Service Oriented Architecture

Building in the cloud

Lecture 3

Dr. Alessio Bonti


Designing SOA for the Cloud

What is Cloud Computing?

What is Cloud Computing?

CONCEPTS

Access on-demand

IT resources are accessed when needed and for the time needed.

Pay-as-you go

Virtual machines, applications and services (when not free) are priced based on a consumption model (i.e. per hour, per request or operation).

Ubiquity

Cloud computing resources are essentially Internet-based and virtually accessible from everywhere Internet is accessible.

What is Cloud Computing?

CONCEPTS

Heterogeneity

Cloud computing does not only offer virtual machines, but the entire IT stack is available "as a service": infrastructure, platforms, and applications.

Elasticity and reactivity

Cloud computing systems are often referred as elastically scalable, since they have the ability to grow and shrink base on need.

API

Web API are the "interaction surface" for most cloud computing services.

What is Cloud Computing?

DEFINITION

How to assemble these concepts together?

Cloud computing is a utility-oriented and Internet-centric way of delivering IT services on demand. These services cover the entire computing stack: from the hardware infrastructure packaged as a set of virtual machines to software services such as development platforms and distributed applications.

☑ On demand access	☑ Heterogeneity
☑ Pay as you go	☒ Elasticity and reactivity
☑ Ubiquity	☒ Web API

Why now?

AN INTERESTING CONJUNCTION

INTERNET CONNECTION BECOMES CAPILLARY

WEB 2.0 IS A MATURE APPLICATION PLATFORM

UTILITY COMPUTING BECOMES MATURE

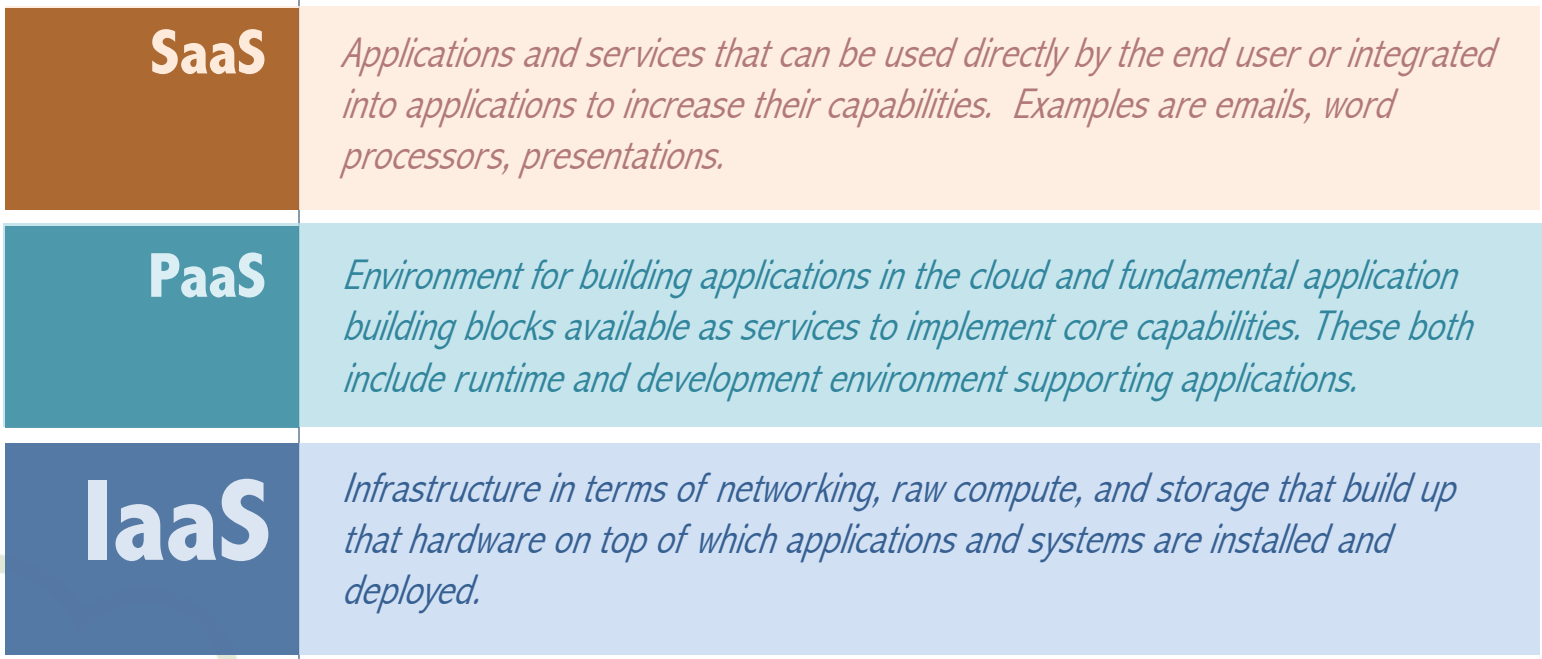
VIRTUALISATION IS ESTABLISHING AS COMMERCIALLY VIABLE

Explore the Cloud Stack Model

Infrastructure as a Service

CLOUD REFERENCE MODEL

At the bottom of the stack…



Infrastructure as a Service

DEFINITION

What is Infrastructure as a Service?

NIST

"The capability provided to the consumer is to provision processing, storage, networks and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)."

P. Mell, T. Grance, "NIST working definition on cloud computing." National Institute of Standard and Technology (NIST) http://csrc.nist.gov/publications/nistpubs/ 800-145/SP800-145.pdf

