouds (at least Amazon Web Services, Microsoft Azure, and Google Cloud Platform). This also includes such powerful cloud based mechanism as Virtual Private Cloud (VPC) that provide VPN based secure environment for multi-tenant customer applications.
- **Service lifecycle management model** that should include all necessary services to support lifecycle stages in the context of specific services. This also includes services composition and orchestration for services deployment and operation.
- **Operations and management support**: SLA, monitoring, audit, (certification)

SADP: Data infrastructure and services related

Data infrastructure and services related

- **Big Data computation models** and supporting platform, distributed and highly scalable systems, in particular Hadoop ecosystem and NoSQL databases.
- **Data management infrastructure and services** that should cover two domains: services data (mostly related to management plane) and business or research data produced as a result of business operation or scientific research.
- **Services and data management continuity** in IoT/sensor networks, edge, cloud, data-driven applications that also include 5G/6G Radio Access Network (RAN), edge and cloud convergence: data ID, data models, data linkage and lineage for different types of data: experiment (stimulus and result), configuration, metadata, process models

SADP: Security and compliance design principles

Security and compliance design principles

- Security architecture and security services lifecycle management which are well defined by numerous standards and supported by the major infrastructure development frameworks. Although security services have their own multi-layer architecture (can also be referred as security plane), their integration with the main infrastructure services, including data infrastructure) is realized via API calls and consistent definitions of the security roles, access control policies and credentials and secrets management.
- Compliance frameworks that define requirements to and recommendations for secure services and infrastructure design and operation. Cloud Security Alliance (CSA) and Compliance Assessment Initiative Questionnaire (CAIQ) provide the best overview of all important standards and regulations to ensure systems security and compliance, and data protection.
- Industry best practices adopted: zero-trust access control, Federated AuthN, AuthZ, Identity Management, session and security context management

SADP: Project Management and DevOps

Project Management and DevOps

- **DevOps and SRE** (Site Reliability Engineering) practices applied to system and services engineering and operation. This should also include continuous monitoring and optimisation on multiple user-centric and business-centric SLI/KPI (Service Level/Key Performance Indicators).
- General compliance with the project management principles, models and procedures applied to infrastructure, services, and data handling and analytics, e.g. IEEE PMI.
- **DevSecOps** that extended the DevOps model and practices by addressing security aspects during the whole system/services lifecycle, intending to address “Security by Design“ concept (however not yet fully developed)

DevOps culture controversy:

- Agile works with tightly cooperated and often compact located team to enable skills and experience sharing, continuous development steering
- May work bad for distributed part time team
- “Fail fast” principle lowers importance of professional architecture design
- User stories doesn’t substitute consistent Requirements Engineering

SADP In Education and Training

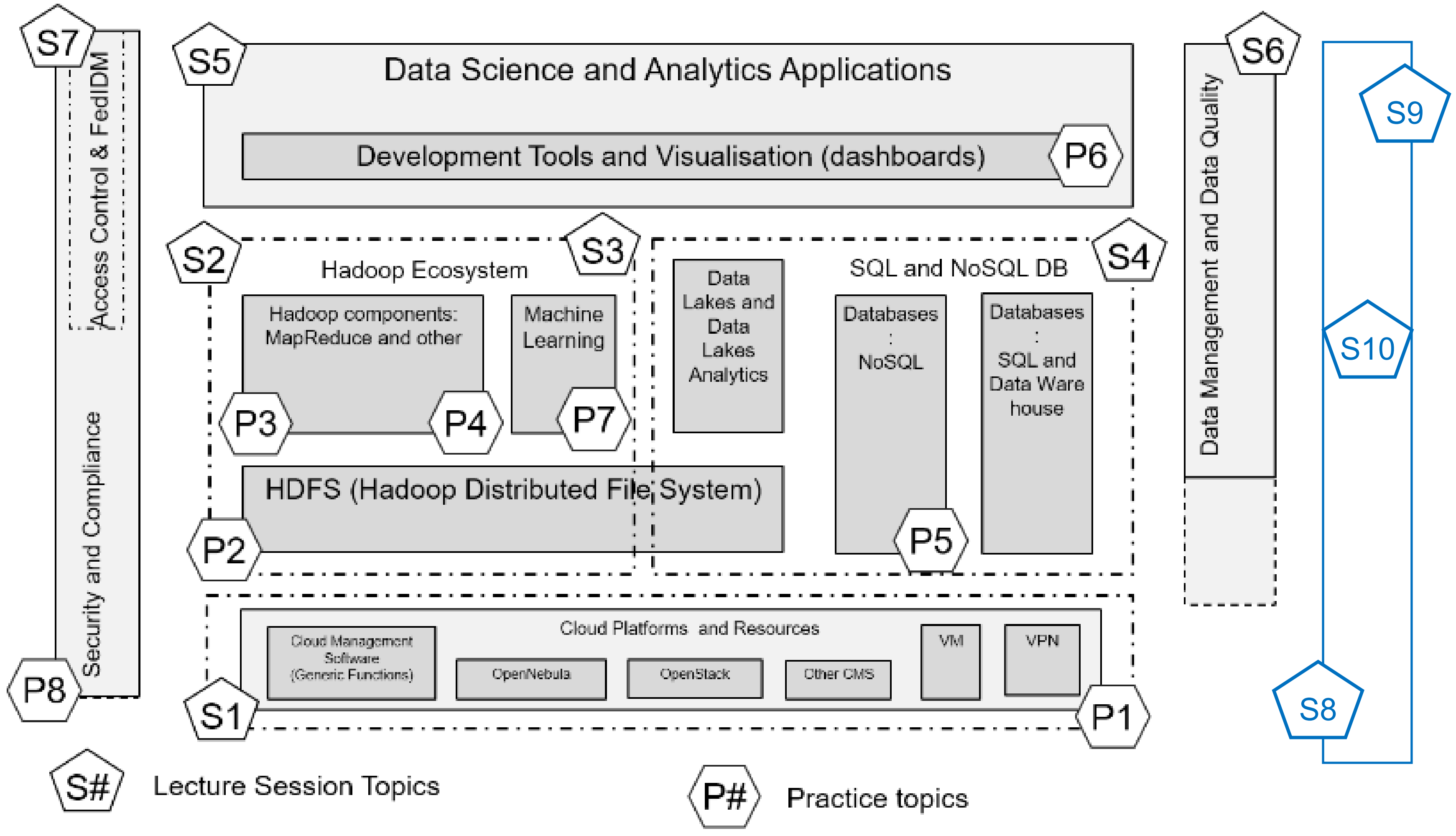
- SADP and architecture design patterns in the course Big Data Infrastructure Technologies for Data Analytics (BDIT4DA)
  - NTUU Igor Sikorski “Kyiv Polytechnic Institute” and Vodafone Ukraine Data Science Academy
- SADP and cloud based design patterns
  - DevOps and Cloud based Software Engineering at the University of Amsterdam (UvA)
- SLICES Academy: Starting with Summer School and providing online MOOC courses on demand
  - SADP and Experimental Data Management

