A Primer to building Government Web Applications with Microservices

GDS, PDD, AI

GOVTECH
SINGAPORE

The quest for robustness and scalability

- Your project is a living entity
  - Budgeted and paid for in terms of years or decades
  - Changing user workloads
  - Subject to evolving requirements, work processes and security landscape
- How quickly can you change and adapt?

As project stakeholders, we want the applications and services that we build for our project to be robust and scalable.
The project is a living entity, budgeted and paid for in terms of years or decades.
Over this period of time, the project will face different user loads, especially on launch day, or during peak periods or when the public takes a sudden interest in your service.
The project may also face changing requirements, due to change of government or organizational policy or work processes.
We are also faced with an evolving security landscape, we face threats from hackers, malware and even security vulnerabilities in our own software.
That leaves us with the question on how quickly can you change and adapt your applications?

The Microservice Movement

- Microservices are compatible with all the current tech trends!
  - Agile, DevOps, Cloud Computing, Infrastructure-as-Code
- Best practices advocated by all the big tech companies!
  - Netflix, Amazon, Google, Facebook, Uber
  - Disposable and Automatable Infrastructure
  - Common engineering solutions to accommodate different problems

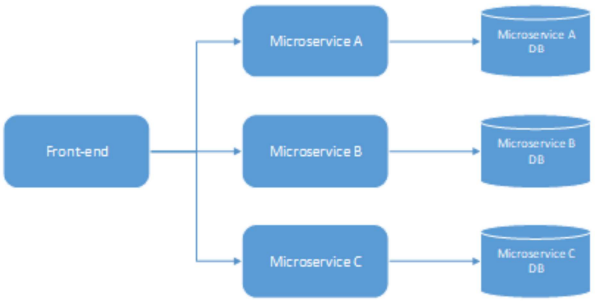
Nowadays, we are seeing more and more commercial platforms being built with microservices. Agile, DevOps, Cloud Computing, Infrastructure-as-Code, these tech trends are all compatible with microservices.

We also have a variety of big tech companies to learn from when it comes to implementing microservices. Netflix, Amazon, Google, Facebook and Uber.

These tech companies all advocate disposable and automatable infrastructure, and share common engineering solutions to accommodate different problems.

Microservices

- Application functionality split into modular microservices
- Loosely coupled bounded contexts
- Should be simple enough to be implemented and maintained by a small team or individuals.

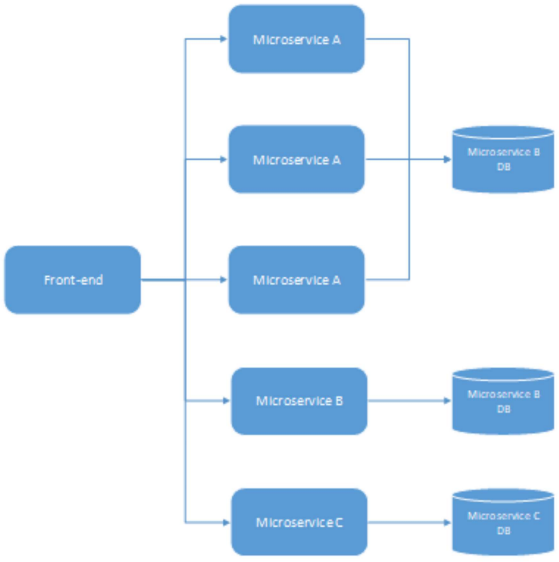


GOVTECH SINGAPORE

With microservices, you split your application functionality into modular microservices.
The key phrase to keep in mind is loosely coupled bounded contexts.
What this means is that every microservice is designed to be self contained, and the microservice should cover a specific area of work.
Each microservice should be simple enough to be implemented and maintained by a small team or individuals. This is compatible with agile methodologies where you are supposed to break down your requirements into smaller tasks.