IaaS solutions

CHARACTERISTICS

IaaS management software

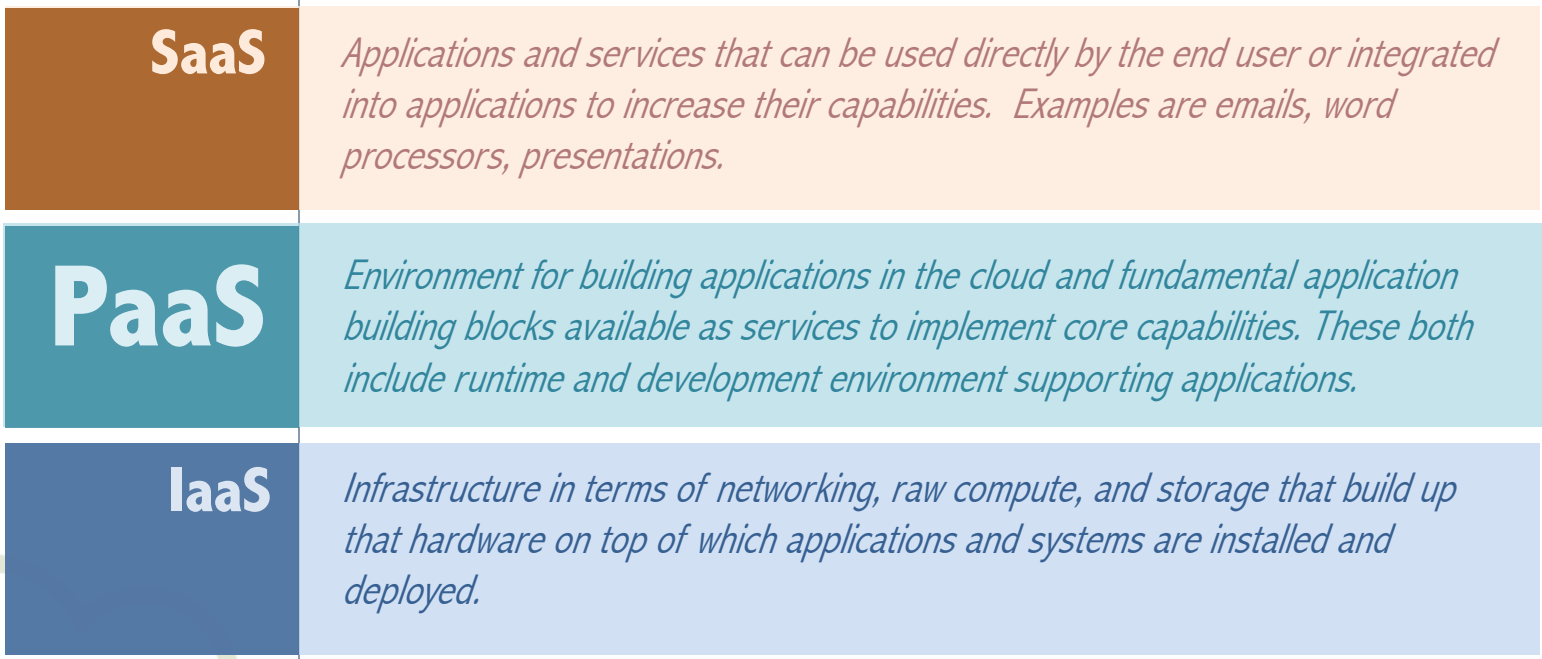
This is the central component for an IaaS solution and enables the provider to control, manage, and serve:

COMPUTE	Primarily in the form of virtual machines, but also physical (bare metal) servers, and recently lighter solutions such as *containers*.
STORAGE	Disk volumes either of fixed or changeable size and generic object storage for a more high-level type of storage.
NETWORK	Network equipment (load balancers, switches, routers and firewalls) emulated via software to isolate compute resources as if they were on different infrastructure.

Platform as a Service

CLOUD REFERENCE MODEL

In the middle of the stack…



Platform-as-a-Service

DEFINITION | At a glance

01

Platform as a Service identifies the collection of tooling, run-time and development environment, pluggable software componentry that put together, enables and facilitates the design, development, and deployment of cloud computing applications and systems.

02

Platform as a Service puts at the centre of the software life cycle management the cloud and the concept of serviceability of anything needed for that purpose.

Platform as a Service

DEFINITION | A view from NIST

NIST

"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

P. Mell, T. Grance, "NIST working definition on cloud computing." National Institute of Standard and Technology (NIST) http://csrc.nist.gov/publications/nistpubs/ 800-145/SP800-145.pdf

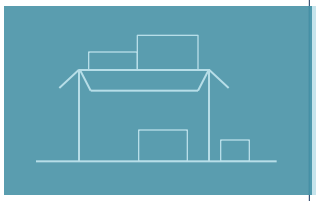
Platform as a Service

KEY FEATURES

What does a PaaS solution offer?

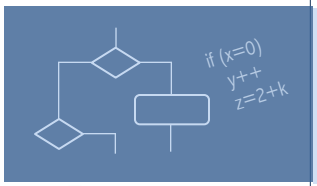
HIGH-LEVEL RUNTIME ENVIRONMENTS FOR RUNNING APPLICATIONS

These generally expose an interface that "lays above" the operating system and includes a specific development and technology stack.



READY TO USE COMPONENTS FOR APPLICATION INTEGRATION

Some examples are: key-value stores, in-memory caaches, job queues, etc..



SPECIFIC PROGRAMMING MODELS FOR DEVELOPMENT

These often target a specific technology stack or application class.

TOOLING SUPPORTING THE SOFTWARE DEVELOPMENT LIFE-CYCLE

Source control, build services, deployment services, dashboards for application monitoring.

Platform as a Service

CONCEPTS

Runtime environments

The key feature of a PaaS approach is that it provides an environment where application can be deployed to execute.

This environment is generally "above" the operating system and implies the use of a specified technology and development stack.

Implications:

01 MORE EMPHASIS ON DEVELOPMENT THAN INSTALLATION

02 THERE IS LESS CONTROL OVER THE ENTIRE STACK

03 LANGUAGES/STACKS ARE LIMITED TO THOSE OFFERED

Platform as a Service

CONCEPTS

Runtime environments

Some examples:

01

WEB APPLICATION DEVELOPMENT WITH JAVA

An environment that contains a pre-installed and configured stack containing a specific Java Enterprise Edition (JEE). Users will simply deploy the application code such as a WAR file, an EAR package, or a server package.

02

WEB APPLICATION DEVELOPMENT WITH NODE

An environment with Node.JS installed. Users will push the application source code and the platform will do "npm install" to complete the packaging before deployment.

Similar solutions can be found for other languages/platform.

Platform as a Service

CONCEPTS

Ready-to-use components

In a PaaS offering runtime environment are completed by a collection of services that simplify and accelerate application development.

These services cover both basic and advanced capabilities that define the application behaviour.

Observations:

01 DEVELOPMENT FOCUS MORE ON THE CORE LOGIC

02 APPLICATION DELIVERY BECOMES FASTER

03 APPLICATION DEVELOPMENT PATTERNS CHANGE

Platform as a Service

CONCEPTS

Ready-to-use components

Some examples:

01 DATABASE SERVICES (SQL AND NOSQL)

02 OBJECT AND KEY-VALUE STORES

03 APPLICATION CACHE SERVICES

04 JOB QUEUES, WORKFLOW SERVICES

05 MESSAGING AND NOTIFICATIONS SERVICES

06 DATA PROCESSING AND ANALYTICS SERVICES

07 SUPPORT SERVICES (TESTING, SCALABILITY, LOG MONITORING, . . .).

Platform as a Service

CONCEPTS

Programming models for development

In the PaaS space, the web application and services are the two predominant scenarios.

Different languages and stacks are made available to onboard prospective customers. Each of these options may come by enforcing a specific approach to application development.