Big Data Infrastructure Technologies for Data Analytics Master Course (NTUU "KPI" & UvA)

- Module 1: Introduction to the course. Cloud Computing foundation. Cloud service models, cloud resources.
- Module 2: Big Data architecture framework, cloud based Big Data services and platforms.
- Module 3: Big Data Algorithms: MapReduce, Pregel. Hadoop platform and components for Big Data analytics: HDFS, YARN, MapReduce, HBase, Pig, Hive, others.
- Module 4: Data Streams and Streaming Analytics. Kafka, Flume. Spark architecture and popular Spark platforms, DataBricks.
- Module 5: SQL and NoSQL Databases. CAP Theorem. Modern large scale databases AWS Aurora, Azure CosmosDB, Google Spanner.
- Module 6: Data Management and Governance. Research Data Management and FAIR data principles in data management.
- Module 7: Big Data Security and Compliance. Cloud compliance standards and cloud provider services assessment.
- Module 8: Platforms and tools for Data Analytics and NL pipeline automation (such as AWS SageMaker or Azure MLOps, supported with Data Lakes)
- Module 9: Managing Data Science Projects. Research methods and project organisation. Data Science Process Models, DataOps and MLOps.
- Module 10: Sustainable Architecture Design principles and cloud based design patterns, which will be extended with discussed in this paper and design patterns.

slicesPP

Big Data Infrastructure Components and Course Modules


S7
Access Control & FedIDM
Security and Compliance
P8
S5
Data Science and Analytics Applications
Development Tools and Visualisation (dashboards)
P6
S2
Hadoop Ecosystem
S3
Hadoop components: MapReduce and other
Machine Learning
P3
P4
P7
HDFS (Hadoop Distributed File System)
P2
Data Lakes and Data Lakes Analytics
SQL and NoSQL DB
S4
Databases : NoSQL
Databases : SQL and Data Ware house
P5
Cloud Platforms and Resources
Cloud Management Software (Generic Functions)
OpenNebula
OpenStack
Other CMS
VM
VPN
S1
P1
S6
Data Management and Data Quality
S9
S10
S8
S# Lecture Session Topics
P# Practice topics

- Module 9: Platforms and tools for Data Analytics and NL pipeline automation (such as AWS SageMaker or Azure MLOps, supported with Data Lakes)
- Module 8: Managing Data Science Projects. Research methods and Data Science project organisation. Data Science Process Models, DataOps and MLOps.
- Module 10: Sustainable Architecture Design principles and cloud based design patterns, which will be extended with discussed in this paper and design patterns.

slicesPP

Importance of Data Management topics in University Curricula

- Data Management and Governance (DMG) and corresponding infrastructure services are important for data driven and data centric architecture models
- FAIR data principles: Research data must be Findable, Accessible, Interoperable, Reusable
  - Metadata and namespace management
  - Lineage and provenance, linked data
  - PID (Persistent Identifier) and data discoverability

Technical implementation of the **FAIR data principles** requires comprehensive **data management infrastructure** to support

- **Data Storage**
- **Metadata Registries**
- Data publication
- Data discovery
- Linked data and data lineage (provenance)
- Multiple datasets access for analysis

slicesPP

DevOps and Cloud based Software Engineering (DevOps4SE)

- DevSecOps; Secure Software Development Lifecycle Management (SDLM); cloud based tools for secure software development and testing
  - Extended with compliance and Risk assessment to cover the whole product lifecycle
- Data Science project management and DataOps/MLOps processes and platforms
  - MLOps maturity levels
- Sustainable Architecture Design principles and cloud based design patterns

Future work and development

- Further extensive study of architecture styles and principles
  - Link to TOGAF and Enterprise Architecture Model
- SADP and Legacy solutions + “gravity” of siloed design
- Following and contributing to professional blogs like Medium, SCUB LAB
- Identify new architecture design principles in such areas as 5G/6G, AI and LLM/GenAI enabled
- Develop SADP related topics for different Software and System Engineering courses
- Run seminars and tutorials at conferences and professional events
- Possibly write a (text)book

