Distributed Systems I

Software Architecture

Brae Webb & Richard Thomas

March 23, 2026

Mathias Verras
@mathiasverraes

There are only two hard problems in distributed systems:
2. Exactly-once delivery
1. Guaranteed order of messages
2. Exactly-once delivery

Going forward

Investigating architectures that are *distributed*.

Distributed Systems Series

Distributed I	*Reliability* and *scalability* of *stateless* systems.
Distributed II	*Complexities* of *stateful* systems.
Distributed III	*Hard problems* in distributed systems.

What are the benefits?

- Improved *reliability*
- Improved *scalability*
- Improved *latency*

What are the drawbacks?

- Increased *complexity*
- Increased *attack surface*
- Increased *latency*
- Introduce *consistency* problems

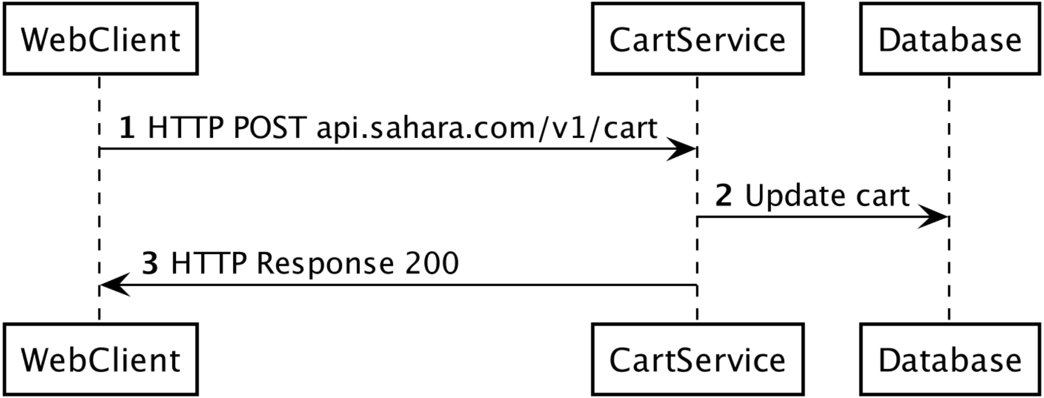
§ *Fallacies*

A few reasons for complexity

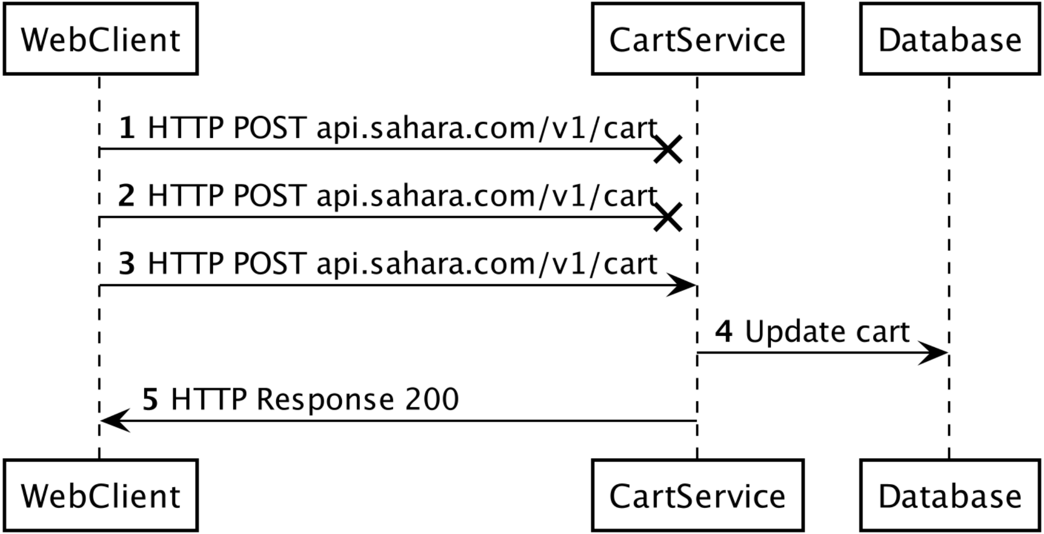
Fallacies of *Distributed Computing*

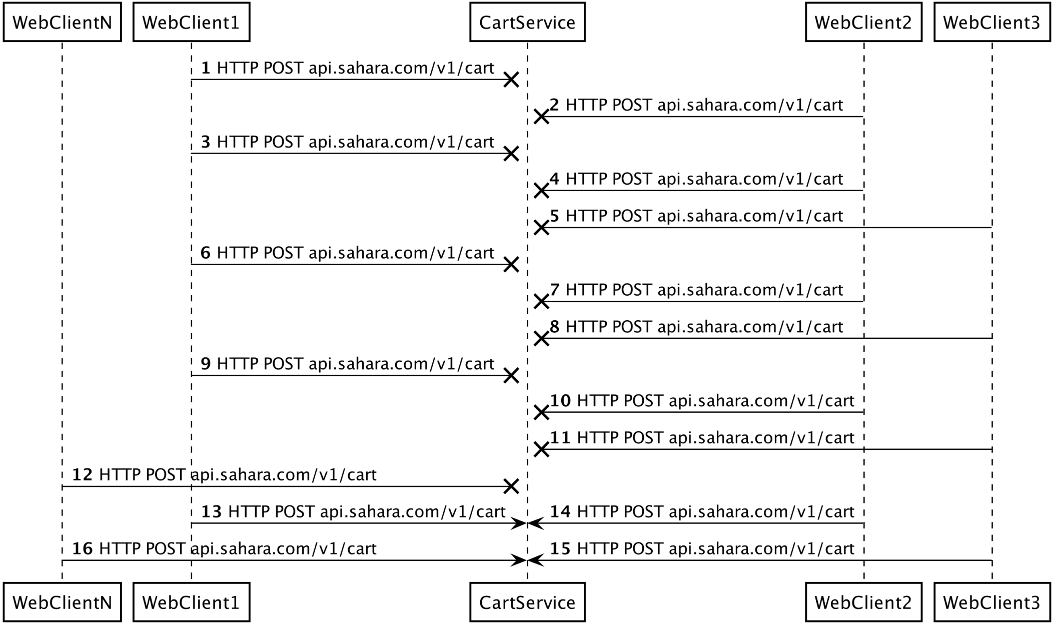
Fallacy #1

The network is reliable


WebClient
CartService
Database
1 HTTP POST api.sahara.com/v1/cart
2 Update cart
3 HTTP Response 200
WebClient
CartService
Database


WebClient
CartService
1 HTTP POST api.sahara.com/v1/cart
WebClient
CartService

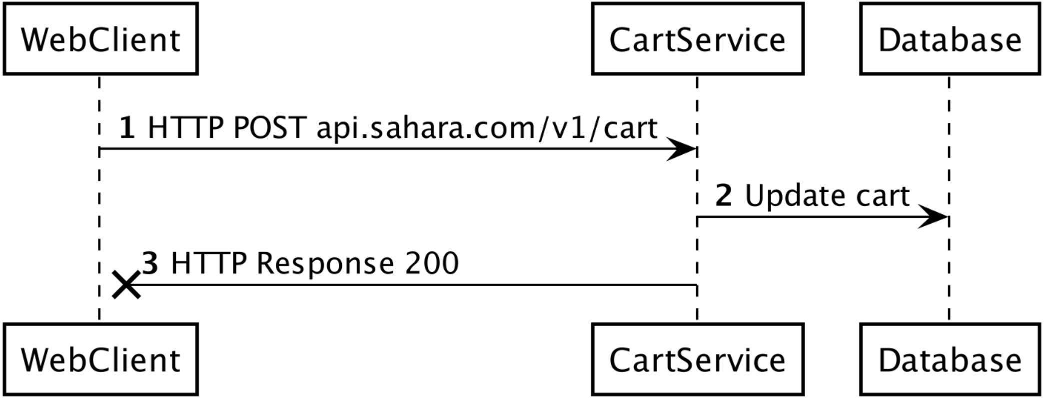

WebClient
CartService
Database
1 HTTP POST api.sahara.com/v1/cart
2 HTTP POST api.sahara.com/v1/cart
3 HTTP POST api.sahara.com/v1/cart
4 Update cart
5 HTTP Response 200
WebClient
CartService
Database

