BITS Pilani

Cloud Computing

SEWP ZG527

Agenda

- ❖ Capacity Management Recap
- ❖ Multi Tenancy in the Cloud
- ❖ 4 Models of Multi Tenancy

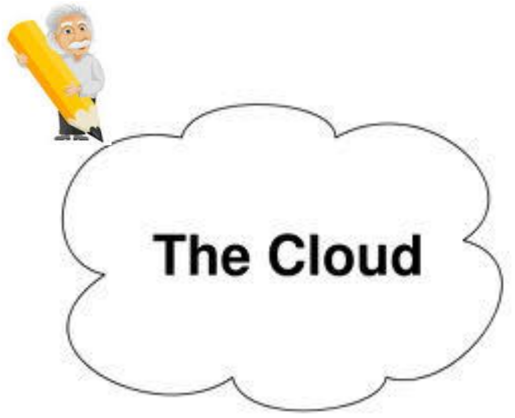
innovate achieve lead

BITS Pilani

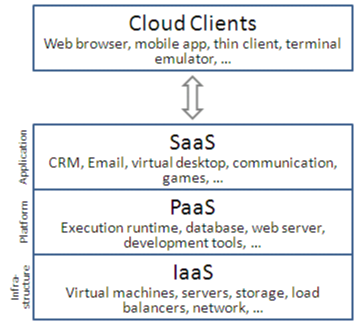
Pilani|Dubai|Goa|Hyderabad

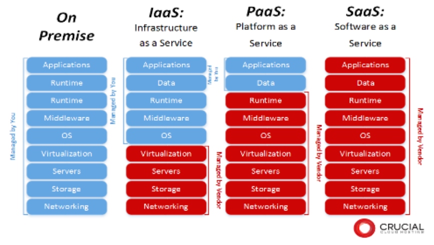
Recap

Anatomy of a Cloud Ecosystem



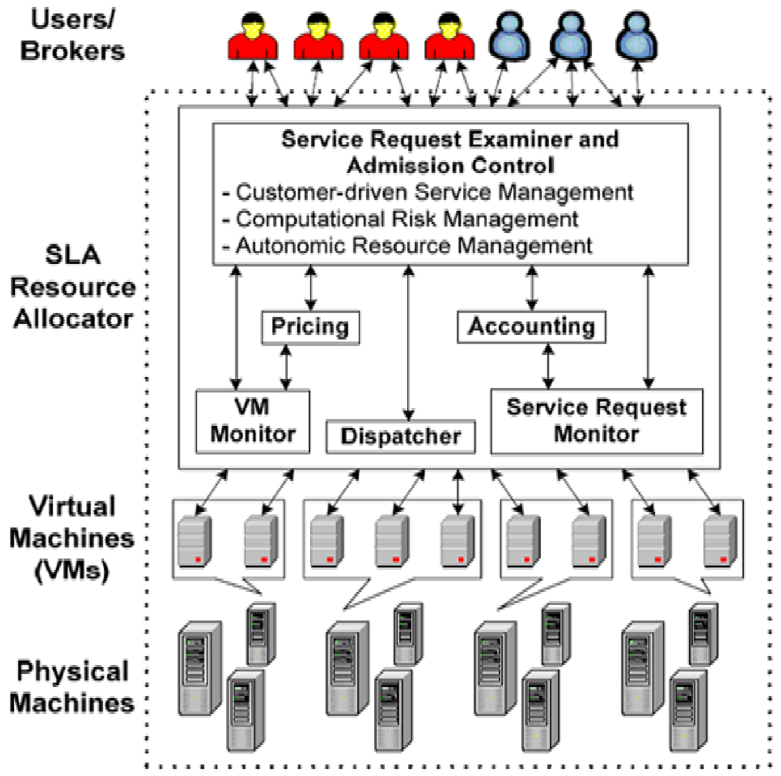
Is it So?????


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure



CLOUD requires managing…… But what will you manage?

- Cloud **Distributed environment**
  - With large scale of systems to manage
  - Support of multi-tenancy
  - Management to maintain SLAs


Users/ Brokers
SLA Resource Allocator
Service Request Examiner and Admission Control
- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management
Pricing
Accounting
VM Monitor
Dispatcher
Service Request Monitor
Virtual Machines (VMs)
Physical Machines

Importance of Capacity Management

When **Capacity** is **Too Little** (under capacity)

- Application Sizing not performed
- Controlled by limits
- Lower usage costs
- Service Levels affected – by how much?
- High impact on business?
  - Loss of service
  - SLA breach penalties

When **Capacity** is **Too Much** (over capacity)

- Unlimited access to resources
- Service Levels unaffected
- Costs – higher resource usage, software licensing
- Impacts on other services
  - Poor performance
  - Wasted resources (multi virtual CPU (*vSMP*) & single-threaded applications)
  - VM Sprawl
  - Increased pressure to manage virtual resources

- **Gartner – *"most organizations overprovision their infrastructure by at least 100%"***

What are the Challenges?

Capacity on Demand- Too Little, Too Much or Just Right?

Automation and Control

Security Protecting your loved ones‟

Scalability- In / Up or Out?