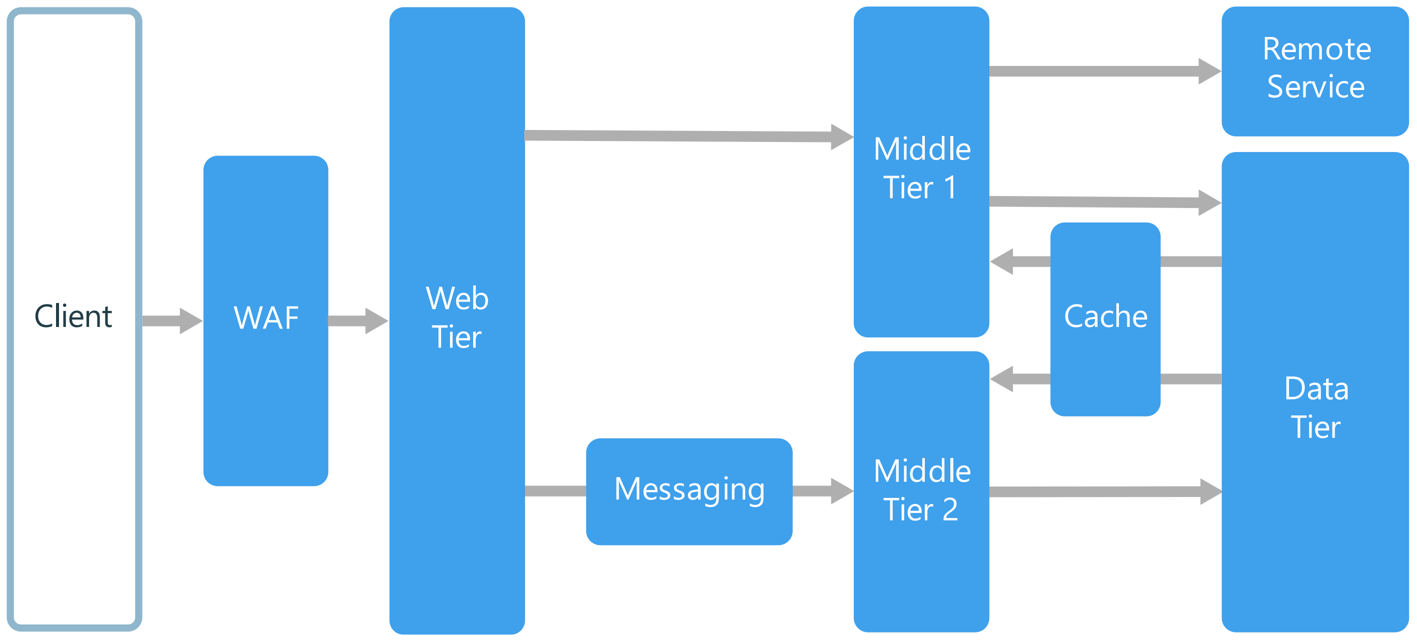
Additional Information (for discussion)

- SADP and Architecture styles illustrated
- FAIR data principles

SADP Illustrated

- Multi-layer architecture style
- 3 and multi/N tiers models
- Microservices
- Web-queue-worker
- Event driven
- Big Data
- HPC/Big Compute