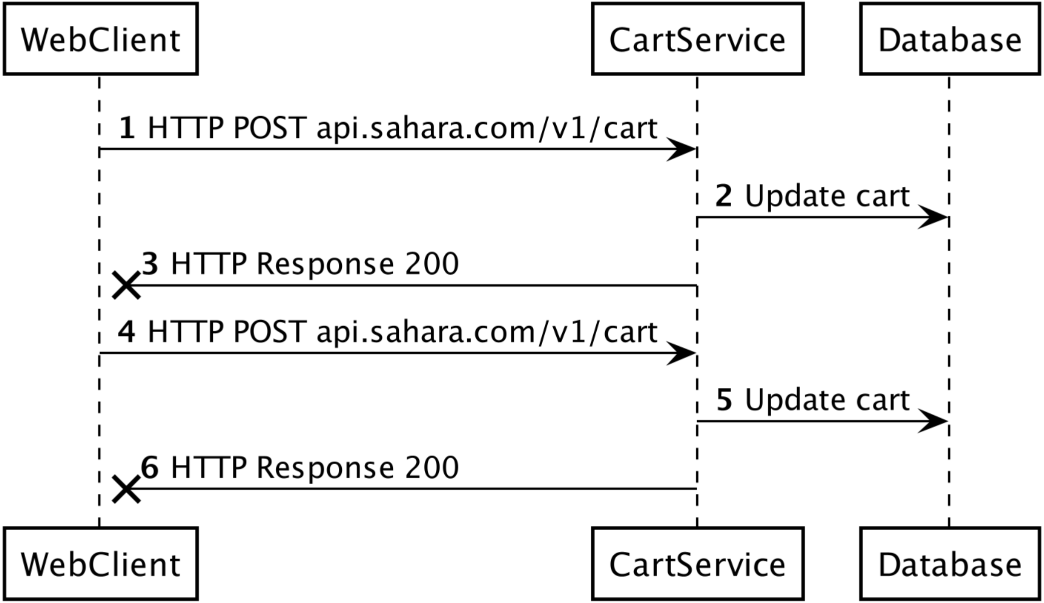

WebClientN
WebClient1
CartService
WebClient2
WebClient3
1 HTTP POST api.sahara.com/v1/cart
2 HTTP POST api.sahara.com/v1/cart
3 HTTP POST api.sahara.com/v1/cart
4 HTTP POST api.sahara.com/v1/cart
5 HTTP POST api.sahara.com/v1/cart
6 HTTP POST api.sahara.com/v1/cart
7 HTTP POST api.sahara.com/v1/cart
8 HTTP POST api.sahara.com/v1/cart
9 HTTP POST api.sahara.com/v1/cart
10 HTTP POST api.sahara.com/v1/cart
11 HTTP POST api.sahara.com/v1/cart
12 HTTP POST api.sahara.com/v1/cart
13 HTTP POST api.sahara.com/v1/cart
14 HTTP POST api.sahara.com/v1/cart
16 HTTP POST api.sahara.com/v1/cart
15 HTTP POST api.sahara.com/v1/cart
WebClientN
WebClient1
CartService
WebClient2
WebClient3

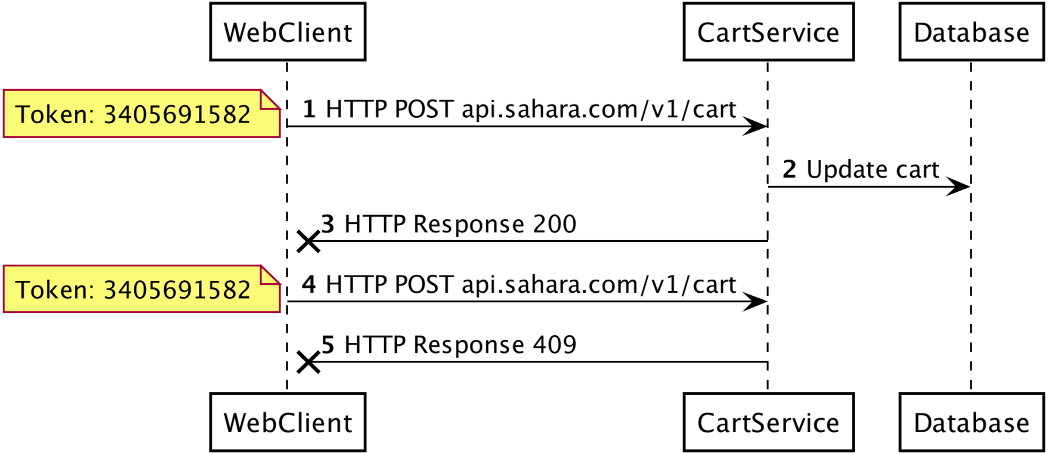
Exponential Backoff

```
retries = 0
do:
    status = service.request()

    if status != SUCCESS:
        retries += 1
        wait(2 ** retries)
while (status != SUCCESS and retries < MAX_RETRIES)
```


WebClient
CartService
Database
1 HTTP POST api.sahara.com/v1/cart
2 Update cart
3 HTTP Response 200
WebClient
CartService
Database


WebClient
CartService
Database
1 HTTP POST api.sahara.com/v1/cart
2 Update cart
3 HTTP Response 200
4 HTTP POST api.sahara.com/v1/cart
5 Update cart
6 HTTP Response 200
WebClient
CartService
Database


WebClient
CartService
Database
Token: 3405691582
1 HTTP POST api.sahara.com/v1/cart
2 Update cart
3 HTTP Response 200
Token: 3405691582
4 HTTP POST api.sahara.com/v1/cart
5 HTTP Response 409
WebClient
CartService
Database