Microservices

- Communication over REST API or message bus
- Microservices are designed to be stateless, therefore scalable



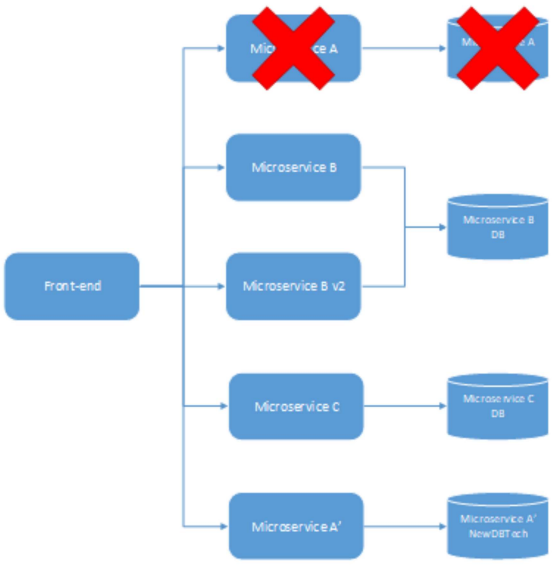
GOVTECH SINGAPORE

Microservices communicate with each other through REST APIs, or through a message bus

Microservices are designed to be stateless. When the state is stored on a database, memory cache or message bus, you can scale your application with more instances of your microservices to handle bigger workloads.

Microservices

- Microservices can be:
  - Disposable
  - Replaceable
  - Backwards Compatible
- Deployments can be:
  - Small and isolated
  - Easily reversed
  - Automatable



GOVTECH SINGAPORE

Microservices can be:

- Disposable: If a microservice instance is malfunctioning, you can throw away that instance and replace it. With DevOps automation, you can do this quickly and easily.
- Replaceable: When you need to change your business logic to adapt to new requirements, or when you adopt new technology, you can replace your microservice module just like batteries
- Backwards compatible: By deploying multiple versions of a microservice, you can maintain backwards compatibility

Since everything is modular, Deployments of your microservices should be:

- Small and isolated enough to deploy without affecting existing microservices
- Similarly, the deployment can be easily reversed
- Automatable with continuous integration/deployment software

Domain Driven Design

- Describe your business requirements
  - A member of the public sends me an application form
  - I need to process the application and send a response
  - Boss reviews the reports of applications and observes statistical trends

We can approach our application microservice design using a Domain Driven Design approach.
First, we describe our business requirements,
A member of the public sends me an application
I need to process the application and send a response
My boss needs to review reports of the applications and observe statistical trends

Domain Driven Design

- Create Ubiquitous Language
  - Entities: Member of Public, Application, Response, Reviewer
  - Processes & Events: Receive Application, Process Application, Send Response, Review Applications
- Identify domains/bounded contexts:
  - Sending/Receiving of Applications
  - Processing of Applications/Sending Responses
  - Reporting and Review of Applications

GOVTECH SINGAPORE

Using the business requirements, we create what we call a Ubiquitous language. The aim of this ubiquitous language is for your developers and product stakeholders to clearly define a context or domain, and the business logic to be performed with this domain.

First, we Identify the different entities, data models (value objects) and processes that will be enacted on the entities and the events that trigger these processes

Entities: Member of public, Thing, I (Staff), Boss, Response

Processes & Events: Receive Thing, Validate Thing, Send Response, Review Thing(s)

Identify domains/bounded contexts

Sending and receiving of Things

Validation, Sending of Responses

Reporting and Review of Things

Domain Driven Design

- Microservices design can be derived from bounded contexts
  - Public Applications Portal
  - Application Processing System
  - Application Review System
- Can break down further
  - Frontend & Backend APIs
  - Data Analytics
- Iterative process
  - Maintain consistency of ubiquitous language, add more bounded contexts

GOVTECH SINGAPORE

Your microservices design, business logic and data models can be derived from your bounded contexts

Public Applications Portal

Application Processing system

Application Review System

We can break down certain domains further:

Frontend & Backend APIs

Data Analytics

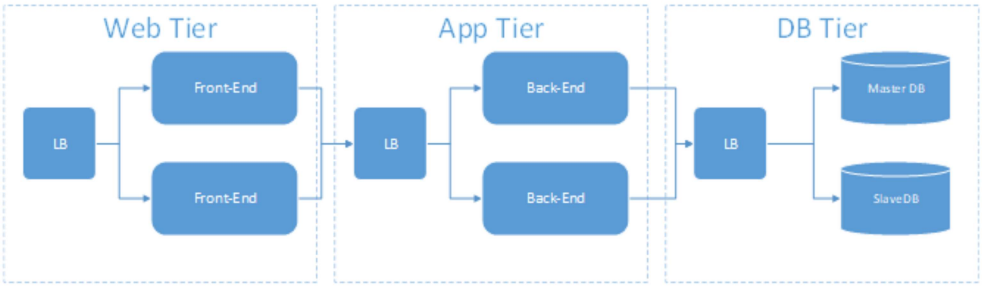
Domain Driven Design is an iterative process