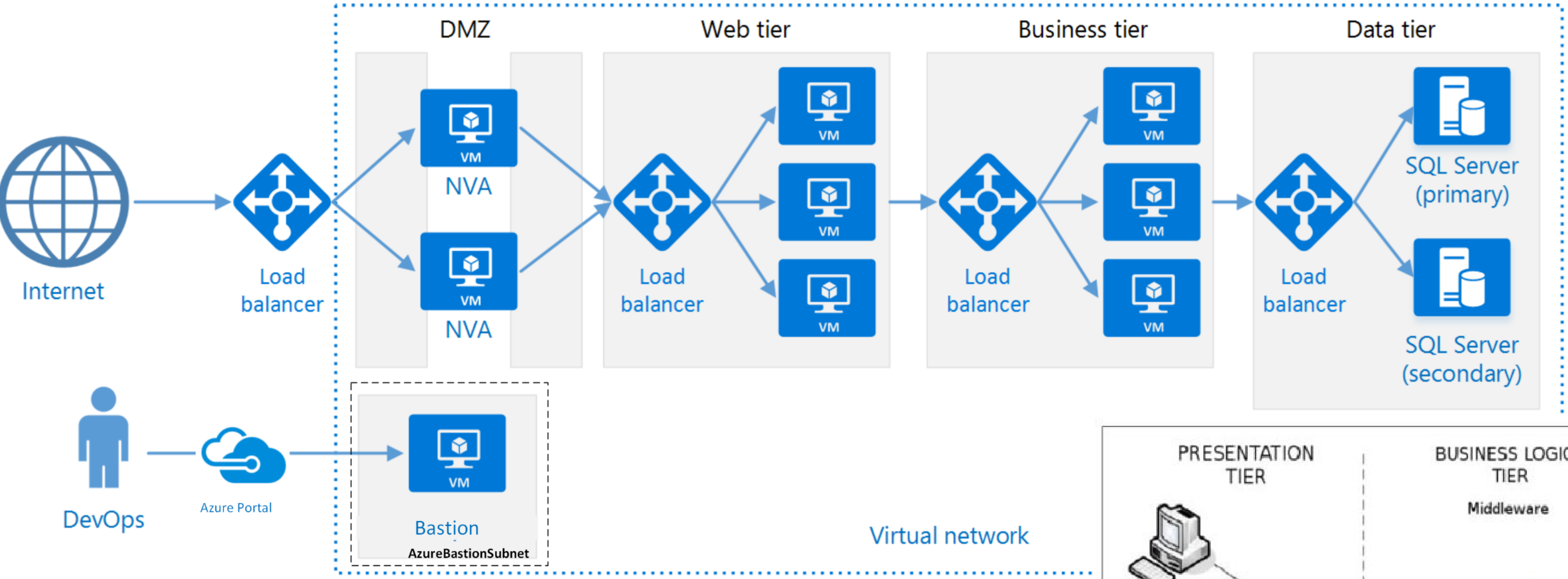
N-Tier Architecture



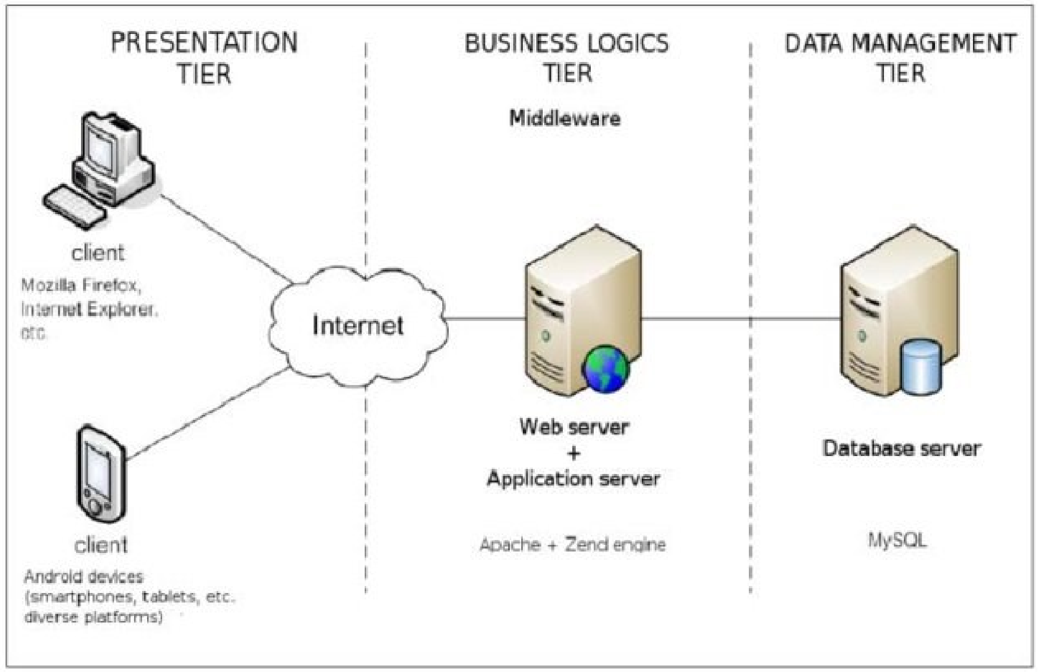
- **N-tier** is a traditional architecture for enterprise applications. Dependencies are managed by dividing the application into *layers* that perform logical functions, such as presentation, business logic, and data access.
- A layer can only call into layers that sit below it. However, this horizontal layering can be a liability.
- It can be hard to introduce changes in one part of the application without touching the rest of the application. That makes frequent updates a challenge, limiting how quickly new features can be added.

slicesPP

3/N tier Architecture Example for Cloud based Applications



- Traditional architecture for enterprise applications
  - Presentation/Web tier – Business Logic (application) – Data tier (assets)

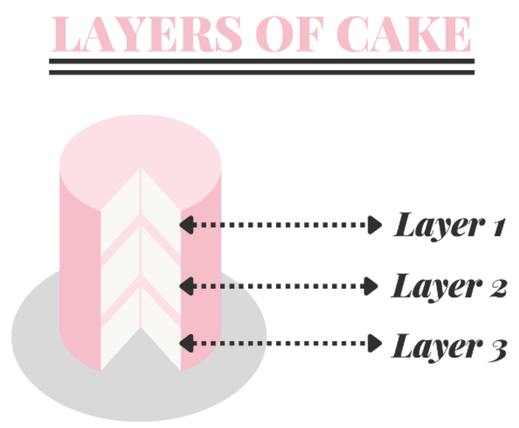

PRESENTATION TIER
BUSINESS LOGICS TIER
DATA MANAGEMENT TIER
Middleware
client
Mozilla Firefox, Internet Explorer, etc.
Internet
client
Android devices (smartphones, tablets, etc. diverse platforms)
Web server + Application server
Apache + Zend engine
Database server
MySQL