Fallacy #2

Latency is zero

Network Statistics

Home to UQ

Home to us-east-1

EC2 to EC2

Network Statistics

Home to UQ 20.025ms

Home to us-east-1

EC2 to EC2

Network Statistics

Home to UQ 20.025ms

Home to us-east-1 249.296ms

EC2 to EC2

Network Statistics

Home to UQ	20.025ms
Home to us-east-1	249.296ms
EC2 to EC2	0.662ms

Fallacy #3

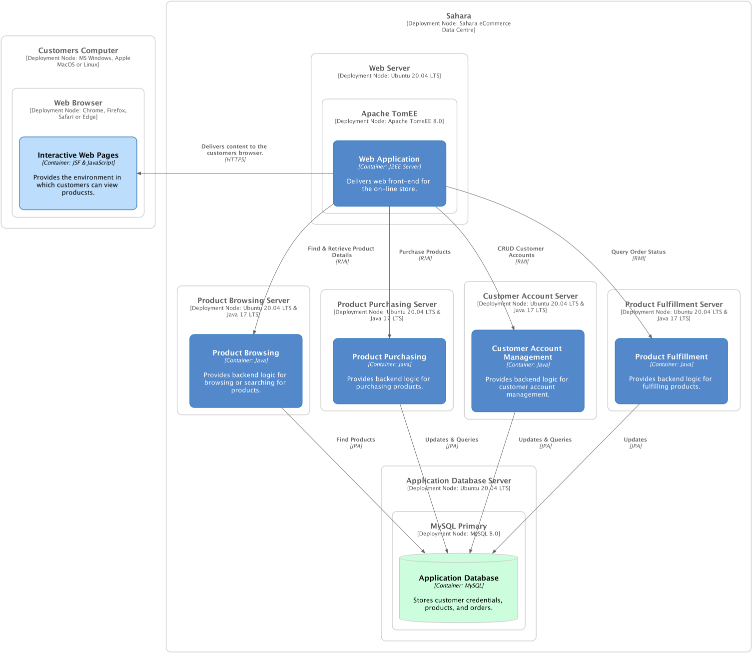
Bandwidth is infinite

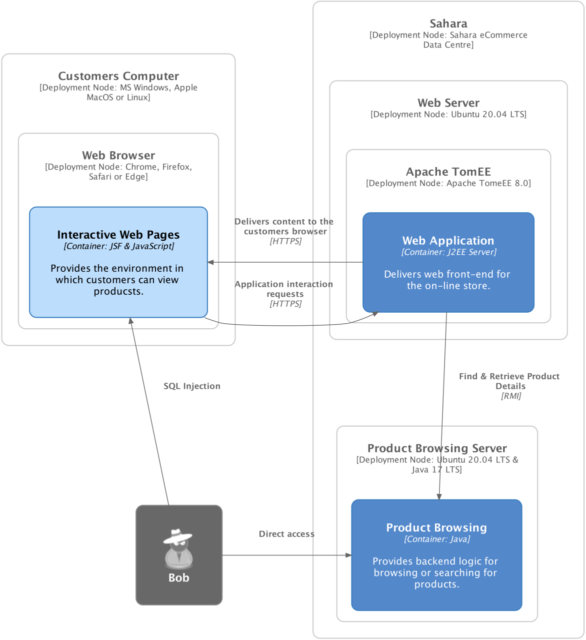
Definition 0. Stamp Coupling

Components which share a composite data structure.

Fallacy #4

The network is secure


Sahara
[Deployment Node: Sahara eCommerce Data Centre]
Customers Computer
[Deployment Node: MS Windows, Apple MacOS or Linux]
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view producsts.
Web Server
[Deployment Node: Ubuntu 20.04 LTS]
Apache TomEE
[Deployment Node: Apache TomEE 8.0]
Web Application
[Container: J2EE Server]
Delivers web front-end for the on-line store.
Delivers content to the customers browser.
[HTTPS]
Find & Retrieve Product Details
[RMI]
Purchase Products
[RMI]
CRUD Customer Accounts
[RMI]
Query Order Status
[RMI]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Purchasing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Purchasing
[Container: Java]
Provides backend logic for purchasing products.
Customer Account Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Customer Account Management
[Container: Java]
Provides backend logic for customer account management.
Product Fulfillment Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Fulfillment
[Container: Java]
Provides backend logic for fulfilling products.
Find Products
[JPA]
Updates & Queries
[JPA]
Updates & Queries
[JPA]
Updates
[JPA]
Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]
MySQL Primary
[Deployment Node: MySQL 8.0]
Application Database
[Container: MySQL]
Stores customer credentials, products, and orders.


Sahara
[Deployment Node: Sahara eCommerce Data Centre]
Customers Computer
[Deployment Node: MS Windows, Apple MacOS or Linux]
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view producsts.
Web Server
[Deployment Node: Ubuntu 20.04 LTS]
Apache TomEE
[Deployment Node: Apache TomEE 8.0]
Web Application
[Container: J2EE Server]
Delivers web front-end for the on-line store.
Delivers content to the customers browser
[HTTPS]
Application interaction requests
[HTTPS]
Find & Retrieve Product Details
[RMI]
SQL Injection
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Direct access
Bob

Fallacy #5

The topology never changes

Fallacy #6

There is only one administrator

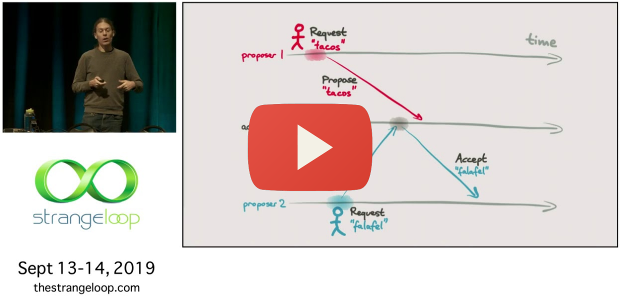
Fallacy #7

Transport cost is zero

Remember

Distributed systems are *hard*. Choosing to use them should be *well considered*.

When you need to, maybe prove it?



https://youtube.com/watch?v=7w4KC6i9Yac

Or, more realistically,

Use *existing* algorithms and software

Distributed Systems Series

Distributed I	*Reliability* and *scalability* of *stateless* systems.
Distributed II	Complexities of stateful systems.
Distributed III	Hard problems in distributed systems.

Stateless vs. Stateful Systems

Stateless Does *not* utilise *persistent data*.

Stateful Does utilise *persistent data*.