Implications:

01 COMPLIANCE TO THE SUPPORTED "FRAMEWORK"

02 STANDARDISATION OF APPLICATION DEVELOPMENT

Platform as a Service

CONCEPTS

Programming models for development

Here are some examples of what type of languages and frameworks could be made available:

RUBY	RAILS vs. SINATRA
PYTHON	DJANGO vs. FLASK vs. PYRAMID

As shown it is not only about a specific programming language, but also about which (web) framework is packaged with the runtime for that language.

A choice of a framework both increases and reduces portability.

Platform as a Service

CONCEPTS

Tooling for the software development life cycle

As IaaS solutions focus more on streamlining administration tasks in relation to infrastructure, PaaS solutions focus on enriching the development experience, with specific support tools.

Together with runtime for hosting application, they offer a set of services to support application development life cycle.

01 THE CLOUD BECOMES CENTRAL TO APPLICATION DEVELOPMENT

02 APPLICATION DEPLOYMENT CAN BE AUTOMATED BY DEFAULT

03 THE INTEGRATION EFFORT IS REDUCED TO A MINIMUM

Platform as a Service

CONCEPTS

Tooling for the software development life cycle

Some of the key supporting features most commonly found in PaaS implementations are:

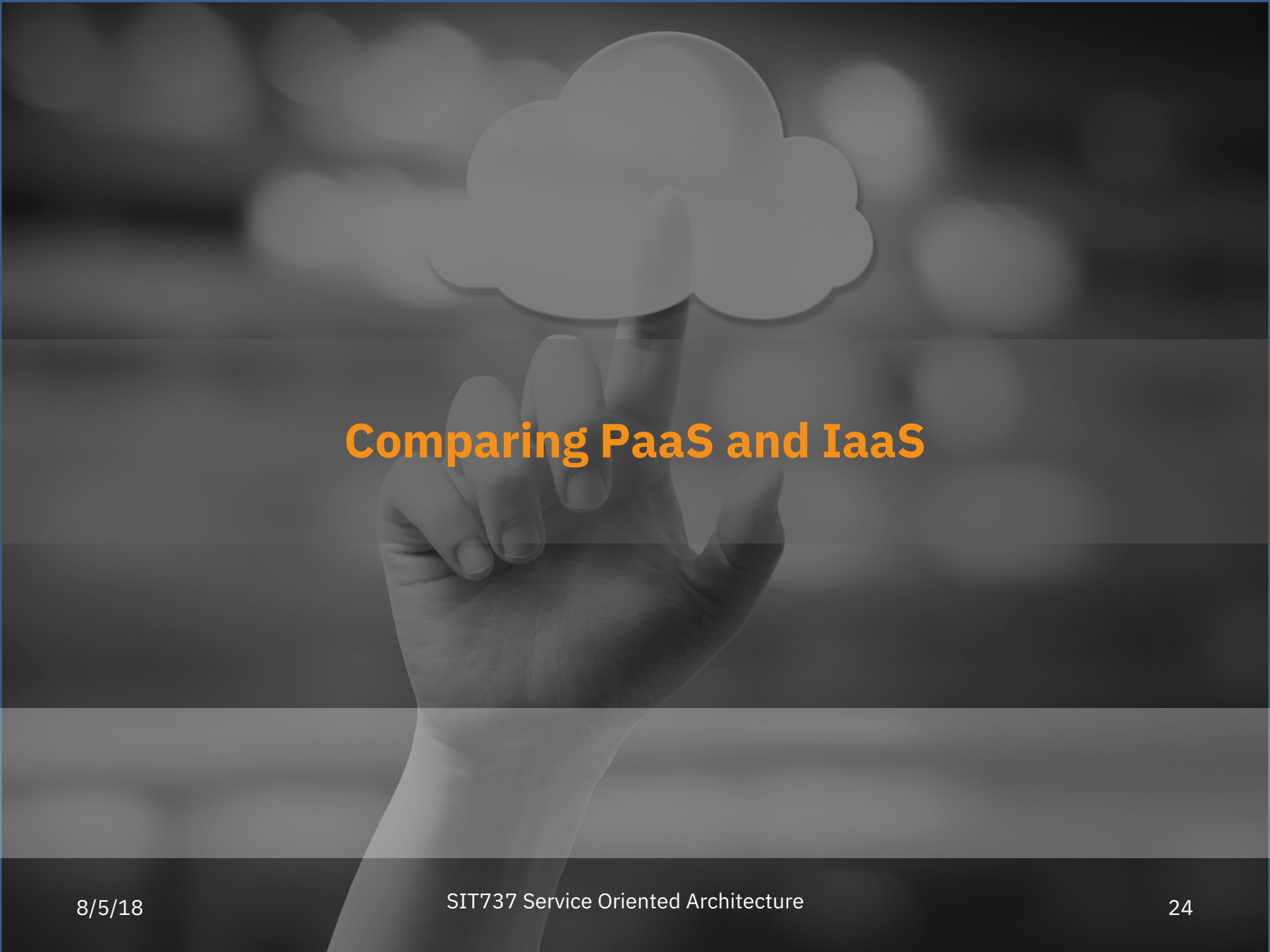
01 SOURCE CONTROL REPOSITORIES

02 CONTINUOUS INTEGRATION PIPELINES

03 (AGILE) PROJECT MANAGEMENT SUPPORT

04 SPACES MANAGEMENT (PROD, DEV, TEST, . . .)

05 INTEGRATED MONITORING DASHBOARDS, SERVICE CATALOGS



Comparing PaaS and IaaS

PaaS development

WHO IS A PaaS OFFERING FOR?

Meet the protagonist, the developer..

Each of the three layers that we identified in the cloud computing reference model "targets" a different profile. Whom is the PaaS for?

IaaS

IaaS solutions provide infrastructure on demand, therefore they support and facilitate those tasks that relate with the administration, deployment, and configuration of systems.

The role benefitting the most is the system administrator.