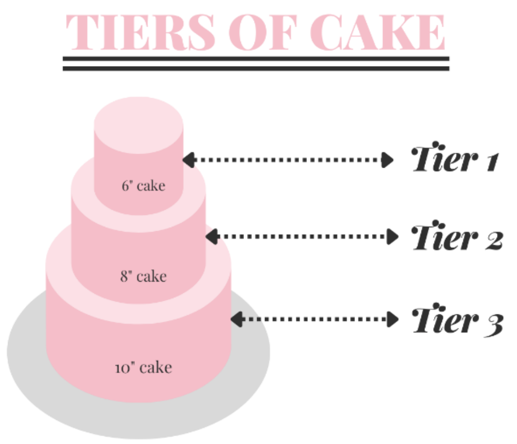
slicesPP

https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

Layers and Tiers

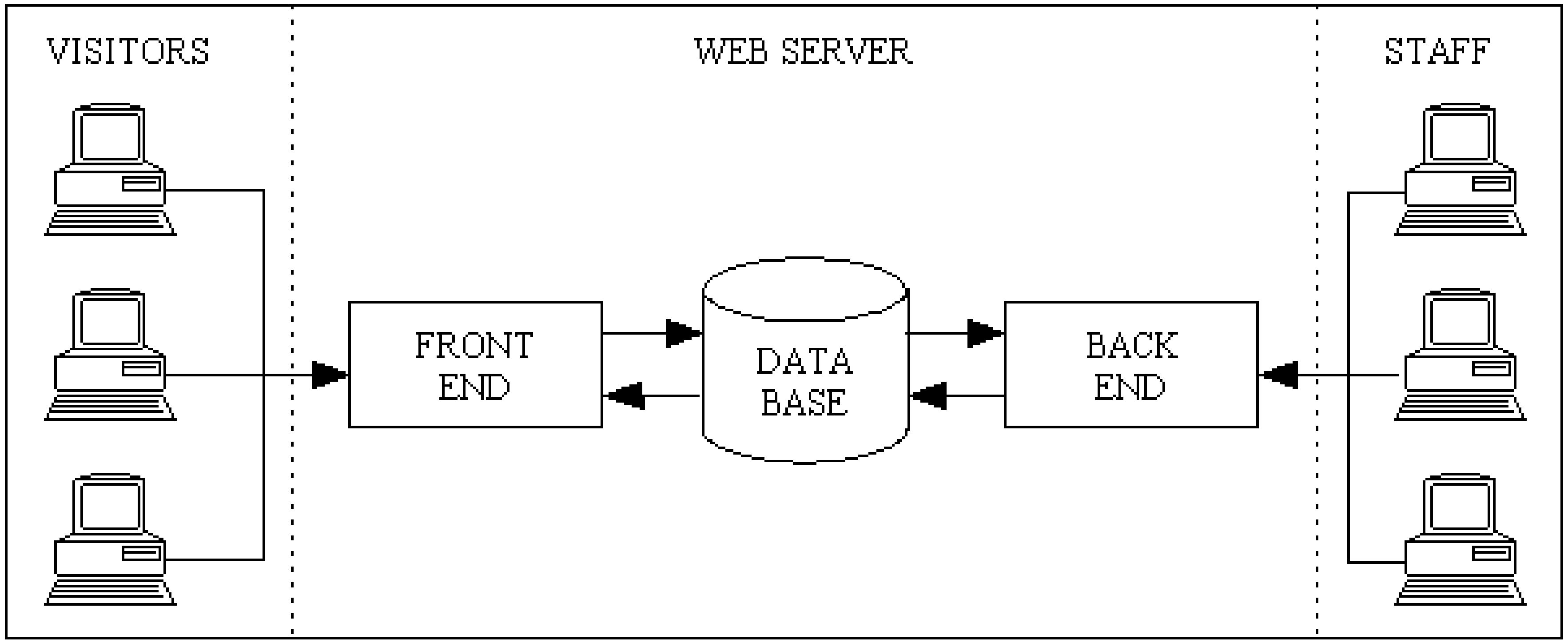
Using cakes metaphor





- Layers are defined to separate functionalities and define interlayer interfaces
  - Multi-layer cloud architecture
- Tiers are defined for physically separate functionalities that communicate via API
  - 3 tier applications architecture
    - Presentation tier
    - Business application/logic tier
    - Data access tier

Frontend and Backend services – General view



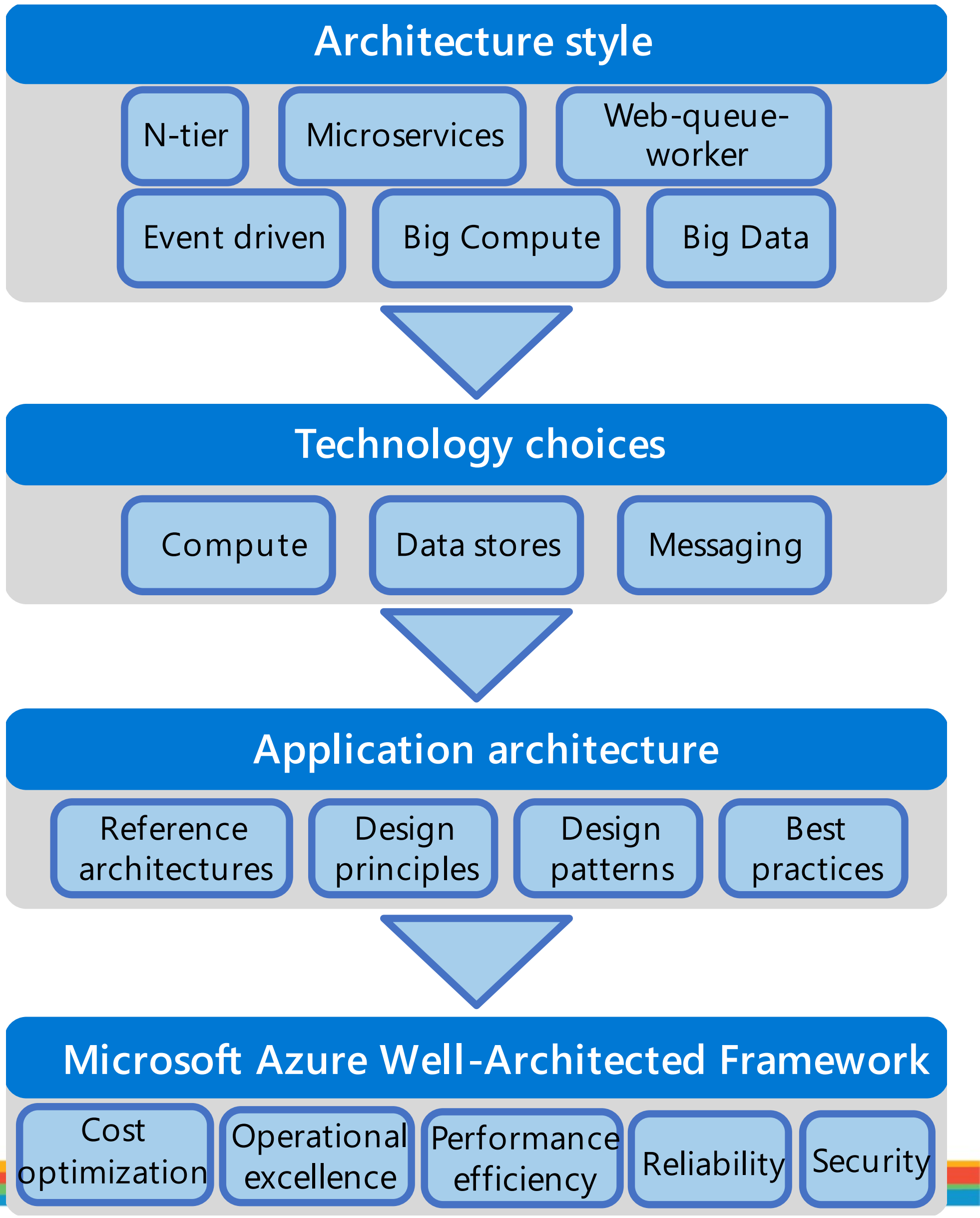
- Using Radicore components in a front-end website, Using Radicore components in a front-end website
  Posted on 1st June 2009 by Tony Marston
  https://www.tonymarston.net/php-mysql/radicore-front-end.html