Question

What makes software *reliable*?

Definition 0. Reliable Software

Continues to work, even when things go wrong.

Definition 0. Fault

Something goes wrong.

Death, taxes, and computer system failure are all inevitable to some degree.

Plan for the event.

– *Howard and LeBlanc*

Reliable software is

Fault *tolerant*

Problem

Individual computers fail *all the time*

Solution

Spread the risk of faults over *multiple computers* or *nodes*

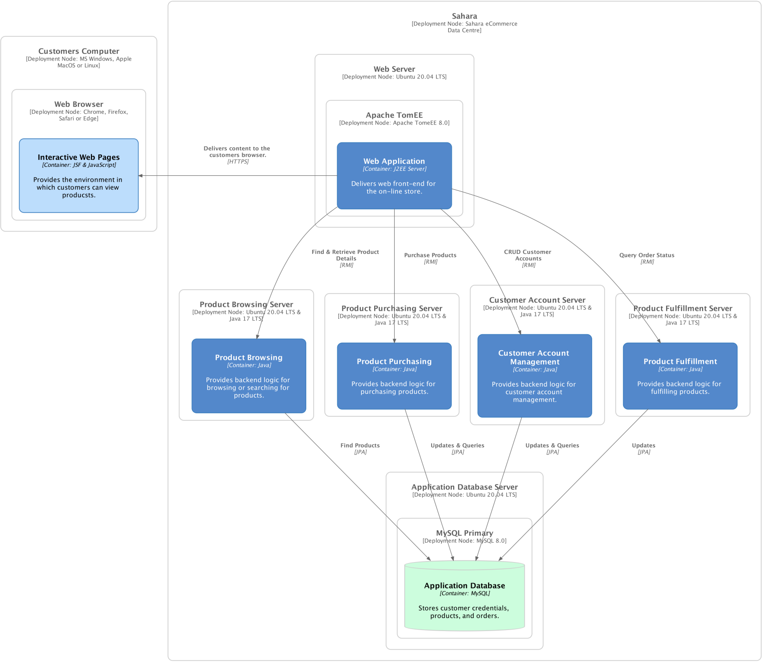
Spreading Risk

If you have software that works with *just one* computer, spreading the software over *two* computers *halves* the risk that your software will fail.

Spreading Risk

If you have software that works with *just one* computer, spreading the software over *two* computers *halves* the risk that your software will fail.

Adding *100* computers reduces the risk by *100*.


Sahara
[Deployment Node: Sahara eCommerce Data Centre]
Customers Computer
[Deployment Node: MS Windows, Apple MacOS or Linux]
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view producsts.
Delivers content to the customers browser.
[HTTPS]
Web Server
[Deployment Node: Ubuntu 20.04 LTS]
Apache TomEE
[Deployment Node: Apache TomEE 8.0]
Web Application
[Container: J2EE Server]
Delivers web front-end for the on-line store.
Find & Retrieve Product Details
[RMI]
Purchase Products
[RMI]
CRUD Customer Accounts
[RMI]
Query Order Status
[RMI]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Purchasing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Purchasing
[Container: Java]
Provides backend logic for purchasing products.
Customer Account Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Customer Account Management
[Container: Java]
Provides backend logic for customer account management.
Product Fulfillment Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Fulfillment
[Container: Java]
Provides backend logic for fulfilling products.
Find Products
[JPA]
Updates & Queries
[JPA]
Updates & Queries
[JPA]
Updates
[JPA]
Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]
MySQL Primary
[Deployment Node: MySQL 8.0]
Application Database
[Container: MySQL]
Stores customer credentials, products, and orders.

Question

Who has used *auto-scaling*?

Auto-Scaling Terminology

Auto-scaling group A *collection of instances* managed by auto-scaling.

Auto-Scaling Terminology

Auto-scaling group A *collection of instances* managed by auto-scaling.

Capacity Amount of instances *currently* in an auto-scaling group.

Auto-Scaling Terminology

Auto-scaling group A *collection of instances* managed by auto-scaling.

Capacity Amount of instances *currently* in an auto-scaling group.

Desired Capacity Amount of instances *we want to have* in an auto-scaling group.

Auto-Scaling Terminology

Auto-scaling group A *collection of instances* managed by auto-scaling.

Capacity Amount of instances *currently* in an auto-scaling group.

Desired Capacity Amount of instances *we want to have* in an auto-scaling group.

Scaling Policy How to determine the desired capacity.

Auto-Scaling Terminology

Auto-scaling group A *collection of instances* managed by auto-scaling.

Capacity Amount of instances *currently* in an auto-scaling group.

Desired Capacity Amount of instances *we want to have* in an auto-scaling group.

Scaling Policy How to determine the desired capacity.

Minimum/Maximum Capacity *Hard limits* on the minimum and maximum number of instances.