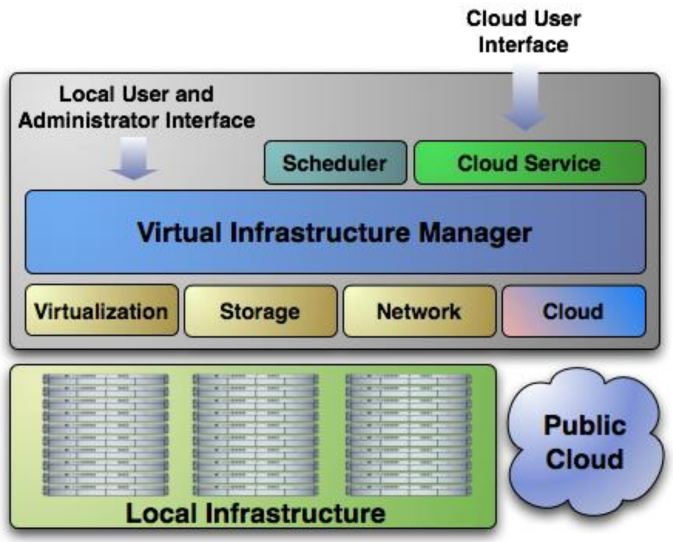
Old Habits – The Potential Risks

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

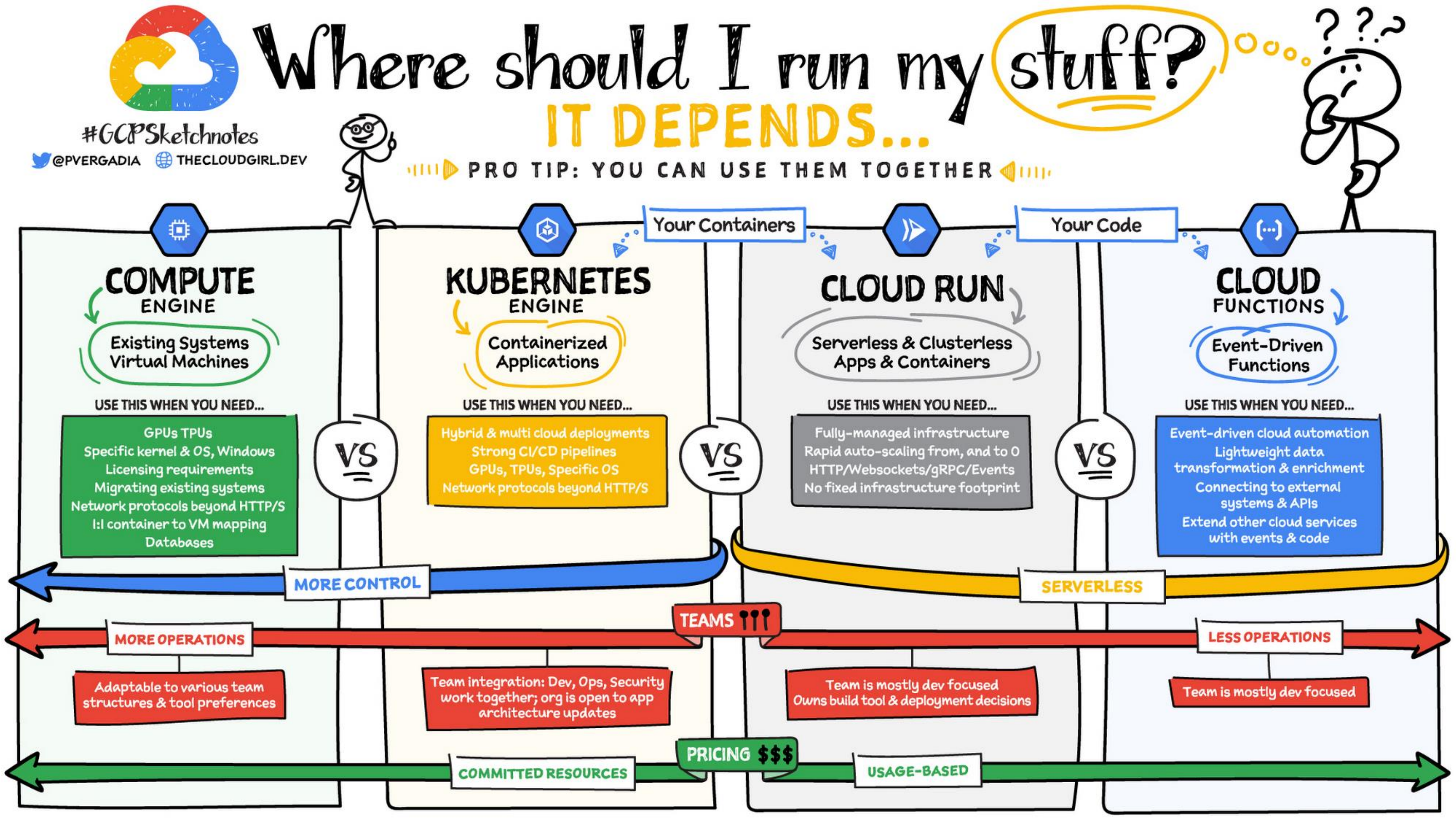
- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.
- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

What is a Virtual Infrastructure Manager?

- A VIM runs on top of a hypervisor in a virtualized environment. The hypervisor allocates and manages virtual machines. The VIM deals with the allocation of resources in the virtual infrastructure. They include computational resources (processors), storage, and network resources. Virtual infrastructure management allows the allocation to happen based on current requirements, rather than being statically allocated.
- The VIM carries out several tasks:
- Allocating resources in accordance with traffic engineering rules.
- Support for defining operational rules.
- Definition of hub-to-facility mapping.
- Providing information for provisioning virtual infrastructure orchestration (VIO).



What is a Cloud Workload



Where should I run my stuff?

IT DEPENDS...

#GCPSketchnotes

@PVERGADIA THECLOUDGIRL.DEV

PRO TIP: YOU CAN USE THEM TOGETHER

Your Containers

Your Code

COMPUTE ENGINE

Existing Systems Virtual Machines

USE THIS WHEN YOU NEED...

GPUs TPUs
Specific kernel & OS, Windows
Licensing requirements
Migrating existing systems
Network protocols beyond HTTP/S
1:1 container to VM mapping
Databases

VS

KUBERNETES ENGINE

Containerized Applications

USE THIS WHEN YOU NEED...

Hybrid & multi cloud deployments
Strong CI/CD pipelines
GPUs, TPUs, Specific OS
Network protocols beyond HTTP/S

VS

CLOUD RUN

Serverless & Clusterless Apps & Containers

USE THIS WHEN YOU NEED...

Fully-managed infrastructure
Rapid auto-scaling from, and to 0
HTTP/Websockets/gRPC/Events
No fixed infrastructure footprint

VS

CLOUD FUNCTIONS

Event-Driven Functions

USE THIS WHEN YOU NEED...

Event-driven cloud automation
Lightweight data transformation & enrichment
Connecting to external systems & APIs
Extend other cloud services with events & code

MORE CONTROL

SERVERLESS

TEAMS

MORE OPERATIONS

LESS OPERATIONS

Adaptable to various team structures & tool preferences

Team integration: Dev, Ops, Security work together; org is open to app architecture updates

Team is mostly dev focused
Owns build tool & deployment decisions

Team is mostly dev focused

PRICING $$$

COMMITTED RESOURCES

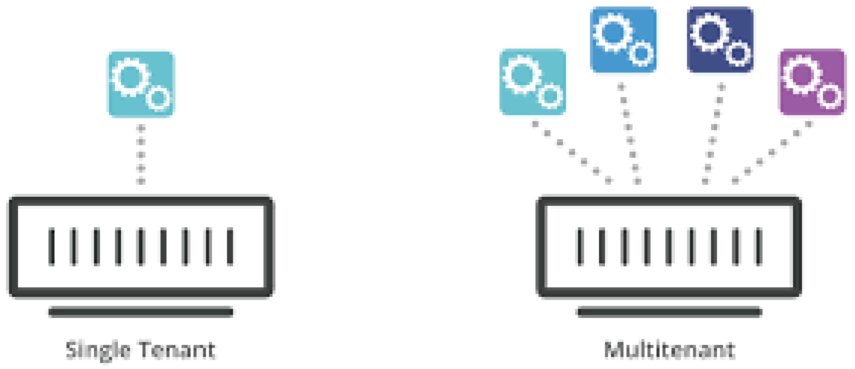
USAGE-BASED

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Multi Tenancy



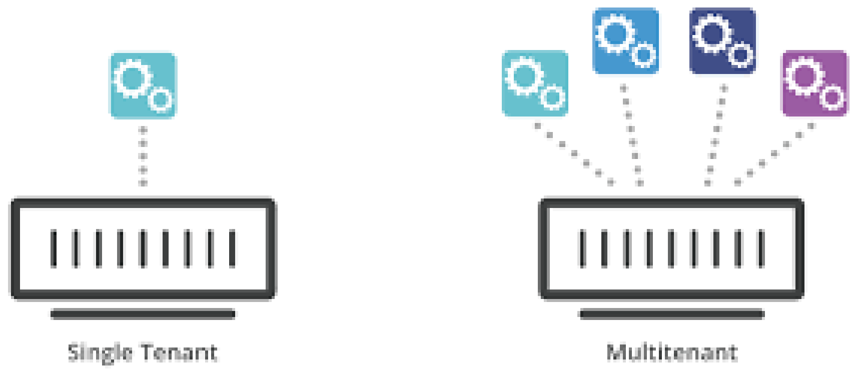
What is Multi Tenancy?