Azure Well-Architected Framework

- Architecture types
- Design principles
- Best practices

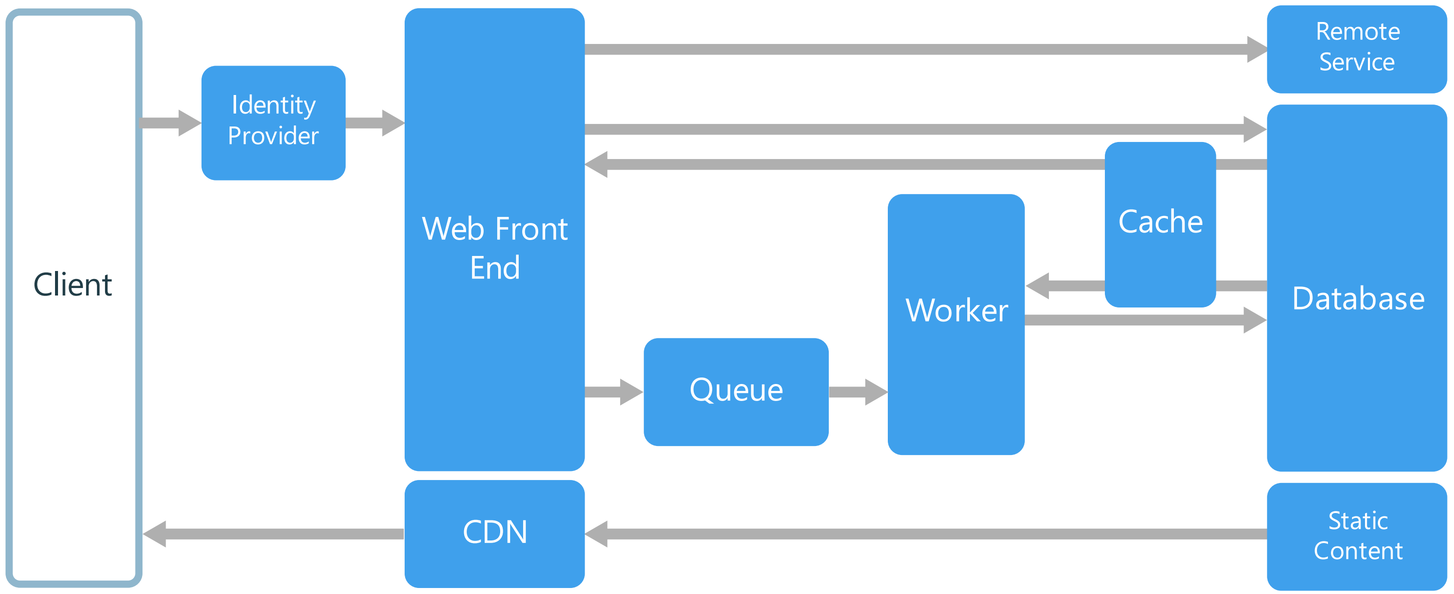
- Azure Well-Architected Security Pillar
  https://learn.microsoft.com/en-us/azure/well-architected/security/overview

- More design patterns
  https://learn.microsoft.com/en-us/azure/architecture/guide/
  https://learn.microsoft.com/en-us/azure/architecture/patterns/
  https://learn.microsoft.com/en-us/azure/architecture/browse/?filter=reference-architecture