What we really want

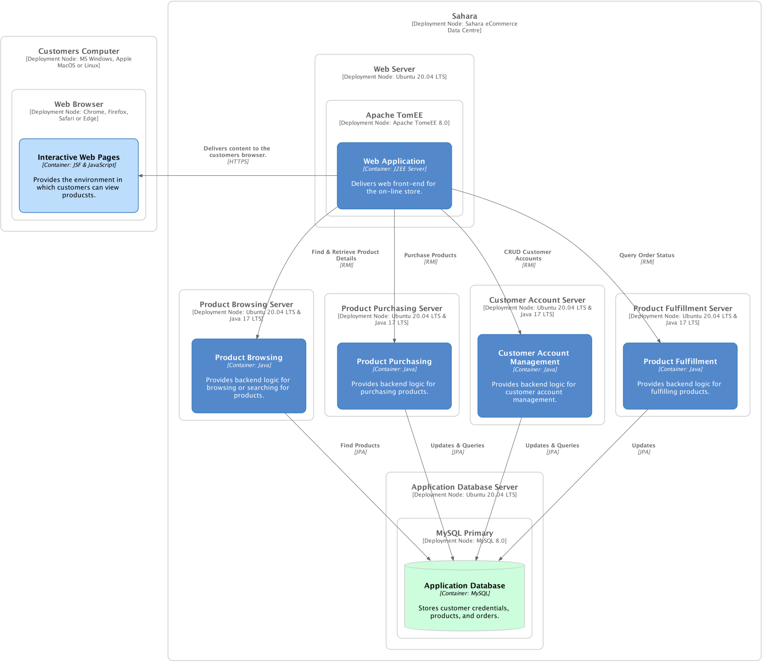
Desired Capacity Amount of *healthy* instances we want to have in an auto-scaling group.

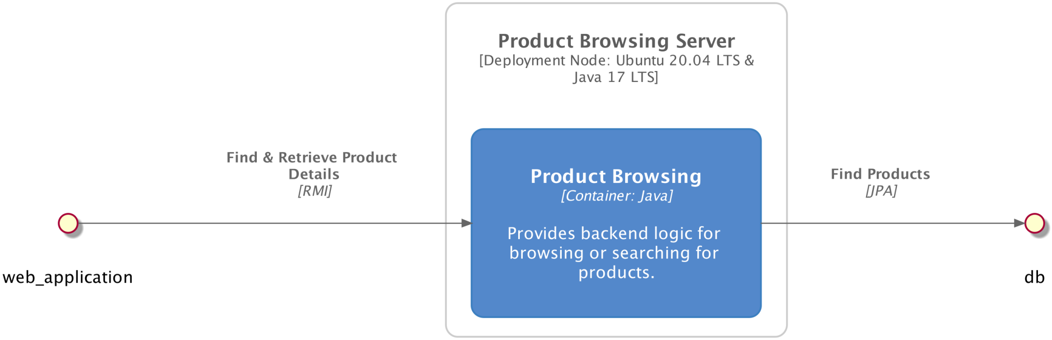
Health check

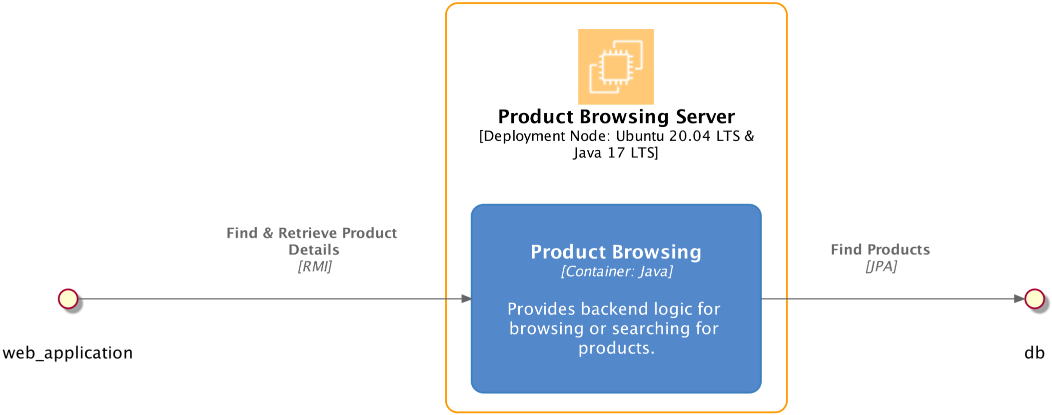
Mechanism to determine whether an instance is *healthy*.