Multi-tenancy is an architectural pattern

A single instance of the software is run on the service provider’s infrastructure

Multiple tenants access the same instance.

In contrast to the multi-user model, multi-tenancy requires customizing the single instance according to the multi-faceted requirements of many tenants.

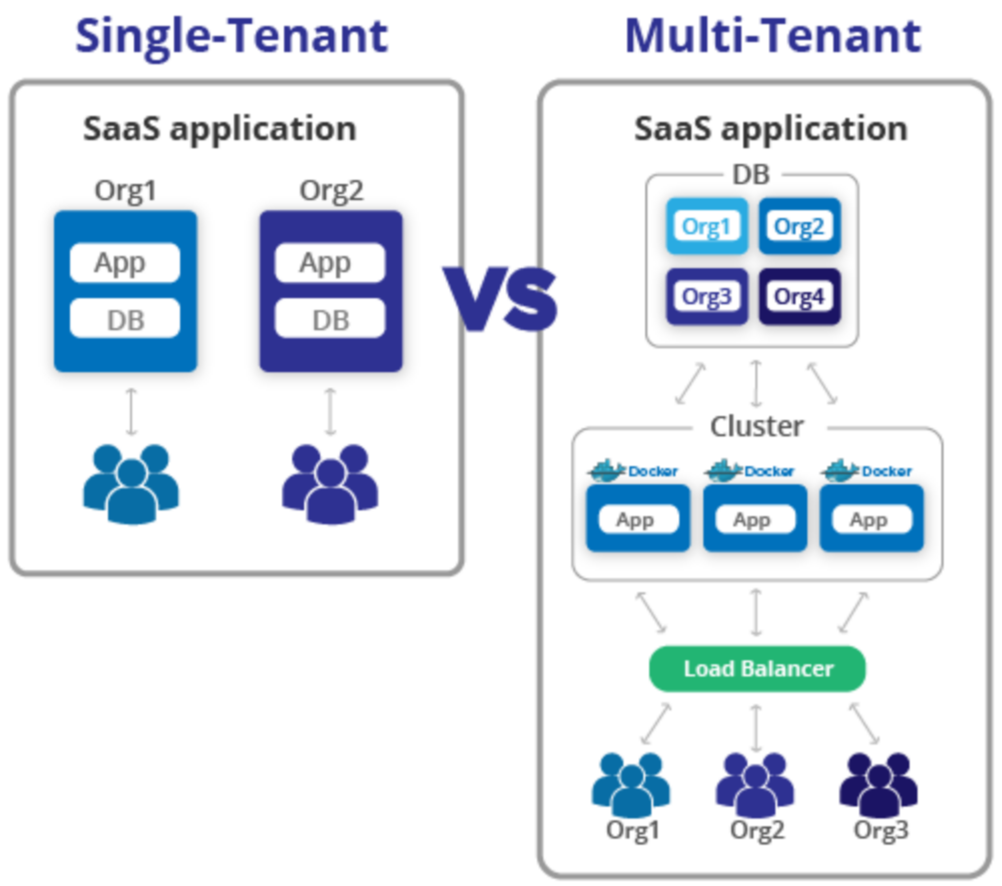


What is Multi Tenancy?

- Multi-tenancy is an architecture in which a **single instance** of a **software application** serves **multiple customers**. Each customer is called a tenant. Tenants may be given the ability to customize some parts of the application, such as color of the user interface (UI) or business rules, **but they cannot customize the application's code**.

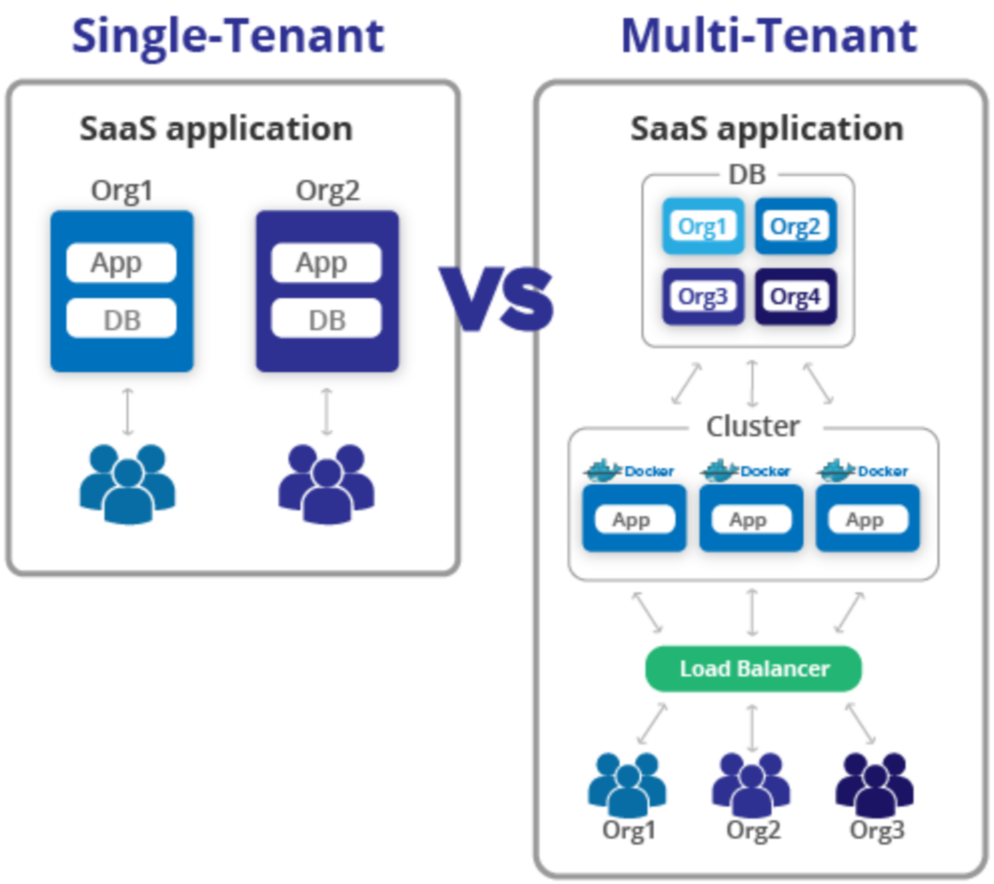
Some Facts

- In the multi tenant architecture, the application is redesigned to handle the resource sharing between the multiple tenants.
- For example, SalesForce.com (service provider) hosts the CRM application using their infrastructure.
- A company who wants to use this hosted CRM application for their business is the customer and the employees of the companies to whom the company provides privileges to access the CRM application are the actual users of the application