Won't get all requirements in the first cut, will need to iterate over your design

Maintain consistency of ubiquitous language, you may need to preserve your existing microservices and add more separate microservices to address your additional requirements

Architecting your application

High Availability Web-App-DB

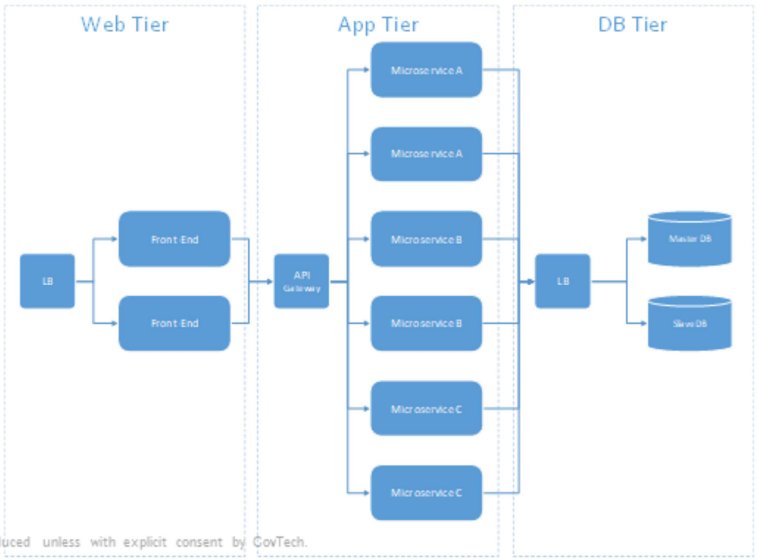


GOVTECH SINGAPORE

Standard High Availability Web-App-DB
Because your application is now comprised of several microservices, your system architecture becomes very important. Here we have the standard high availability Web-App-DB application

Architecting your application

High Availability Microservices

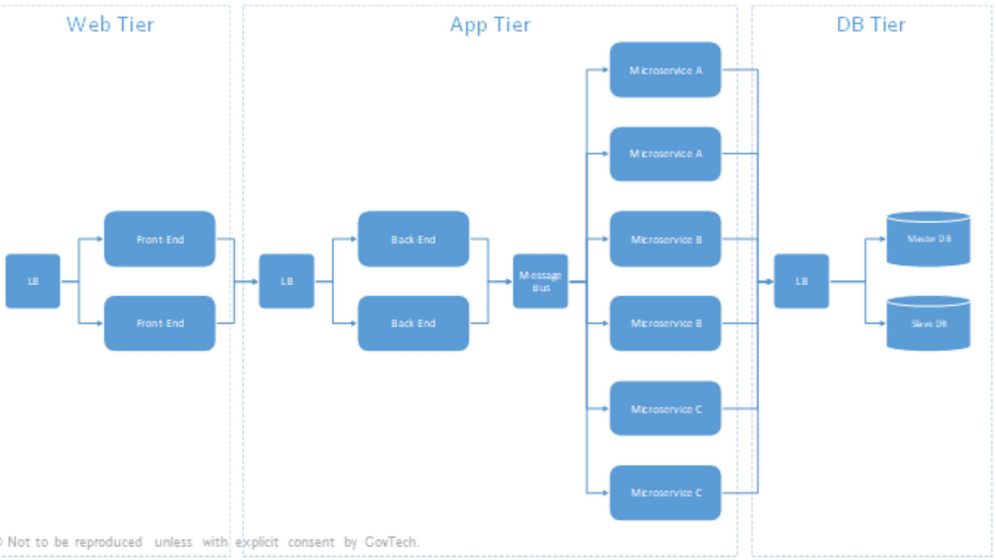


GOVTECH SINGAPORE

Here we have a High Availability Microservices application
When you scale your microservices, it helps to have a service discovery tool to load balance traffic across all your microservice instances.
When you have a large number of microservices, it also helps to have an API gateway to help you manage all the different APIs