Auto-scaling

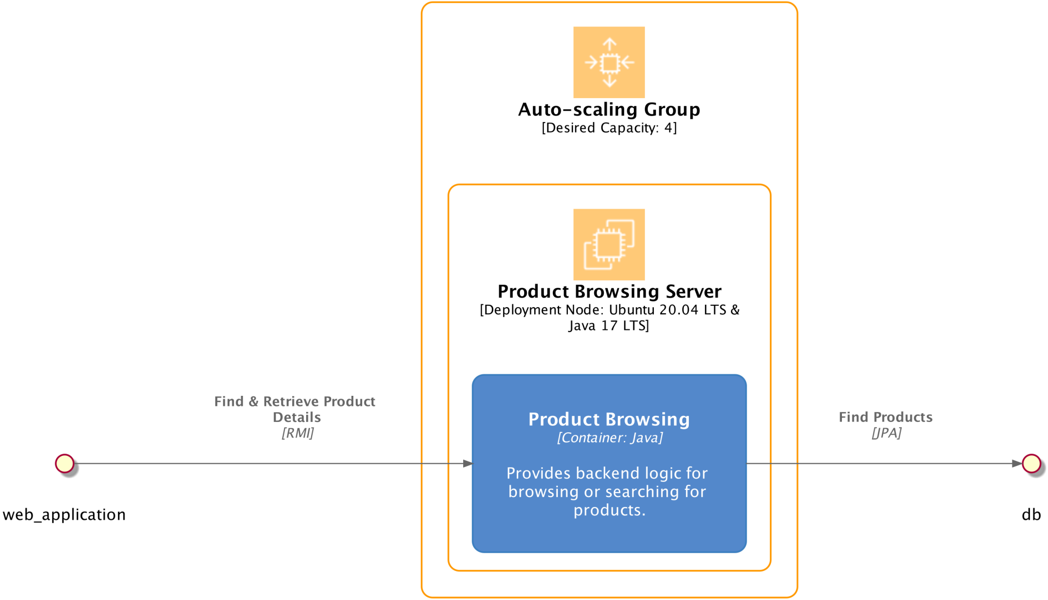
An example

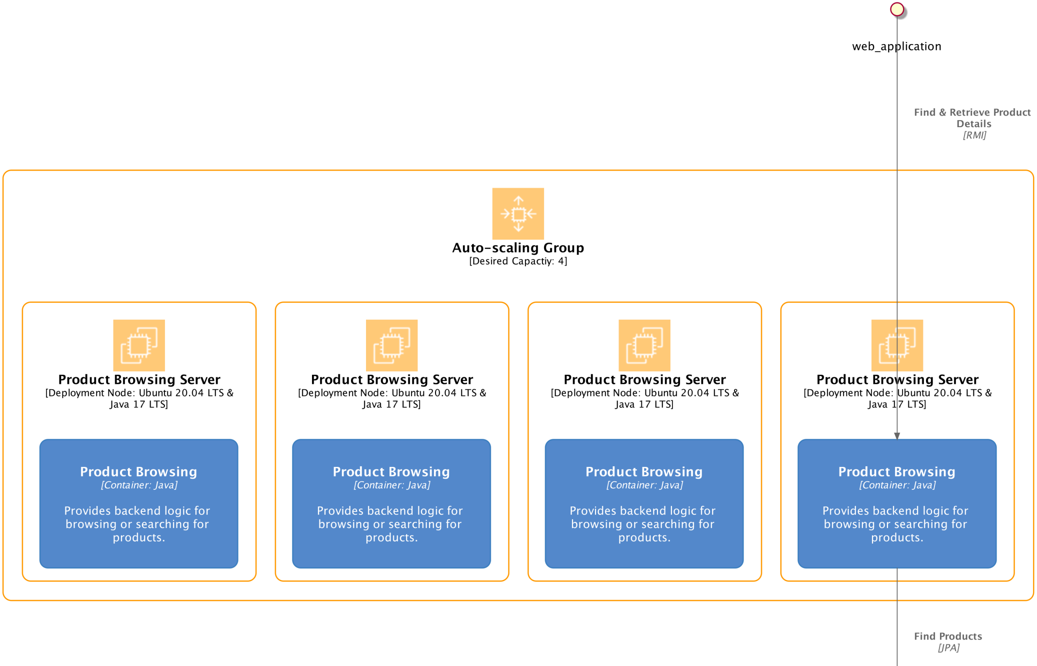

Sahara
[Deployment Node: Sahara eCommerce Data Centre]
Customers Computer
[Deployment Node: MS Windows, Apple MacOS or Linux]
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view producsts.
Delivers content to the customers browser.
[HTTPS]
Web Server
[Deployment Node: Ubuntu 20.04 LTS]
Apache TomEE
[Deployment Node: Apache TomEE 8.0]
Web Application
[Container: J2EE Server]
Delivers web front-end for the on-line store.
Find & Retrieve Product Details
[RMI]
Purchase Products
[RMI]
CRUD Customer Accounts
[RMI]
Query Order Status
[RMI]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Purchasing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Purchasing
[Container: Java]
Provides backend logic for purchasing products.
Customer Account Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Customer Account Management
[Container: Java]
Provides backend logic for customer account management.
Product Fulfillment Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Fulfillment
[Container: Java]
Provides backend logic for fulfilling products.
Find Products
[JPA]
Updates & Queries
[JPA]
Updates & Queries
[JPA]
Updates
[JPA]
Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]
MySQL Primary
[Deployment Node: MySQL 8.0]
Application Database
[Container: MySQL]
Stores customer credentials, products, and orders.