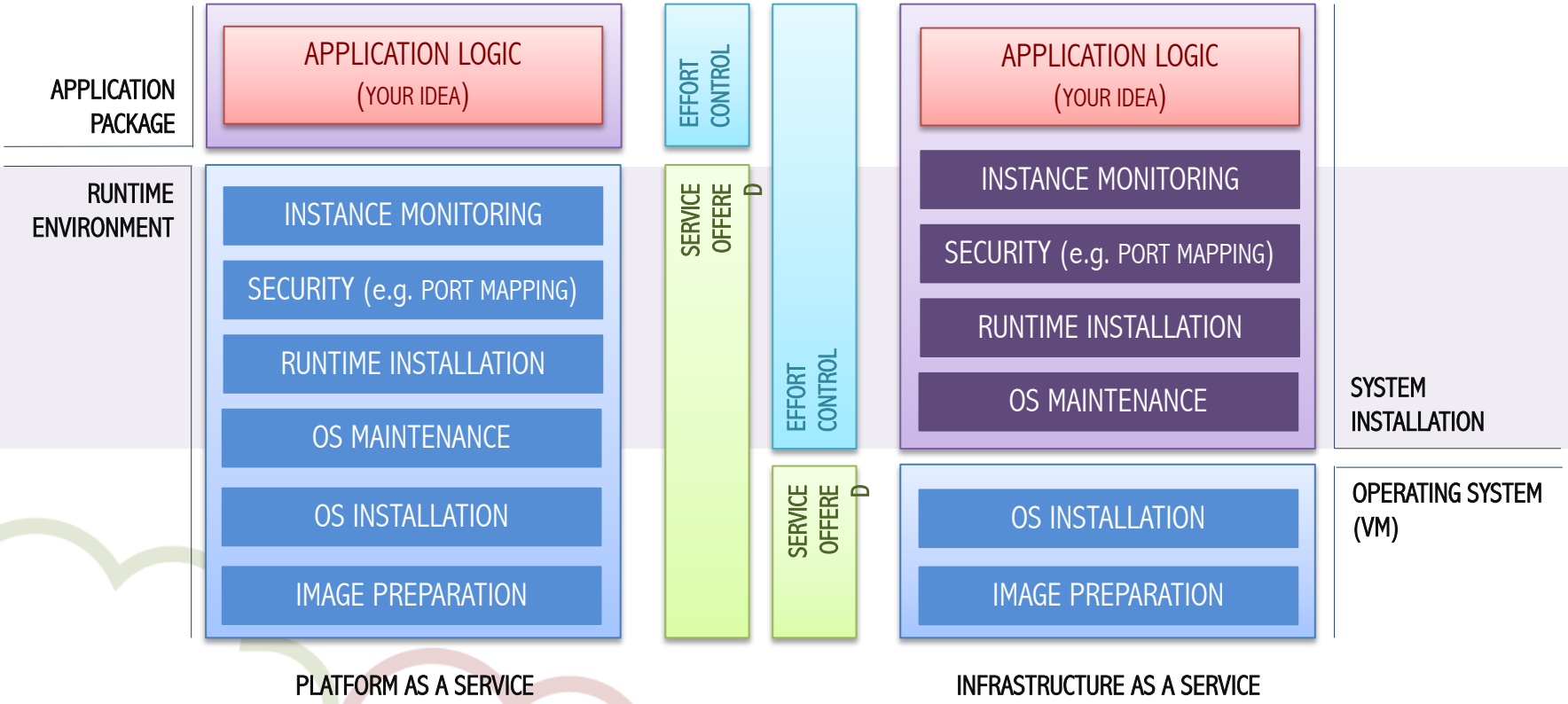
PaaS

PaaS solutions constitute a step upwards in the IT stack, and offer runtime environments and development support services. These are functions more related to the application.

Therefore, the role benefitting the most is the software engineer/developer.

PaaS development

COMPARISON | Comparing PaaS with IaaS



PaaS development

DEVELOPER'S PERSPECTIVE

How developers can benefit from PaaS?

TRADITIONAL DEVELOPMENT

- Select a development environment
- Install the required tooling
- Develop the application
- Select and install the accessory componentry
- Install and setup a build (CI) environment
- Select/install a deployment environment
- Deploy the application
- Maintain componentry and environment
- Get feedback and improve

Idea

CLOUD (PaaS) DEVELOPMENT

- Develop the application
- Connect the necessary componentry
- Configure and plug CI services
- Deploy with a click
- Get feedback and improve
- Monitor application health

PaaS development

DEVELOPER'S PERSPECTIVE

When it is not appropriate to use PaaS ?

01 NEED FOR CONTROLLING THE INFRASTRUCTURE (OS AND HARDWARE)

PaaS solutions abstract away anything below the application runtime environment, and therefore no access is given to those layers (implications on security and data policies).

02 NO LOGIC IS DEVELOPED, BUT A SINGLE FUNCTION IS REQUESTED

PaaS solutions provide environment to develop, integrated capabilities and hosting the resulting system. If we're looking for a single function (e.g. notifications) SaaS offering are more appropriate.

03 NEED TO BE VENDOR LOCK-IN FREE

As we move up to the stack the service offering gets more specialised. In the absence of a de-jure standard regulating PaaS offerings, vendor lock-in could be a higher risk.

PaaS development

DEVELOPER'S PERSPECTIVE

Benefits and advantages

Let's look at the software development process…

SWEET SPOT

- Inception
- Planning
- Design
- Implementation
- Testing
- Deployment
- Maintenance

Core Logic: Your Idea
(what the software does)

Accessory Logic
(what it needs for doing what it does)
Web Server, Storage, Authentication, ….

PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS

Let's walk through the steps of developing on a PaaS

IDEA	*What new cool thing I want to develop?* *What needs addresses?*
1 DEVELOPMENT EFFORT CHARACTERISATION	*How much "work" do I expect to do?* *How many features are required for the MVP? ..and how complex are they?*
2 SELECTION OF CORE AND SUPPORTING FUNCTIONS	*Which basic capabilities are required?* *Are there any components/services I can integrate and use?*
3 CHARACTERISATION OF THE LOGIC TO DEVELOP	*Do we need to implement (from scratch) any core feature?* *What is the nature of the logic to be implemented?*
4 SELECTION OF THE IMPLEMENTATION STACK	*What is the most appropriate language/platform I can use?* *Which one I know better?*
5 SELECTION OF THE TARGET PaaS PLATFORM	*What is the PaaS offering, and which type of approach among those provided, is most appropriate fot the job?*

PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS

1 DEVELOPMENT EFFORT CHARACTERISATION

This phase is about estimating the amount of work needed to implement our idea into an application/service.

A good lead for the estimate is done by identifying what is often referred as the "Minimum Viable Product (MVP)".

The MVP identifies the minimum set of features that make your idea sellable/interesting for a first release.