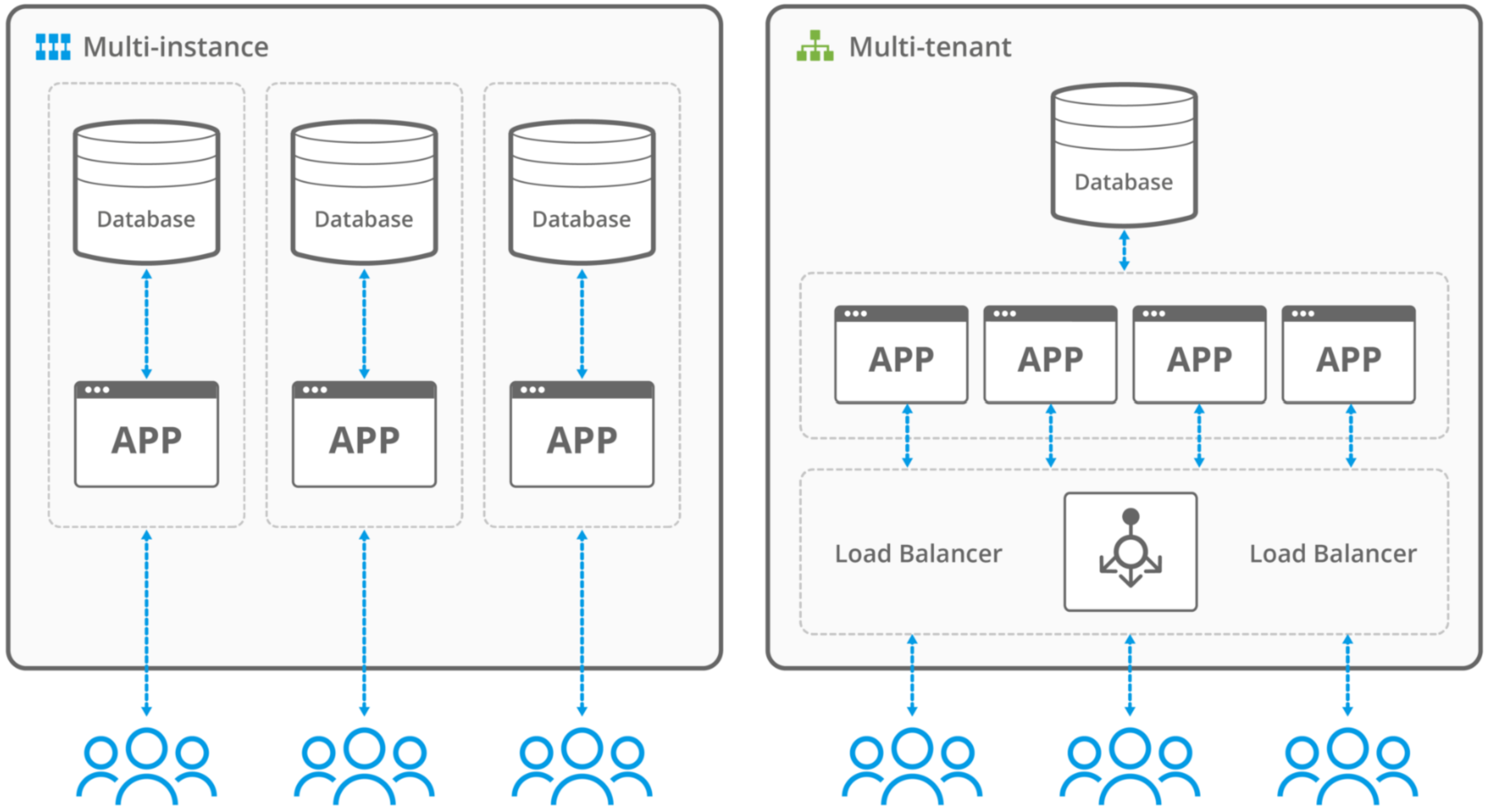


Some Facts

- With this **architecture**, data, configuration, user management, tenant specific functionality etc are shared between the tenants.
- MT contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.
- In virtualization, the user is given the illusion that he owns the complete infrastructure on which application is running through concept of virtual machine.
- The hypervisor plays important role to achieve the separation between the multiple users.



Multi Tenant vs Multi Instance



IAAS – MT Model

In (IaaS), a single physical or virtual infrastructure is shared among multiple tenants. Each tenant is provided with its own logical environment, which includes compute resources such as virtual machines (VMs), storage, networking, and other infrastructure components. There are several ways to implement a multi-tenant deployment model for IaaS:

1. **Virtual machine isolation**: In this model, each tenant is provided with a set of virtual machines that are completely isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.
2. **Network isolation**: In this model, each tenant is provided with its own virtual network that is isolated from other tenants. This approach provides better performance and efficiency than virtual machine isolation, but may require more complex network management.
3. **Resource pool isolation**: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.
4. **Hybrid model**: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

PAAS – MT Model

1. **Application-level isolation**: In this model, each tenant's application is isolated from other tenants' applications at the application level. This can be achieved through containerization or virtualization, and may also involve isolating the application's data and configuration settings.
2. **Tenant-level isolation**: In this model, each tenant is provided with a dedicated instance of the platform, which is isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.
3. **Resource pool isolation**: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.
4. **Hybrid model**: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

SAAS – MT Model

There are several ways to implement a multi-tenant deployment model for SaaS:

1. **Database-level** isolation: In this model, each tenant's data is stored in a separate database schema, which provides complete isolation and security. This approach can be efficient and scalable, but may require complex database management.
2. **Application-level isolation**: In this model, each tenant's data is kept separate at the application level, which may involve partitioning data and configuration settings. This approach can be more flexible and easier to manage than the database-level isolation model, but may be less secure.
3. **Hybrid model**: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require complete data isolation, while others may be willing to share some components of the application.

Benefits

Cost Savings –

- An application instance usually incurs a certain amount of memory and processing overhead which can be substantial when multiplied by many customers, especially if the customers are small.
- As the single instance is shared between multiple tenants this cost overhead can be segregated between multiple tenants.

Data aggregation–

- In non MT architecture, the data for different customers will be located in different database schemas and pulling information from all of them can be a very cumbersome task.
- In MT architecture, instead of collecting data from multiple data sources, with potentially different database schemas, all data for all customers is stored in a single database schema.
- Thus, running queries across customers, mining data, and looking for trends is much simpler.