slicesPP

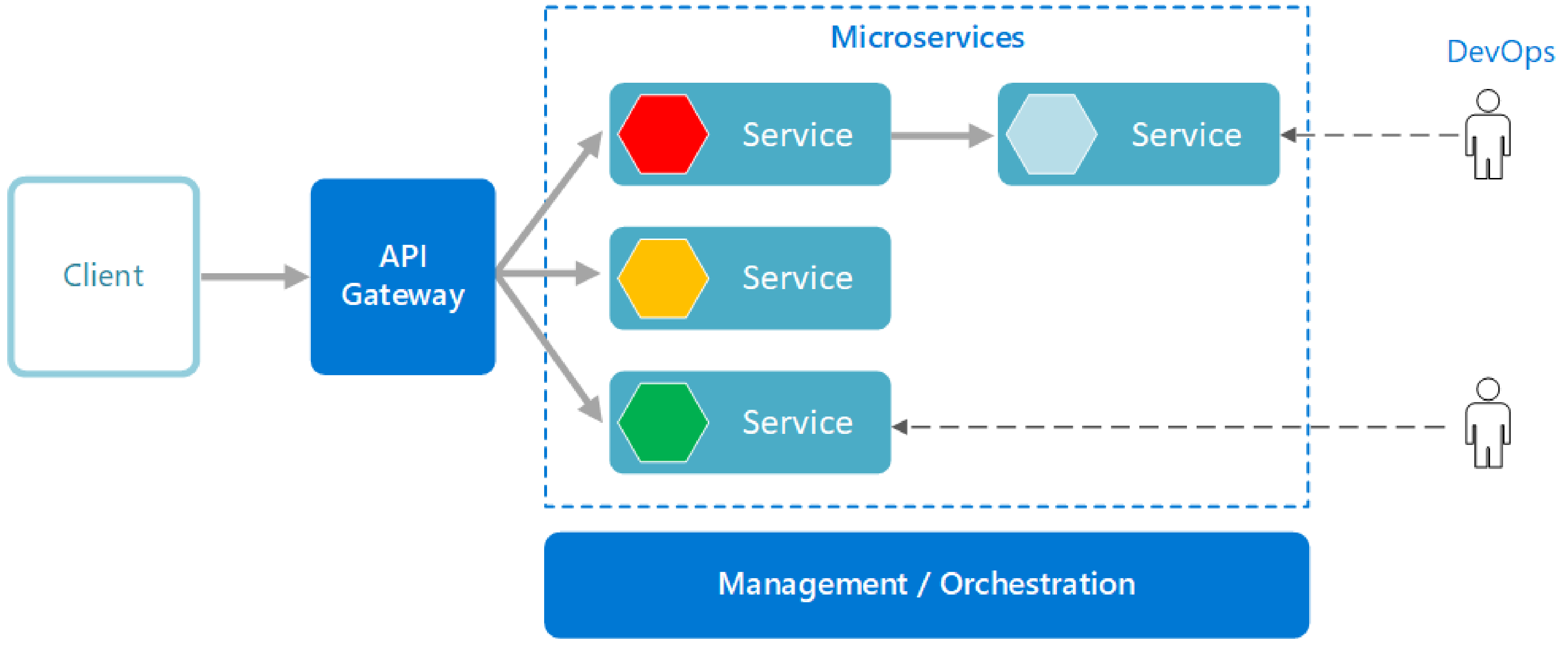
Web-Queue-Worker



- For a purely PaaS solution, consider a **Web-Queue-Worker** architecture.
- In this style, the application has a web front end that handles HTTP requests and a back-end worker that performs CPU-intensive tasks or long-running operations.
- The front end communicates to the worker through an asynchronous message queue.
- Web-queue-worker is suitable for relatively simple domains with some resource-intensive tasks.
- The use of managed services simplifies deployment and operations. But with complex domains, it can be hard to manage dependencies.
- The front end and the worker can easily become large, monolithic components that are hard to maintain and update.

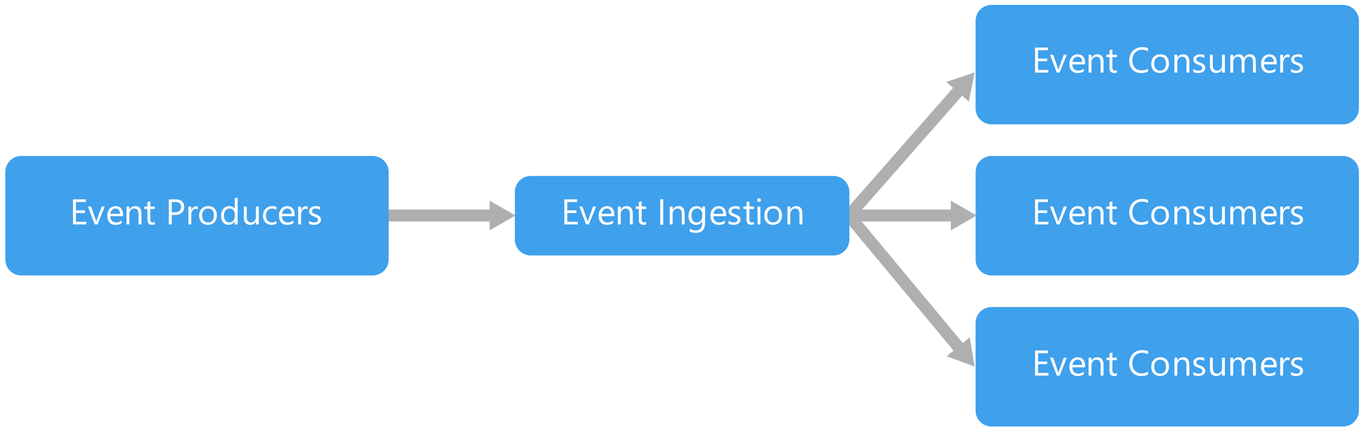
slicesPP

Microservices



- A microservices application is composed of many small, independent services. Each service implements a single business capability. Services are loosely coupled, communicating through API contracts.
- Each service can be built by a small, focused development team. Individual services can be deployed without a lot of coordination between teams, which encourages frequent updates.
- A microservice architecture is more complex to build and manage than either N-tier or web-queue-worker. It requires a mature development and DevOps culture.
- But done right, this style can lead to higher release velocity, faster innovation, and a more resilient architecture.