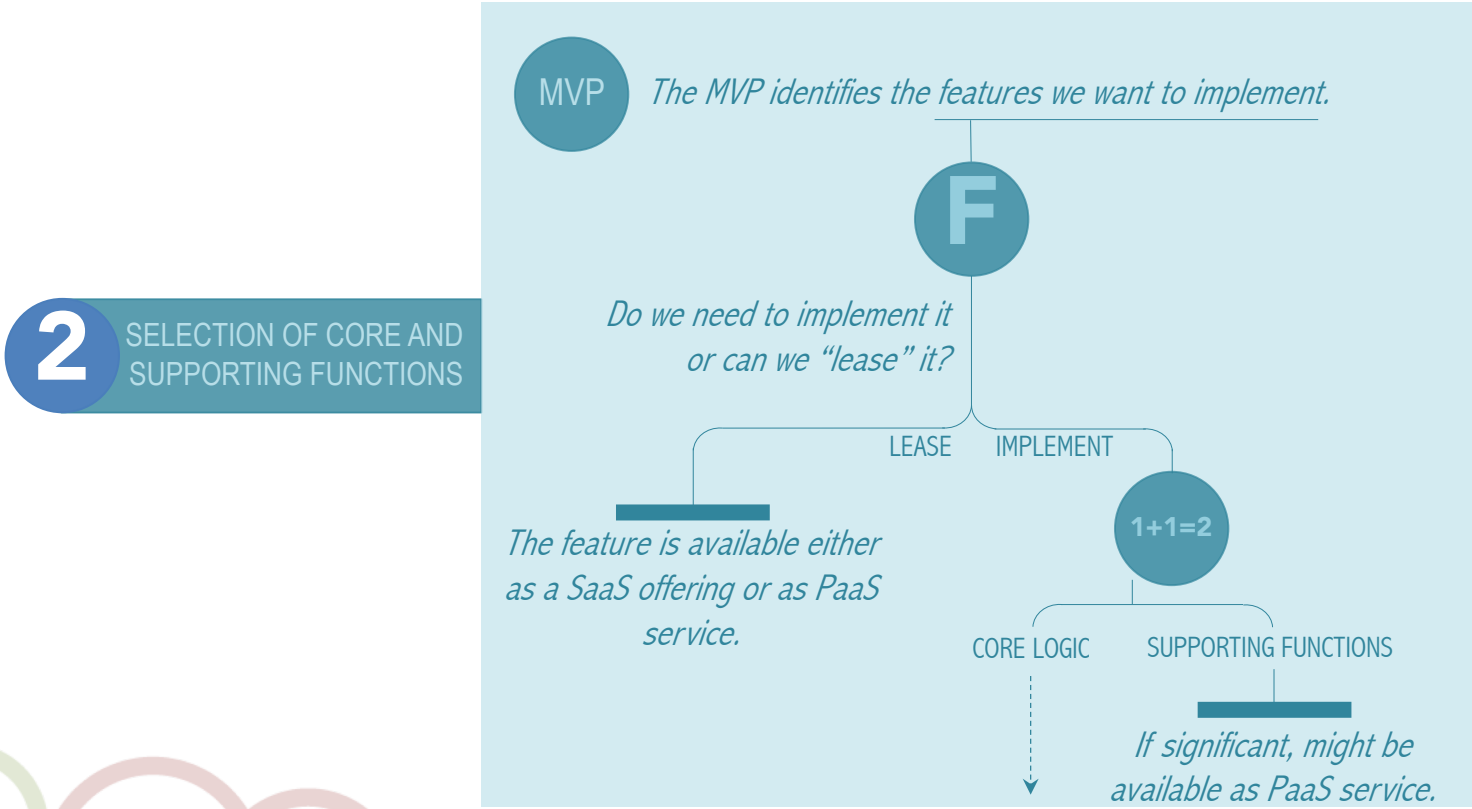
The rough estimate of the work effort for each feature and their interdependencies provides us with an idea of the work to be completed.

This is the first checkpoint in the process. Is it worth for? How quickly we can do it? How much it could cost ?

PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS

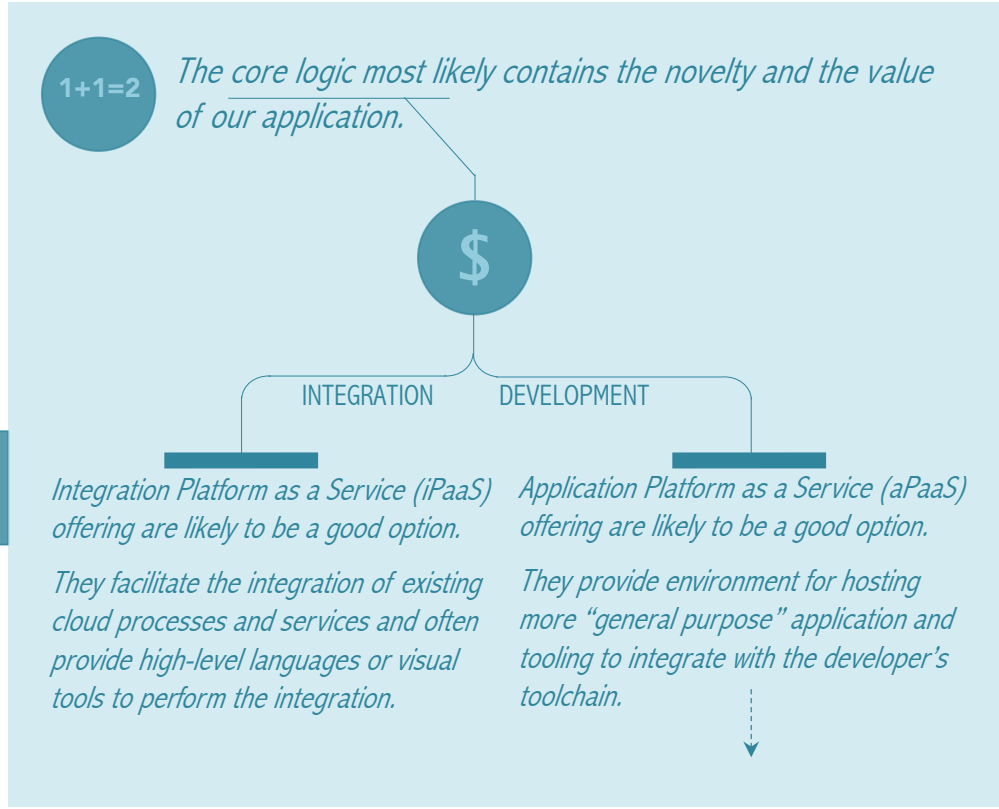


PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS

3 CHARACTERISATION OF THE LOGIC TO DEVELOP



PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS

4 SELECTION OF THE IMPLEMENTATION STACK

If the core logic is primarily expressed through development, the choice of a stack/framework is the next step.