Benefits

Release management –

- MT simplifies the release management process.
- In a traditional release management process, packages containing code and database changes are distributed to client desktop and/or server machines.
- These packages then have to be installed on each individual machine.
- With the multitenant model, the package typically only needs to be installed on a single server. This greatly simplifies the release management process.

Disadvantages

Complexity –

- Because of the additional customization complexity and the need to maintain per-tenant metadata, multitenant applications require a larger development effort.

Risks-

- At the same time, multi-tenancy increases the risks and impacts inherent in applying a new release version.
- As there is a single software instance serving multiple tenants, an update on this instance may cause downtime for all tenants even if the update is requested and useful for only one tenant.
- Also, some bugs and issues resulted from applying the new release could manifest in other tenants' personalized view of the application.
- Because of possible downtime, the moment of applying the release may be restricted depending on time usage schedule of more than one tenant.

Characteristics of MT Architecture

Customization –

- Multi-tenant applications are typically required to provide a high degree of customization to support each target organization's needs. Customization typically includes the following aspects:
  - ❖ **Branding**: allowing each organization to customize the look-and-feel of the application to match their corporate branding (often referred to as a distinct "skin").
  - ❖ **Workflow**: accommodating differences in workflow to be used by a wide range of potential customers.
  - ❖ **Extensions to the data model**: supporting an extensible data model to give customers the ability to customize the data elements managed by the application to meet their specific needs.
  - ❖ **Access control**: letting each client organization independently customize access rights and restrictions for each user.

Quality of service

- Multitenant applications are expected to provide adequate isolation of security, robustness and performance between multiple tenants which is provided by the layers below the application in case of multi-instance applications

MT- Different Levels

- Implementing the highest degree of resource sharing for all resources may be prohibitively expensive in development effort and complexity of the system.
- A balanced approach, where there is fine grained sharing of resources only for important resources, may be the optimum approach.
- The four levels of multi-tenancy are described in the following list for any given resource in a cloud system, the appropriate level could be selected
    - **Custom instances**
    - **Configurable instances**
    - **Configurable, multi-tenant efficient instances**
    - **Scalable, configurable, multi tenant efficient resources**

MT- Different Levels

Custom instances

- Lowest level of MT
- Each customer has own custom version of application
- Different versions of application are running differently
- Extremely difficult to manage as needs dedicated support for each customer

Configurable instances

- Same version of application is shared between the customers with customizations specific to each customer
- Different instances of same application are running
- Supports customization like logos on the screen, tailor made workflows
- Managing application is better that custom instances approach as only one copy needs to be managed
- Upgrades are simple and seamless

MT- Different Levels

Configurable, multi-tenant efficient instances

- Same version of application is shared between the customers through a single instance of application
- More efficient usage of the resources
- Management is extremely simple

Scalable, configurable, multi tenant efficient resources

- All features of level 3 are supported.
- Application instances are installed on cluster of computers allowing it to scale as per demand.
- Maximum level of resource sharing is achieved.
- Example, Gmail

Security in MT

- The key challenge in multi-tenancy is the secure sharing of resources. A very important technology to ensure this is authentication.
- Clearly each tenant would like to specify the users who can log in to the cloud system. Unlike traditional computer systems, the tenant would specify the valid users, but authentication still has to be done by the cloud service provider.
- Two basic approaches can be used: **a centralized authentication system** or a **decentralized authentication system** .
- In the **centralized system**, all authentication is performed using a centralized user data base. The cloud administrator gives the tenant's administrator rights to manage user accounts for that tenant. When the user signs in, they are authenticated against the centralized database.

Security in MT

- In the **decentralized system**, each tenant maintains their own user data base, and the tenant needs to deploy a federation service that interfaces between the tenant's authentication framework and the cloud system's authentication service.
- **Decentralized authentication** is useful if **single sign-on is important**, since centralized authentication systems will require the user to sign on to the central authentication system in addition to signing on to the tenant's authentication system.
- However, **decentralized authentication systems** have the **disadvantage that they need a trust relationship between the tenant's authentication system and the cloud provider's authentication system**. Given the self-service nature of the cloud (i.e., it is unlikely that the cloud provider would have the resources to investigate each tenant, and ensure that their authentication infrastructure is secure), centralized authentication seems to be more generally applicable.

Multi Tenancy: Resource Sharing

- Two major resources that need to be shared are storage and servers.
- The basic principles for sharing of these resources are described first.
- This is followed by a deeper discussion that focuses on the question of how these resources can be shared at a fine granularity, while allowing the tenants to customize the data to their requirements.

Sharing storage resources:

- In a multi-tenant system, many tenants share the same storage system. Cloud applications may use two kinds of storage systems: File systems and databases, where the term database is used to mean not only relational databases, but NoSQL databases as well.
- The discussion is focused on sharing data for different users in a database. The focus is also on multi-tenant efficient approaches where there is only one instance of the database which is shared among all the tenants

Resource Sharing Approaches

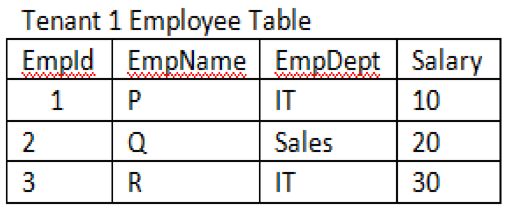
STORAGE Sharing Scenarios in MT

Dedicated tables per tenant

- We discuss only the approach where only one instance of database is shared between the tenants.
- Each tenant has separate copy of table
- Only tenant is given the privileges to access these tables, no other can access it
- Customizations can be easily added to the tenant's tables

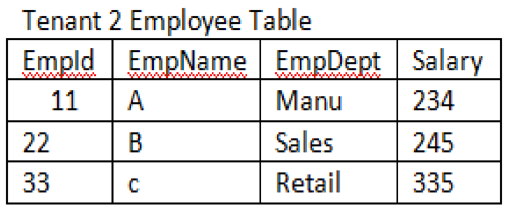
Shared table approach

- Tables are shared between the tenants
- Needs to isolate between the tenants data in different rows using the unique tenant id assigned to each tenant
- More space efficient than dedicated table approach
- Needs more compute resources as it needs to use view to make join to retrieve tenant specific data
- Metadata table needs to be maintained for tenant's information
- Managing customization is difficult