Architecting your application

High Availability Microservices w/ Message Bus

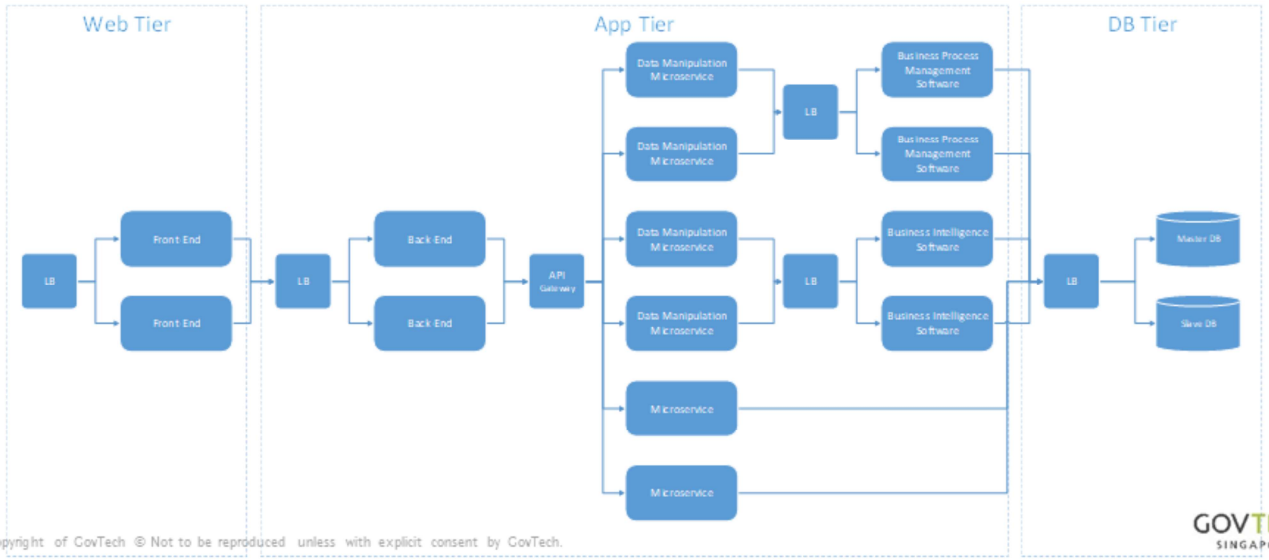


GOVTECH SINGAPORE

When you require real time processing or you are doing large batch processing of records, it may be better to use a message bus, where all your microservices listen to events from a backend service, perform operations and respond back to your backend using the message bus.

Architecting your application

Commercial Off-the-Shelf Software



GOVTECH SINGAPORE

Commercial of the Shelf Software should still be used where possible

Offer Powerful functionality such as Business Process Management or Business Intelligence or Data Analytics

Most newer COTS products will expose an API to allow interaction with the underlying functionality

COTS products should be used to the point where it is too hard to customize the COTS product further to meet the project requirements

May need to develop multiple microservices to massage data between other data sinks/sources

What microservices should be

- Modular!
  - Loosely coupled, so that each microservice can be easily maintained or replaced from development and ops perspectives
- Stateless
  - State should be stored in database, memory cache or message bus in order to support scaling
- Robustness and scalability follows from architecting your microservices correctly

Singapore Government Tech Stack

- Library of Microservices
  - WOG Application Analytics
  - Your app here!
- Application Infrastructure
  - Nectar Platform-as-a-Service
  - APEX API Gateway
- Infrastructure
  - Government Private Cloud
  - Government Data Cloud
  - R-Cloud
- Data
  - MyInfo
  - Enterprise Data Hub

GOVTECH SINGAPORE

The Singapore Government Tech Stack is an ongoing effort to build and reuse a library of services on top of secure and robust infrastructure.

References

- Microservices
  https://technology.amis.nl/2016/09/25/can-learn-microservices-movement/

- Domain Driven Design
  https://airbrake.io/blog/software-design/domain-driven-design

References

- Content with references to Singapore Government cybersecurity policies have been redacted. Please consult your government partner for the appropriate compliance requirements for your project.

References

- Singapore Government Tech Stack https://www.smartnation.sg/docs/default-source/cos2018/singapore-government-technology-stack-factsheet.pdf

GOVTECH SINGAPORE