Digital Banking Case Study : Micro Service approach

Agenda

1. Microservices – Changing business needs
2. Case Study – Digital Banking
3. Solution Design
4. Summary
5. Challenges & Mitigation
6. Roadmap

Microservices – Changing Business needs

SERVICES

More number of services (operations bundled into services)

SCALABILITY

Ensure scalability for specific operations basis the demand

MICROSERVICES

RESOURCE UTILIZATION

Pareto principle (80% of traffic comes from 20% of services)

DEPLOYMENT

Need based deployment model (applications not just in Java EE or .NET)

AGILITY

Agility in infrastructure, source code & associated projects as per organizational changes

CLOUD

Cloud based applications & deployment across devices

*"**Mesh App and Service Architecture**: The intelligent digital mesh will require changes to the architecture, technology and tools used to develop solutions. The mesh app and service architecture (MASA) is a multichannel solution architecture that leverages cloud and server-less computing, containers and **microservices** as well as APIs and events to deliver modular, flexible and dynamic solutions"* – **GARTNER, TECHNOLOGY TRENDS 2017**

*"**2017 will bring more complexity** — complexity that will overwhelm enterprises that don't get ahead of the problem. IoT solutions will be built on **modern microservices** and be distributed across edge devices, gateways, and cloud services"* – **FORESTER, TECHNOLOGY TRENDS 2017**

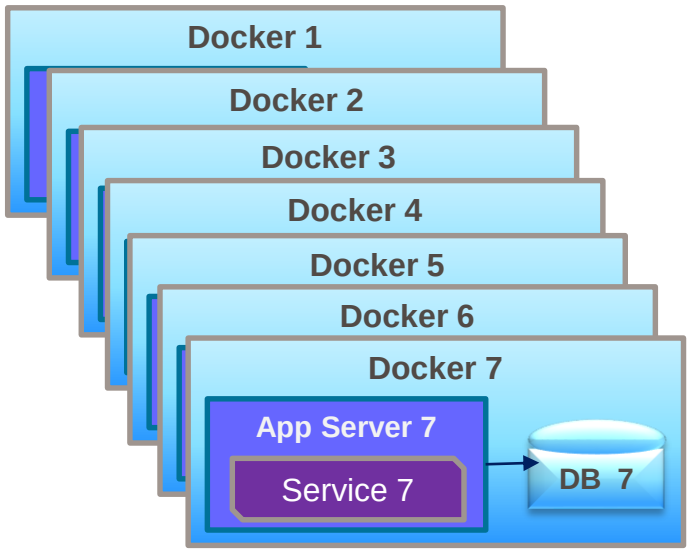
Digital Banking Case Study

DIGITAL BANKING

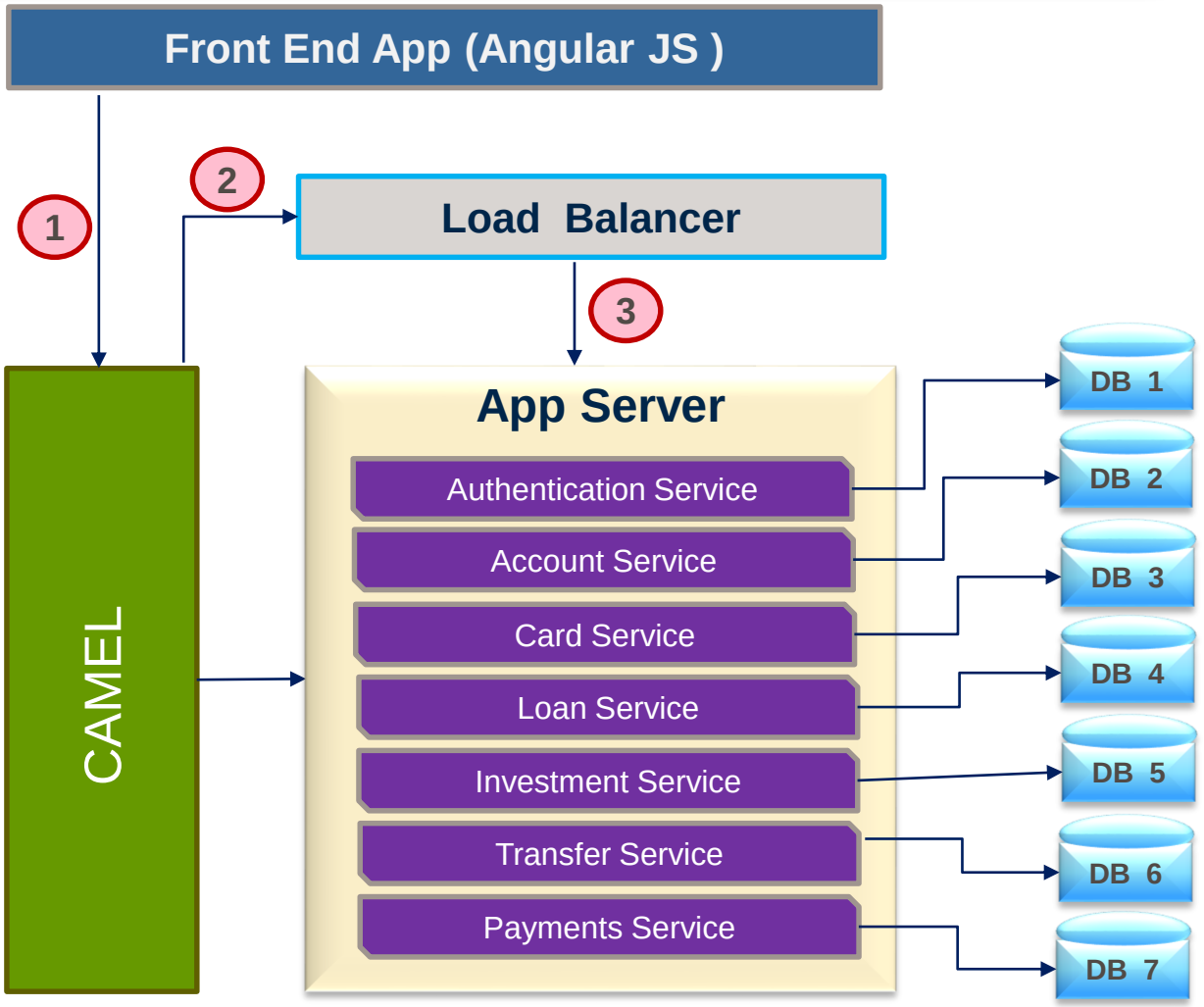
Developing a digital bank using the micro service model