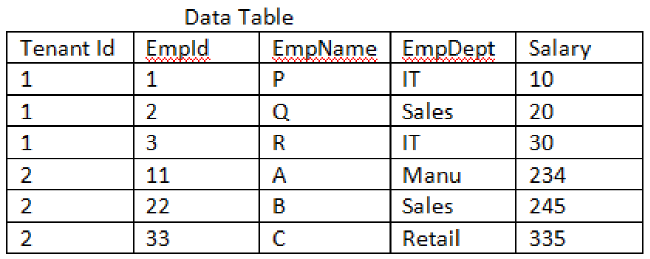
Tenant 1 Employee Table

EmpId	EmpName	EmpDept	Salary
1	P	IT	10
2	Q	Sales	20
3	R	IT	30



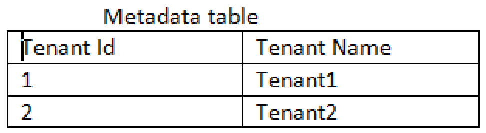
Tenant 2 Employee Table

EmpId	EmpName	EmpDept	Salary
11	A	Manu	234
22	B	Sales	245
33	c	Retail	335



Data Table

Tenant Id	EmpId	EmpName	EmpDept	Salary
1	1	P	IT	10
1	2	Q	Sales	20
1	3	R	IT	30
2	11	A	Manu	234
2	22	B	Sales	245
2	33	C	Retail	335



Metadata table

Tenant Id	Tenant Name
1	Tenant1
2	Tenant2

Support for Customization

Customization:

- It is important for the cloud infrastructure to support customization of the stored data, since it is likely that different tenants may want to store different data in their tables.
- For example, an automobile repair shop, different shops may want to store different details about the repairs carried out.
- Three methods for doing this are described in the next slides. It is to be noted that **difficulties for customization** occur only in the **shared table method**.
- In the dedicated table method, each tenant has their own table, and therefore can have different schema.

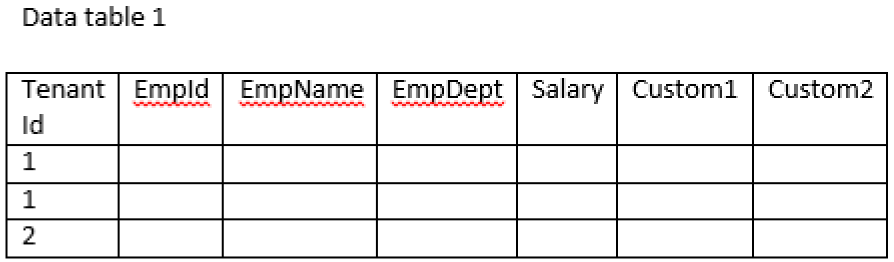
Support for Customization

Pre Allocated Columns:

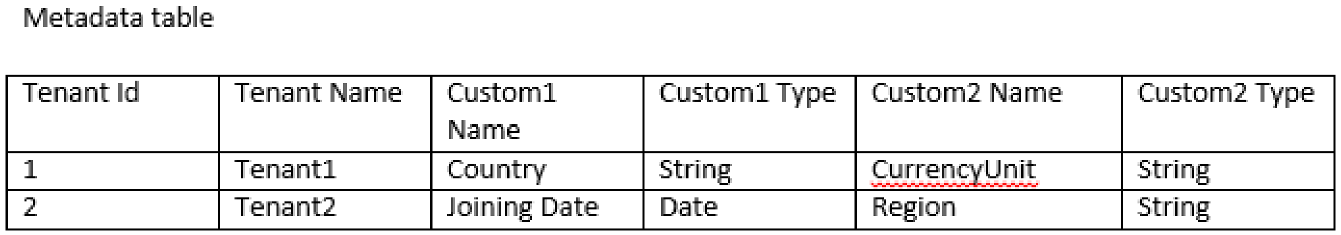
- In shared table approach, it’s very complex to provide support for the customizations.
- Each tenant might have his unique requirements to store data in the tables and using shared table approach, managing such requirements needs to come up with proper data architecture.

Pre allocated columns

- Fixed number of columns is reserved for custom columns
- If the numbers of custom column are too less than the reserved custom columns then space are wasted
- If the numbers of custom columns are more than the reserved custom columns then customer will feel restricted.


Data table 1

Tenant Id	EmpId	EmpName	EmpDept	Salary	Custom1	Custom2
1						
1						
2						


Metadata table

Tenant Id	Tenant Name	Custom1 Name	Custom1 Type	Custom2 Name	Custom2 Type
1	Tenant1	Country	String	CurrencyUnit	String
2	Tenant2	Joining Date	Date	Region	String

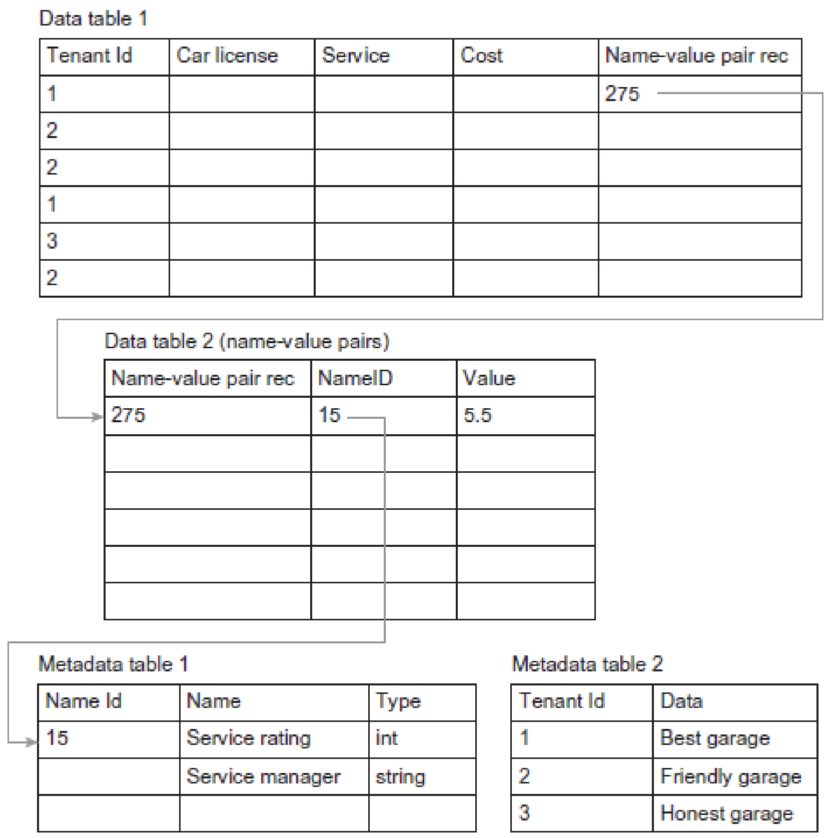
Support for Customization

Name-Value Pairs:

- The major problem with the pre-allocated columns technique is that there could be a lot of wasted space.
- If the number of columns is too low, then users will feel constrained in their customizations.
- However, if the number is too big, there will be a lot of space wastage.