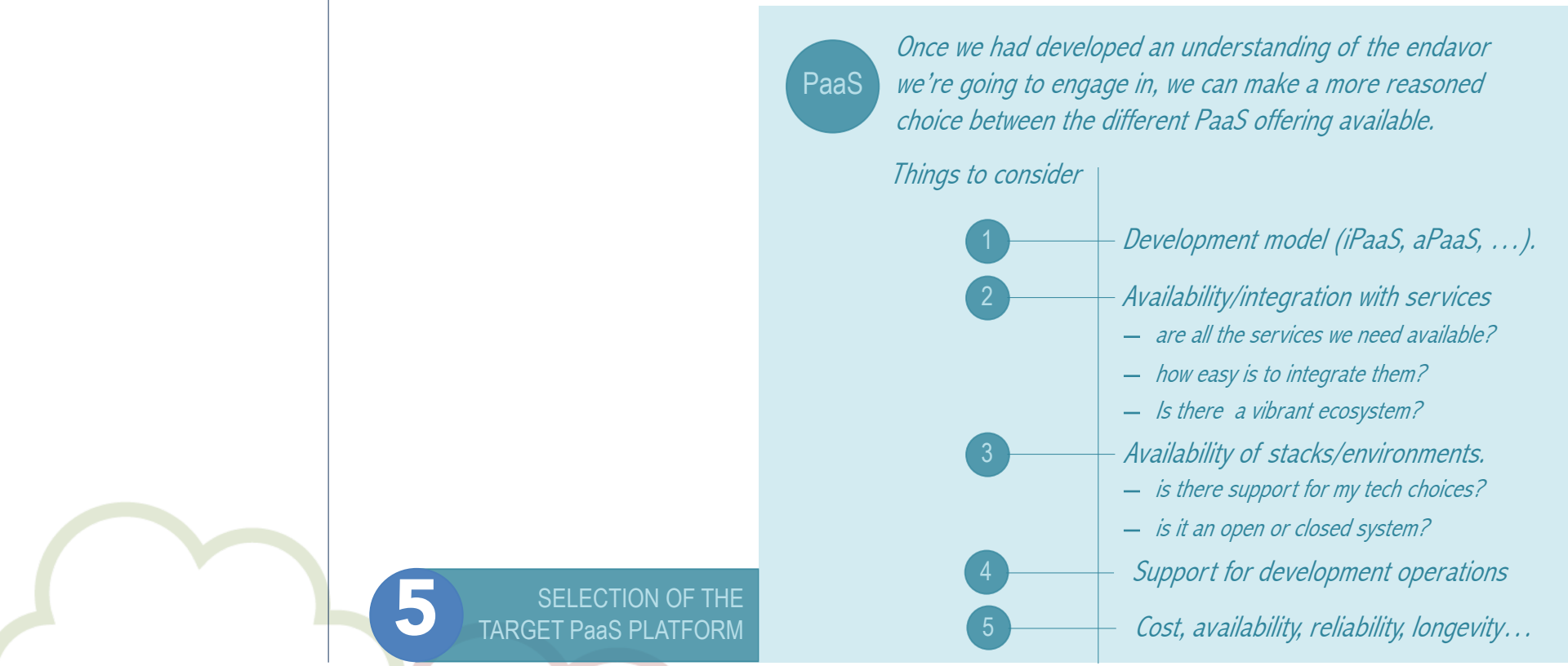
How to choose between the different options?

1. *What is the language that we know better?*
2. *Which framework features will facilitate most of the logic to implement?*
3. *Which libraries and integration does it offer?*
4. *What is the scale/size of what we plan to build (next releases)?*

PaaS development

DEVELOPMENT PROCESS

Cloud development with PaaS



PaaS development

DEVELOPMENT PROCESS

Development support services

For Application Platform as a Service (aPaaS) solutions, the management of application code is essential and (often) exposed to the developer that can control and tune it at wish.

This is realised through:

01 SOURCE CODE CONTROL INTEGRATION

02 CONTINUOS INTEGRATION SERVICES

03 INTEGRATION WITH THE DEVELOPER'S LOCAL ENVIRONMENT

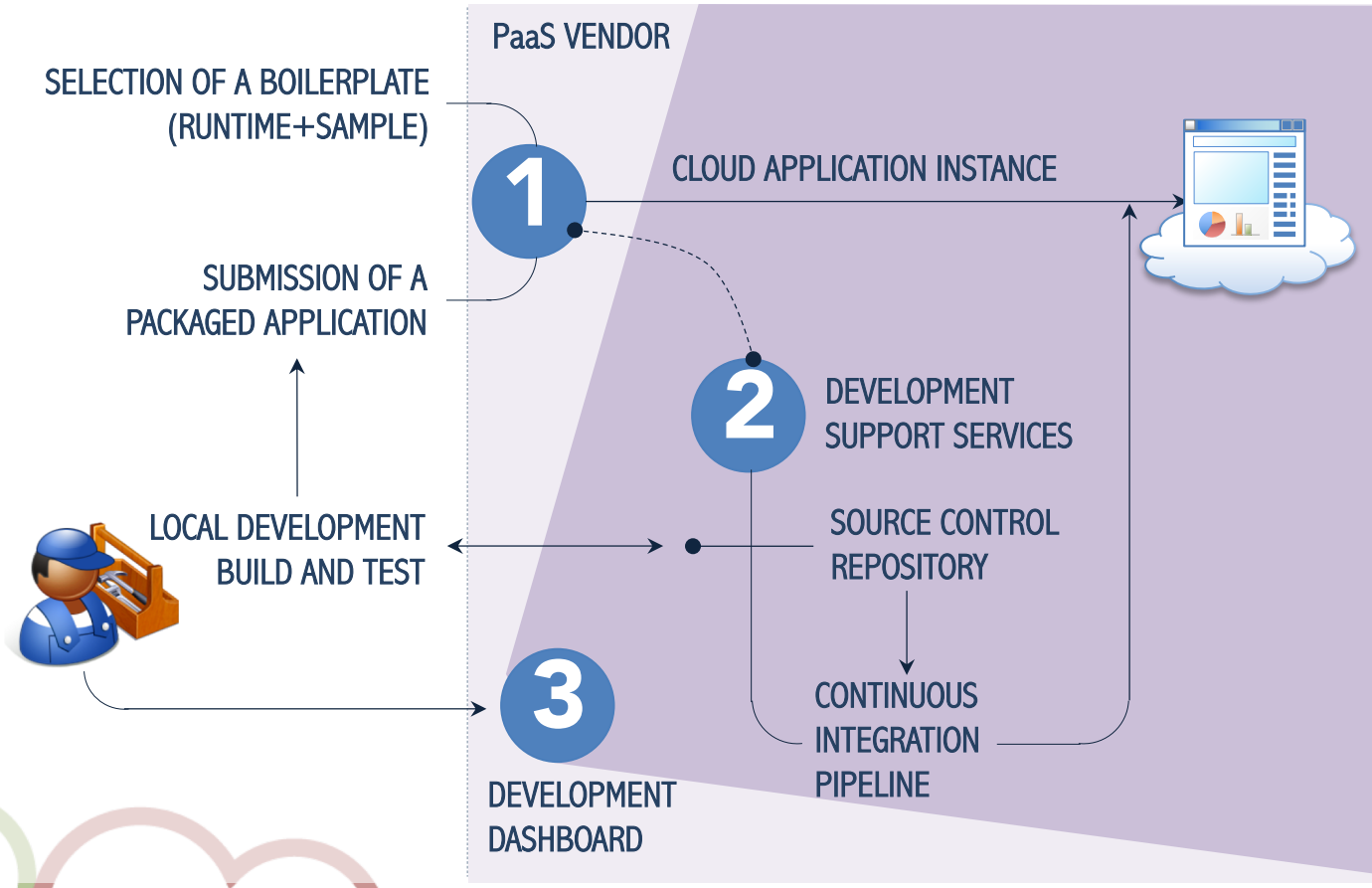
04 DEPLOYMENT (AND VERSIONING) SUPPORT

05 SUPPORT FOR DEVELOPMENT TEAMS

PaaS development

DEVELOPMENT PROCESS

Exploring the developer experience




Business Case

Business case

THE STARTUP

Meet TomorrowTech

TomorrowTech is a new startup focused on exploring new technologies and characterised by a strong drive for innovation.

Tom Falks, the company's visionary founder has just came back from the IoT World Forum, where he had a preview of new IoT technology that will be in the market in 2-3 months.

Tom had a brilliant idea about a new mobile application that could bring added value to this technology and calls his staff for an urgent meeting to define a strategy to pursue this new opportunity.

Business case

THE PLAN

Inspecing the requirements

At the meeting, Tom illustrates his idea and briefly highlights some of the key elements for a winning strategy:

01 DELIVERTY TIMELINE IS 2 MONTHS

02 FREQUENT RELEASES TO KEEP ENGAGED THE VENDOR

03 FAST PROTOTYPING: FROM CONCEPT TO LIVE IN DAYS

04 APPLICATION(S) MANAGEMENT IS ESSENTIAL

05 ABILITY TO PLUG INTO A RICH ECOSYSTEM OF CAPABILITIES

06 VISIBILITY END-TO-END FOR THE DEVELOPMENT LIFE CYCLE

Business case

THE PLAN

… and the additional requirements

Being a new startup, Tom and team have not made any serious commitments in infrastructure, but this project may require a considerable amount of resources at least for the next 2 months, later the success of the application will shape the need.

07 ABILITY TO VARY INFRASTRUCTURE COMMITMENTS OVER TIME

Moreover, it is likely that the application will require the use of different stacks to leverage at best the team skill sets cut down the time to market.

08 MULTIPLE STACKS/LANGUAGES NEEDED FOR IMPLEMENTATION

Business case

THE PLAN

… and the additional requirements

Tom's team is an avid consumer of (and contributor to) open source technologies, and tapping into the community's wisdom is seen as an essential element for success.

Moreover, Tom would like to bring this concept further create an environment that facilitates the sharing of ideas, contribution to the codebase so that other members of the company can easily participate to the development.

09 TOOLING FOR CODE SHARING AND TEAMWORK SUPPORT

Business case

THE PLAN

Technology selection

The team decides that cloud computing technologies are essential to the successful completion of the project. There is some doubt about which specific offering could be the best fit.

?

Can you articulate which cloud computing offering is more effective for this business case, and which characteristics of the selected offering help Tom and team meet the highlighted requirements?

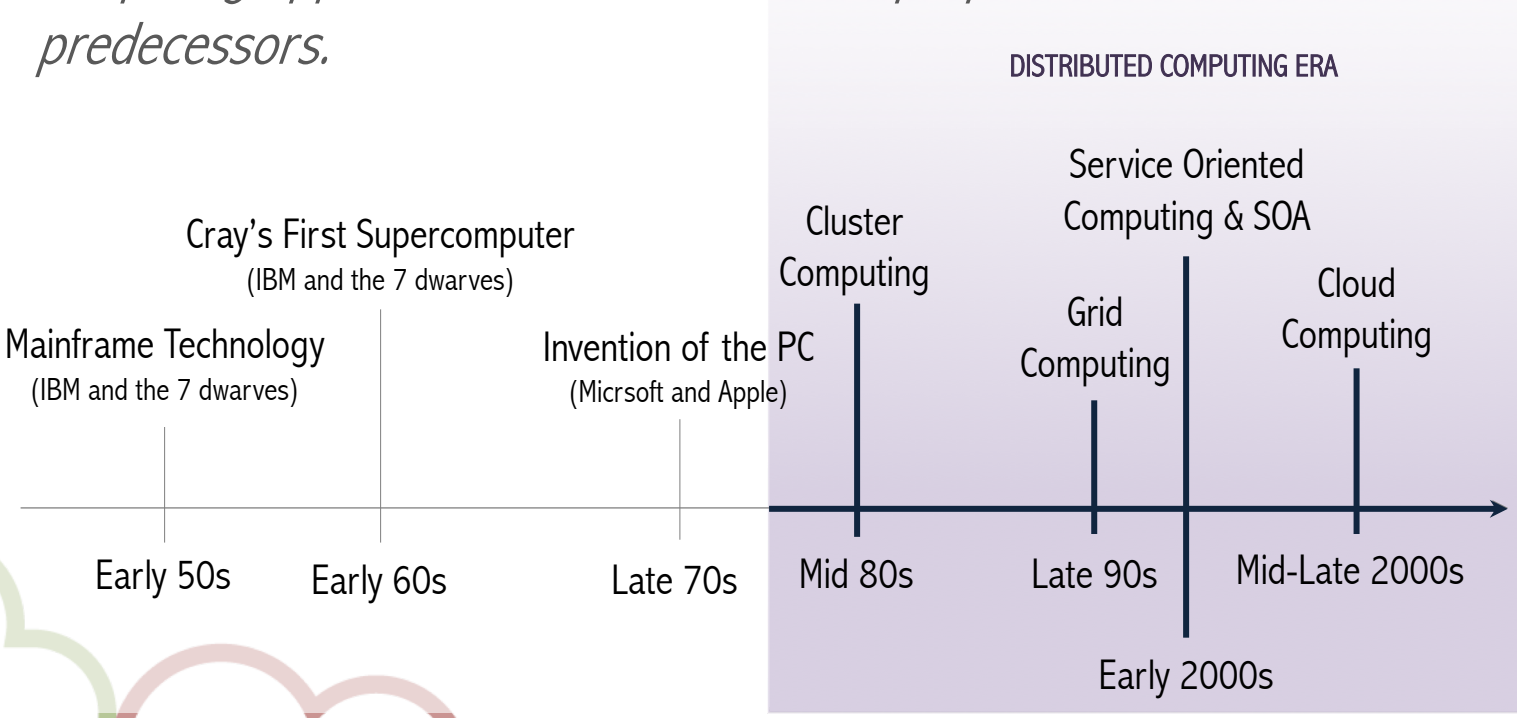
Distributed Computing Primer

Distributed Computing Primer

HISTORY

Looking at things in perspective…

Cloud computing constitutes the latest evolution of distributed computing approaches and exhibits most properties of its predecessors.