1. Ability for a user to login using customer id and password
2. View a summary of accounts, cards, loans and investments
3. View transaction history on accounts, cards, loans and investments
4. Ability to transfer funds
5. Ability to make Bill Payments

Solution Design – Overview of the architecture



Container / Deployment Solution



Logical Components Architecture and Flow

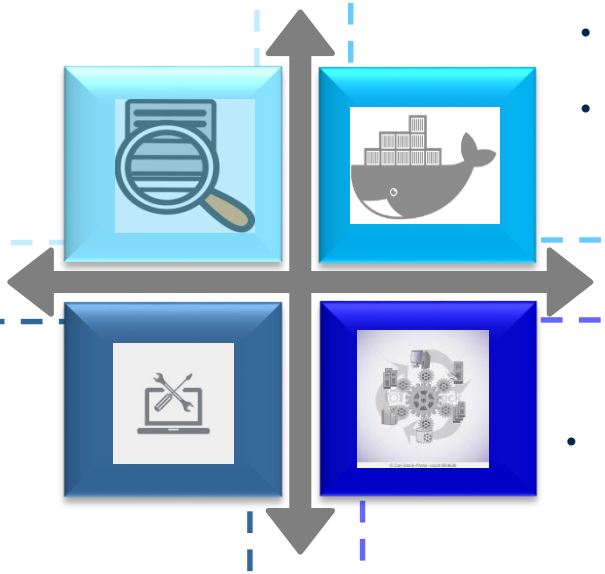
Solution Design – Overview of the architecture

Application Performance and Monitoring

- Monitoring to be setup for every container.
- Services running inside the container to be monitored as well.
- We plan to use ELK stack to run analytics on the logs
- Log aggregation to happen on an external storage service.

Containerization

- Each micro service to run in its own container which includes the database.
- Container to container communication to be only through APIs.
- We intend to start with one container per micro service but build it in a scalable manner with load balancing.



Source Code Maintenance

- Source code will be maintained on Github.
- IBM's cloud environment will be used to do build and deploy the application.
- Local development will be done collaboratively between Capgemini and IBM

Service Choreography

- AngularJS to manage service choreography For example, if the summary page needs calls to the accounts service, cards service and loans service, it will be managed through AngularJS.
- Each API call to undergo authentication using a standard security token (OATH authentication mechanism)

Quick glance at service split up

Micro Service	Functionality	Account Types	Technologies
AuthenticationService	Authenticate and authorize the user	Retail customers	Spring, Hibernate, Postgresql
AccountService	View list of accounts, account summary and list of transactions	Checking Accounts, Savings Accounts	Spring Boot, JPA, Postgresql
CardService	View list of cards, card summary and list of transactions	Credit Cards, Virtual Credit Cards	Dropwizard, mysql
LoanService	View list of loans and loan summary	Personal Loan, Auto Loan, Home Loan	Nosql
InvestmentService	View list of investments, investment summary and list of transactions	Term Deposit, Recurring Deposit	TBD
TransferService	Setup payees and transfer between own accounts, other accounts in same bank and other accounts in other banks	Transfers between a) account to account b) account to card c) account to loan d) account to investment	Python, Django
PaymentsService	Setup merchants and make payments to merchants.	Payments to a) Telephone biller b) Electricity Biller	TBD

- The application to be built using the micro service approach and hosted on the IBM Blumix cloud.
- We will cover select account types only to keep a restricted scope.
- Frontend will be Angularjs, bootstrap and HTML 5.
- Polyglot approach will be used for development

Summary - Technology Portfolio

7

Key technology areas addressed with comprehensive focus and multiple assets

User Interface Engg	AngularJS by Google, jQuery write less, do more., HTML5 + CSS3
Open Source Frameworks	Spring Batch, MyFaces, Spring Web Flow, Spring Web Services, Struts, GemFire
Core J2EE	JSF, PrimeFaces, JSP, JBoss by Red Hat
Integration Technology	WebSphere MQ, Oracle Service Bus, Apache Camel
No Sql	mongoDB, Neo4j Graph DB, redis
DevOps	Chef, puppet labs, sonarqube, kubernetes, docker
Blockchain	Blockchain, Ethereum, BigchainDB, Hyperledger

Challenges & Mitigation

Breaking down a monolythic database
How to build a common security model
How to optimize the # running containers
Building foreign key relations between data

Future Road Map - Exploring Blockchain

- Once the application is developed to a good extent, we can evaluate a use case of doing international transfers using block chain technology.
- We can create 2 fictional banks, each dealing with a different currency.
- Each bank is hosted on a different url.
- Transfers would use a distributed database.

- We can use Corda to write the contracts between the banks.

- Both banks would use this as a shared database.

THANK YOU