Name-Value Pairs

- A metadata table for tenant is maintained
- A data table is specified with standard common columns and has extra column at end to point to another name value pair table
- Name value pair table (aka pivot table) actually includes the custom fields for this record.
- The actual custom fields are stored with data type and other metadata information.
- Space efficient as compared to pre allocated columns method
- Joins are involved to fetch tenant specific data



Data table 1

Tenant Id	Car license	Service	Cost	Name-value pair rec
1				275
2				
2				
1				
3				
2				

Data table 2 (name-value pairs)

Name-value pair rec	NameID	Value
275	15	5.5

Metadata table 1

Name Id	Name	Type
15	Service rating	int
	Service manager	string

Metadata table 2

Tenant Id	Data
1	Best garage
2	Friendly garage
3	Honest garage

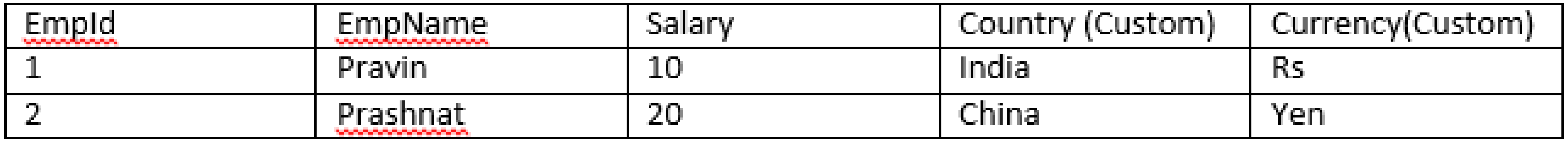
XML method :

- The last column of database table is reserved for storing the information in XML format

Support for Customization

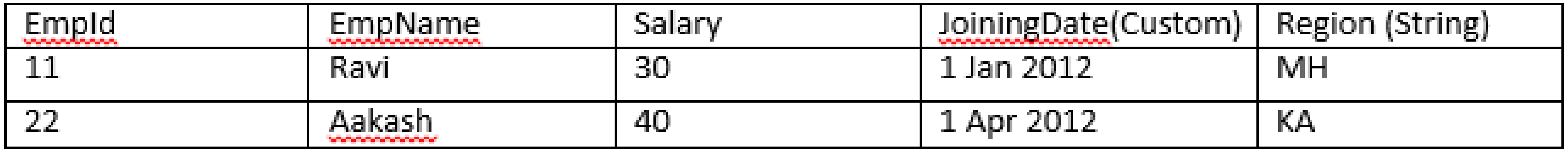
- Let’s consider Tenants have following table structure that needs to be mapped to the **Name-Value pair table structure**.
- Each tenant table has:
- standard common fields and
- few custom fields having varying data type

Tenant1 Employee table



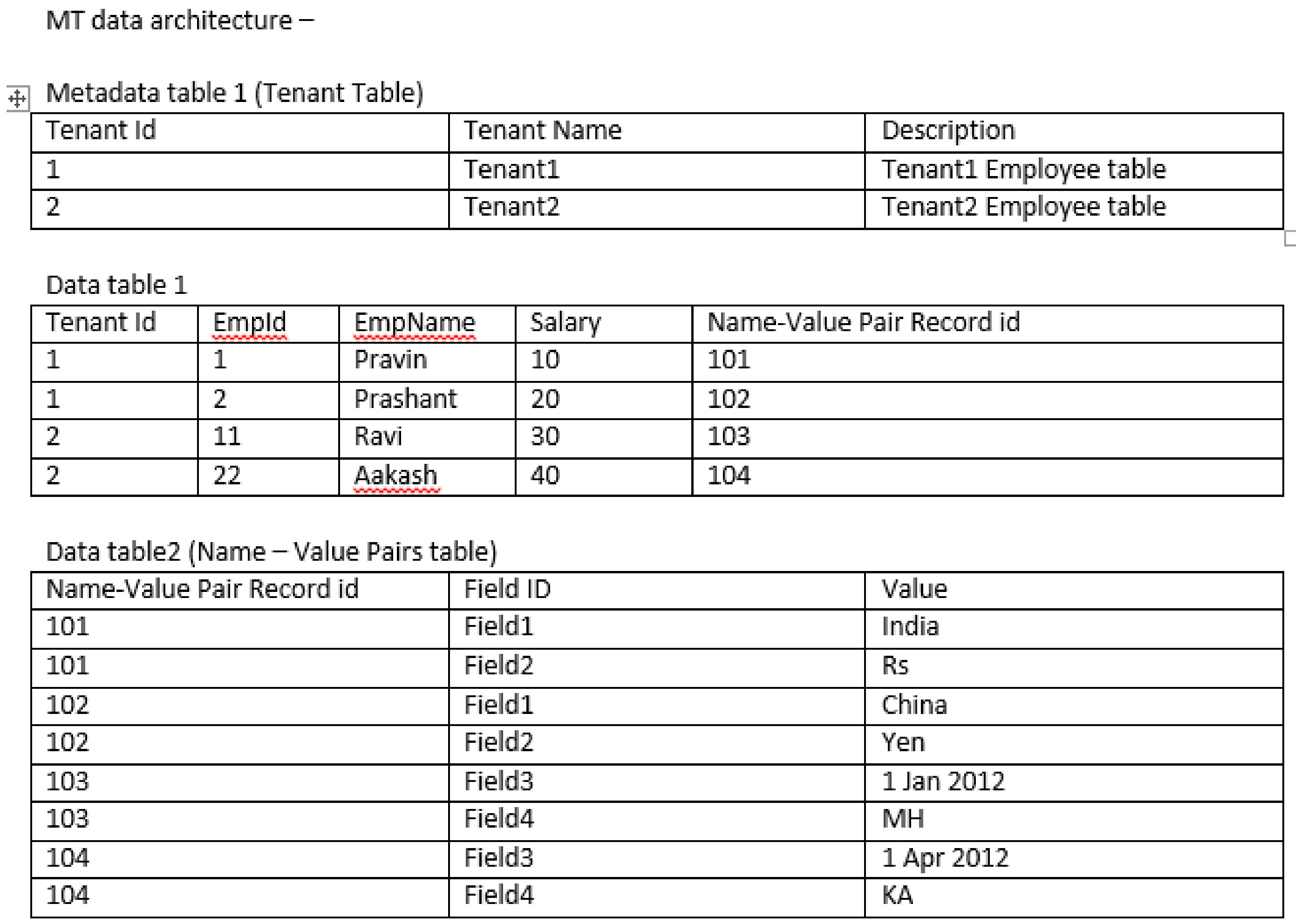
EmpId	EmpName	Salary	Country (Custom)	Currency(Custom)
1	Pravin	10	India	Rs
2	Prashnat	20	China	Yen

Tenant2 Employee table



EmpId	EmpName	Salary	JoiningDate(Custom)	Region (String)
11	Ravi	30	1 Jan 2012	MH
22	Aakash	40	1 Apr 2012	KA

Sample MT Architecture



MT data architecture –

Metadata table 1 (Tenant Table)