What is “Distributed Computing” ?

DEFINITION

Collective working as one system

Tanembaum et al.

“A distributed system is a collection of independent computers that appears to its users as a single coherent system.”

Communicating through messages

Colouris et al.

“A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages.”

Characterisation

CONCEPT The ultimate purpose is…

CONNECTING USERS WITH RESOURCES

DESIGN GOALS Four goals drives their design:

HETEROGENEITY	*Different components must be allowed to interoperate.*
OPENNESS	*Interaction should occurr through well defined interfaces and protocols.*
TRANSPARENCY	*Resources should be accessible despite their location, number, or other users accessing it.*
AVAILABILITY	*The failure of a system component should not prevent to prevent the system to operate.*

Characterisation

PROPERTIES

Concurrency

Distributed systems are a collection of independent computing nodes. They are inherently concurrent and there is no global clock governing their operation. This poses new challenges especially in relation to coherence and transparency of access.

Scalability

Scalability is a desired property for distributed systems so that they can gracefully support an increased demand and/or an increased number of resources / operations.

Approaches, Paradigms, & Models

MIDDLEWARE

Cluster Computing

Constitute the first incarnation of distributed systems. Became popular with the availability of the PC and low-end servers. Characterised by a collection of homogeneous machines and high bandwidth interconnect. The system was inherenly additive.

Grid Computing

These originated as network of clusters connected at national or international scale. Users have access through a service portal and submit jobs for execution. This approach emphasises even more the concept of computing on demand (i.e. utility computing).

Approaches, Paradigms, & Models

MODELS

Client Server

This application model became widely popular with the advent of the World Wide Web as it was particularly suited to support the interaction based on a single server and multiple clients.

The server can be organised into one or more tiers according to the complexity of the service delivered. The client can be either thin or fat according the amount of logic embedded in it.

Peer-to-Peer (P2P)

This model is characterised by every node being able to serve either as a client or server. The most common implementation of this model are P2P network (e.g. file sharing P2P).

Approaches, Paradigms, & Models

MODELS

Notification Buses

This model is primarily characterised by the presence of a bus, which represent a "logic highway" thorugh which messages are exchanged among components. Each node (or some nodes) can push messages to the bus (e.g. raise events) and others can listen to the bus for specific messages.

Publish / Subscribe

This model is a further refinement of a notification bus, and organises it into topics (i.e. channels) and subtopics, which enable nodes to selectively publish to and subscribe to. In this model, the infrastructure provides built-in services to perform the routing and message selection.

Approaches, Paradigms, and Models

PROGRAMMING

Remote Procedure Call

Distributed Object Frameworks

Service Oriented Computing

Service Oriented Computing is one of the most popular approaches to distributed systems and applications design and implementation.

They allow modelling a distributed system as a collection of interacting services that interact through very well defined interaces and protocols.

The concept of service not only permeates the implementation of distributed systems, but also the design and implementation of single node applications (e.g. the operating system).

Service Orientation

CONCEPTS

What is a Service?

A service is a computing entity that is based on the following four tenets (Don Box, Microsoft):

BOUNDARIES ARE EXPLICIT

A service-oriented application is often spread across different organisation and thrust authorities. This could make their invocation costly, this is why the invocation is made explicit (note: compare with disributed objects RMI, Remoting, …).

SERVICES ARE AUTONOMOUS

Services encapsulate one or more functionalities, which can act as building block for building complex systems. They do not belong to a single system but are designed to be integrated to multiple systems. Hence, they are autonomous and minimum assumption can be made when designig systems consuming services.

Service Orientation

CONCEPTS

What is a Service?

(... continued)

SERVICES SHARE SCHEMAS AND CONTRACTS, NOT CLASS OR INTERFACE DEFINITIONS

A service advertises a contract describing the structure of messages it can send and/or receive and additional constraint—if any—on their ordering. This facilitates interoperations across heterogeneous systems, but does require stability of schemas.

SERVICES COMPATIBILITY IS DETERMINED BY POLICIES

Schemas and contracts define the structural compatibility of services, which can be validated via machine based techniques. Semantic compatibility is defined by policies, which define capability and requirement of a service. These are general defined as expressions that must hold true to enable the normal operation of a service.

Service Orientation

CONCEPTS

Service Oriented Architectures (SOA)

SOA is the architectural style supporting service orientation[1]. It defines the architecture of a sytem as a collection of interacting services.

SOA defines the roles, practices and principles that should be followed to implement service oriented enterprise applications.

ROLES	*Service consumer and service provider*
PATTERNS	*Service orchestration and service coreography*
PRINCIPLES	*Standardised service contract, loose coopling, abstraction, reusability, autonomy, lack of state, discoverability, composability.*

[1] Object Management Group - http://www.opengroup.org/subjectareas/soa

Service Orientation

TECHNOLOGIES

How do we implement SOA?

SOA can leverage the basic plumbing defined by RPC abstraction, but in reality two other technologies have been used:

WEB SERVICES

WS constituted the prominent technology in early 2000 to implement SOA. They leverage Internet technologies for building distributed applications. Several factors made WS the technology of choice:

1. *They allow interoperability across diverse platforms and programming languages.*
2. *They are based on well known and accessible standards (XML, SOAP, WSDL, …).*
3. *They provide intituitive and simple ways to integrate and compose different systems.*
4. *They provide support of enterprise features (e.g. service discovery, and metering).*

Service Orientation

TECHNOLOGIES

How do we implement SOA?

(… continued)

REST SERVICES

REST stands for Representational State Transfer and it is an architectural model for describing the interactions of distributed components through a set of primitive operations and the concept of hyperlinked resources.

It is most popular (and perhaps the only one) implementation is over the HTTP protocol by leveraging HTTP verbs (HEAD, POST, PUT, GET, DELETE, and OPTIONS).

Compared to web services REST introduces less overheads in the communication. Its leaner and simpler design is what made such approach preferred over WS, and it now constitutes the interface of choice for most cloud computing services.

Approaches, Paradigms, & Models

TRENDS

Application Service Providers (ASPs)

As service orientation became popular application service providers (ASPs) started to proliferate. ASPs introduced the capability to host and make available on behalf of third parties applications and services by leveraging data centers.

Trough multi-tenancy multiple applications and services were hosted within the same data center, thus enabling ASPs to apply economies of scale by offering this service to multiple customers and recovers from the running and management costs. Customer paid a subscription to get their application hosted and served.

This constitute the very first instance of the economic model that is at the basis of many cloud computing offerings.