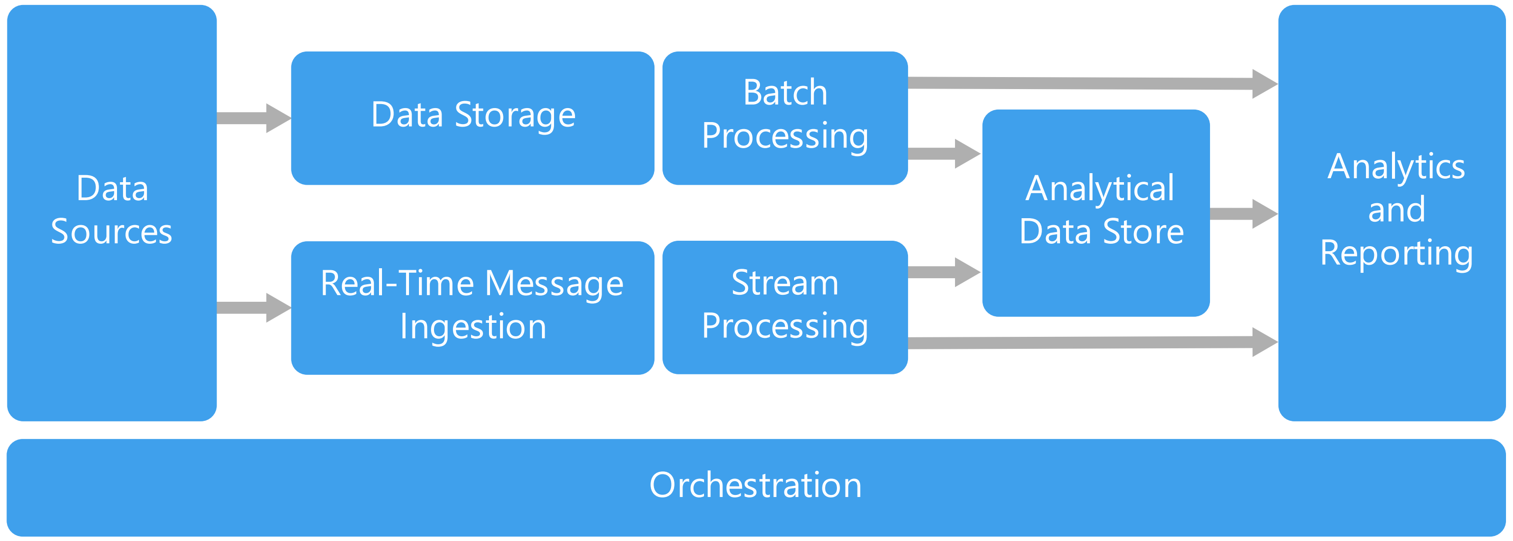
Event-Driven Architectures



- **Event-Driven Architectures** use a publish-subscribe (pub-sub) model, where producers publish events, and consumers subscribe to them.
- The producers are independent from the consumers, and consumers are independent from each other.
- Consider an event-driven architecture for applications that ingest and process a large volume of data with very low latency, such as IoT solutions.
- The style is also useful when different subsystems must perform different types of processing on the same event data.

Azure Big Data Architecture Style

https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data



- Azure includes many services that can be used in a big data architecture. They fall roughly into two categories:
- Managed services, including Azure Data Lake Store, Azure Data Lake Analytics, Azure Synapse Analytics, Azure Stream Analytics, Azure Event Hubs, Azure IoT Hub, and Azure Data Factory.
- Open source technologies based on the Apache Hadoop platform, including HDFS, HBase, Hive, Spark, Oozie, Sqoop, and Kafka. These technologies are available on Azure in the Azure HDInsight service.

FAIR Data Principles: Metadata Management (GO FAIR recommendations)

Findable:

- F1 (meta)data are assigned a **globally unique and persistent identifier**;
- F2 data are **described with rich metadata**;
- F3 metadata clearly and explicitly include the **identifier** of the data it describes;
- F4 (meta)data are **registered or indexed** in a searchable resource;

Interoperable:

- I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
- I2. (meta)data use vocabularies that follow FAIR principles;
- I3. (meta)data include qualified references to other (meta)data;

Accessible:

- A1 (meta)data are retrievable by their identifier using a standardized communications protocol;
  - A1.1 the protocol is open, free, and universally implementable;
  - A1.2 the protocol allows for an authentication and authorization procedure, where necessary;
- A2 metadata are accessible, even when the data are no longer available;

Reusable:

- R1 meta(data) are richly described with a plurality of accurate and relevant attributes;
- R1.1 (meta)data are released with a clear and accessible data usage license;
- R1.2 (meta)data are associated with detailed provenance;
- R1.3 (meta)data meet domain-relevant community standards;

slicesPP

FAIR from the technical point of view – Required Infrastructure functionality

- Findable
  - Metadata and PDI – infrastructure and tools
  - Metadata Registries and handles resolution, API
  - Policies and SLA
- Accessible
  - Repositories and data storage: infrastructure and management
  - Policy and access control: infrastructure and API management
  - Data access protocols
  - Usage Policy and Sovereignty
  - Data protection, compliance, privacy and GDPR
- Interoperable
  - Standard data formats
  - Metadata Registries and API
  - FAIR maturity level and certification
- Reusable
  - Data provenance and lineage
  - Preservation
  - Metadata, PID and API – linked or embedded into datasets

Technical implementation of the FAIR data principles requires comprehensive **data management infrastructure** to support

- **Data Storage**
- **Metadata Registries**
- Data publication
- Data discovery
- Linked data and data lineage (provenance)
- Multiple datasets access for analysis

slicesPP