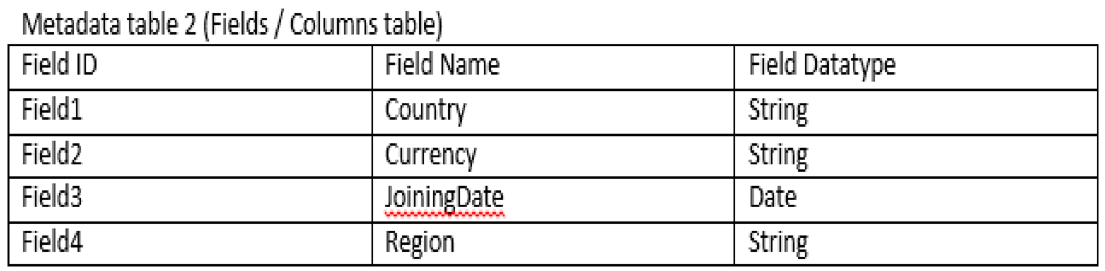
Tenant Id	Tenant Name	Description
1	Tenant1	Tenant1 Employee table
2	Tenant2	Tenant2 Employee table

Data table 1

Tenant Id	EmpId	EmpName	Salary	Name-Value Pair Record id
1	1	Pravin	10	101
1	2	Prashant	20	102
2	11	Ravi	30	103
2	22	Aakash	40	104

Data table2 (Name – Value Pairs table)

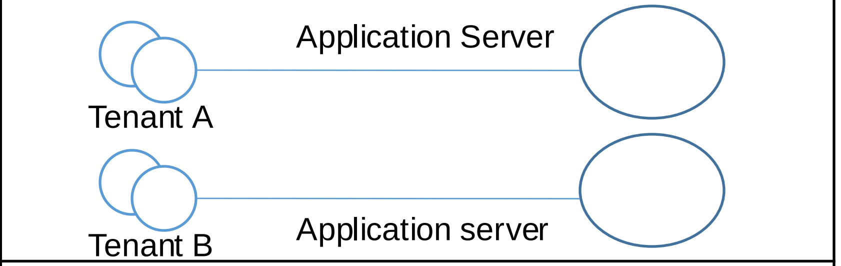
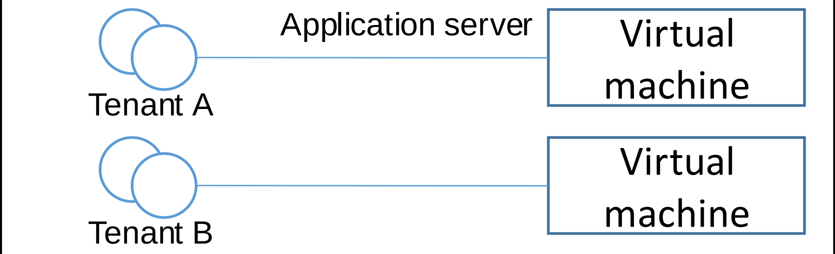
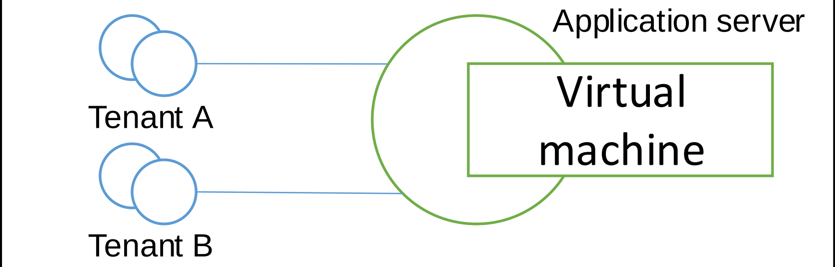
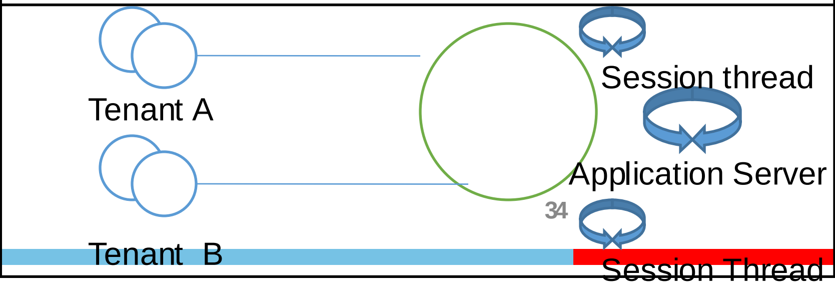
Name-Value Pair Record id	Field ID	Value
101	Field1	India
101	Field2	Rs
102	Field1	China
102	Field2	Yen
103	Field3	1 Jan 2012
103	Field4	MH
104	Field3	1 Apr 2012
104	Field4	KA



Metadata table 2 (Fields / Columns table)

Field ID	Field Name	Field Datatype
Field1	Country	String
Field2	Currency	String
Field3	JoiningDate	Date
Field4	Region	String

Multi Tenancy: Resource Sharing

Fully isolated Application server Each tenant accesses an application server running on a dedicated servers.	 Application Server Tenant A Tenant B Application server
Virtualized Application Server Each tenant accesses a dedicated application running on a separate virtual machine.	 Application server Virtual machine Tenant A Virtual machine Tenant B
Shared Virtual Server Each tenant accesses a dedicated application server running on a shared virtual machine.	 Application server Virtual machine Tenant A Tenant B
Shared Application Server The tenant shared the application server and access application resources through separate session or threads.	 Session thread Tenant A Application Server Tenant B Session Thread

Did You Know ?

Here are some interesting facts about multi-tenancy in the cloud:

1. Multi-tenancy is one of the key features of cloud computing that makes it so attractive to businesses. It allows multiple customers to share the same infrastructure and services, which can lead to cost savings and improved efficiency.
2. One of the biggest challenges of multi-tenancy is ensuring that each tenant's data is kept separate and secure. This requires careful planning and implementation of security measures to prevent data leakage and unauthorized access.
3. Multi-tenancy can be implemented at different levels in the cloud, including infrastructure, platform, and software. Each level requires different techniques and technologies to ensure proper isolation and security.
4. In the public cloud model, multi-tenancy is typically implemented using virtualization and resource sharing. Each tenant is assigned their own virtual resources, which are isolated from other tenants.

Did You Know ?

1.In the private cloud model, multi-tenancy can be implemented using virtualization or containerization, and is typically achieved through strict access controls and resource partitioning.

2.Hybrid cloud environments can offer the benefits of both public and private cloud models, but can also add complexity and require careful management to ensure proper isolation and security.

3.Multi-tenancy can offer significant cost savings for businesses, as they only pay for the resources they use. This can also lead to better resource utilization and scalability, as resources are shared among multiple tenants.

4.Multi-tenancy is not suitable for all types of applications, and some applications may require dedicated resources and environments. It is important to carefully evaluate the requirements of each application and choose the appropriate deployment model.

Overall, multi-tenancy is a key feature of cloud computing that can offer significant benefits to businesses, but it requires careful planning and implementation to ensure proper isolation and security.

Q & A……..

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Credits

•Hwang, Kai; Dongarra, Jack; Fox, Geoffrey C.. Distributed and Cloud Computing: From Parallel Processing to the Internet of Things (Kindle Locations 3532-3533). Elsevier Science. Kindle Edition.