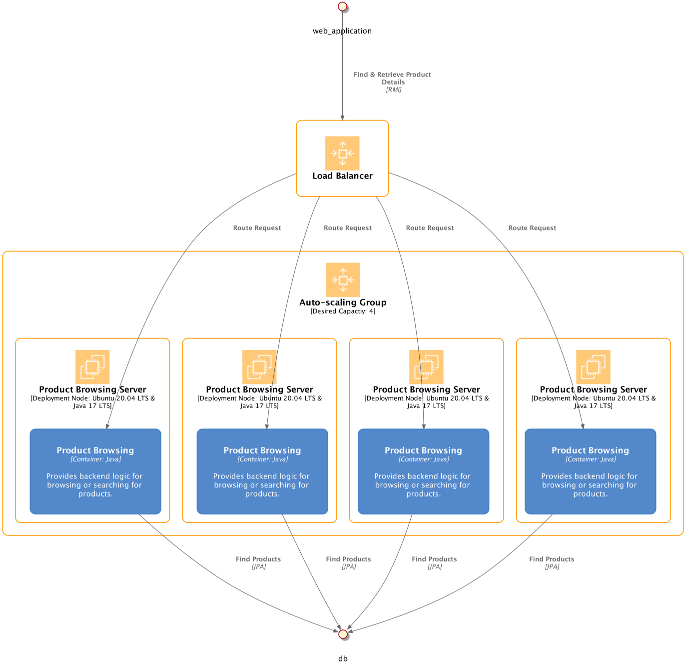

Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Find & Retrieve Product Details
[RMI]
Find Products
[JPA]
web_application
db



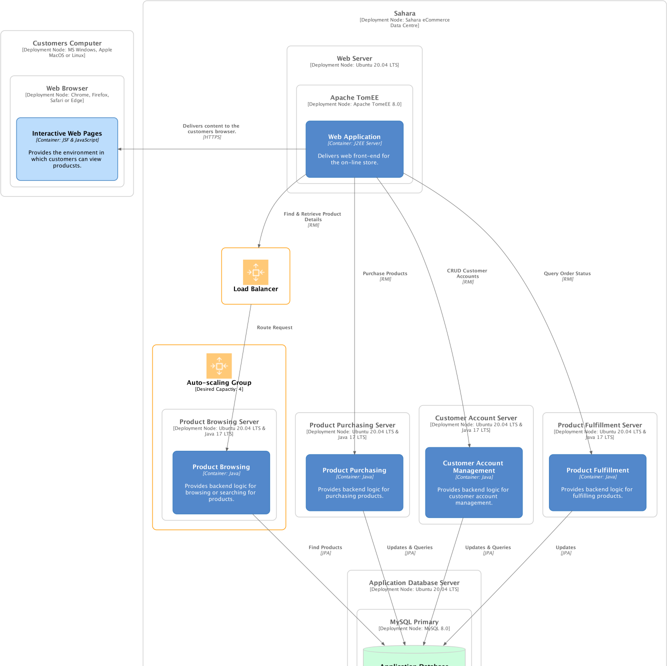
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Find & Retrieve Product Details
[RMI]
Find Products
[JPA]
web_application
db


Auto-scaling Group
[Desired Capacity: 4]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Find & Retrieve Product Details
[RMI]
Find Products
[JPA]
web_application
db


web_application
Find & Retrieve Product Details
[RMI]
Auto-scaling Group
[Desired Capactiy: 4]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Find Products
[JPA]



web_application
Find & Retrieve Product Details
[RMI]
Load Balancer
Route Request
Route Request
Route Request
Route Request
Auto-scaling Group
[Desired Capactiy: 4]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Find Products
[JPA]
Find Products
[JPA]
Find Products
[JPA]
Find Products
[JPA]
db


Sahara
[Deployment Node: Sahara eCommerce Data Centre]
Customers Computer
[Deployment Node: MS Windows, Apple MacOS or Linux]
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view producsts.
Web Server
[Deployment Node: Ubuntu 20.04 LTS]
Apache TomEE
[Deployment Node: Apache TomEE 8.0]
Web Application
[Container: J2EE Server]
Delivers web front-end for the on-line store.
Delivers content to the customers browser.
[HTTPS]
Find & Retrieve Product Details
[RMI]
Load Balancer
Purchase Products
[RMI]
CRUD Customer Accounts
[RMI]
Query Order Status
[RMI]
Route Request
Auto-scaling Group
[Desired Capactiy: 4]
Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Purchasing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Purchasing
[Container: Java]
Provides backend logic for purchasing products.
Customer Account Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Customer Account Management
[Container: Java]
Provides backend logic for customer account management.
Product Fulfillment Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]
Product Fulfillment
[Container: Java]
Provides backend logic for fulfilling products.
Find Products
[JPA]
Updates & Queries
[JPA]
Updates & Queries
[JPA]
Updates
[JPA]
Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]
MySQL Primary
[Deployment Node: MySQL 8.0]

In Summary

Simplicity

Reliability

Scalability

In Summary

Simplicity *Minimal network communication* (compared to *other* distributed systems), less impacted by fallacies.

Reliability

Scalability

In Summary

Simplicity *Minimal network communication* (compared to *other* distributed systems), less impacted by fallacies.

Reliability Traffic is spread to various services, still *partially operational* if one goes down. Auto-scaling allows for *basic replication*.

Scalability

In Summary

Simplicity *Minimal network communication* (compared to *other* distributed systems), less impacted by fallacies.

Reliability Traffic is spread to various services, still *partially operational* if one goes down. Auto-scaling allows for *basic replication*.

Scalability Auto-scaling and load balancing allows *individual services to scale*. However, the *database is a bottleneck*.