

Distributed Computing I

Software Architecture

Brae Webb

March 27, 2023



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

Previously in CSSE6400...

Service-based Architecture

Previously in CSSE6400...

Simplicity For a distributed system



Modularity Services



Extensibility New services



Deployability Independent services



Testability Independent services



Security API layer



Reliability Independent services



Interoperability Service APIs



Scalability Coarse-grained services



Previously in CSSE6400...

Simplicity For a distributed system



Reliability Independent services



Scalability Coarse-grained services



Previously in CSSE6400...

Simplicity *For a distributed system*



Previously in CSSE6400...

Simplicity



§ *Fallacies*

Question

What is a *fallacy*?

Definition 1. Fallacy

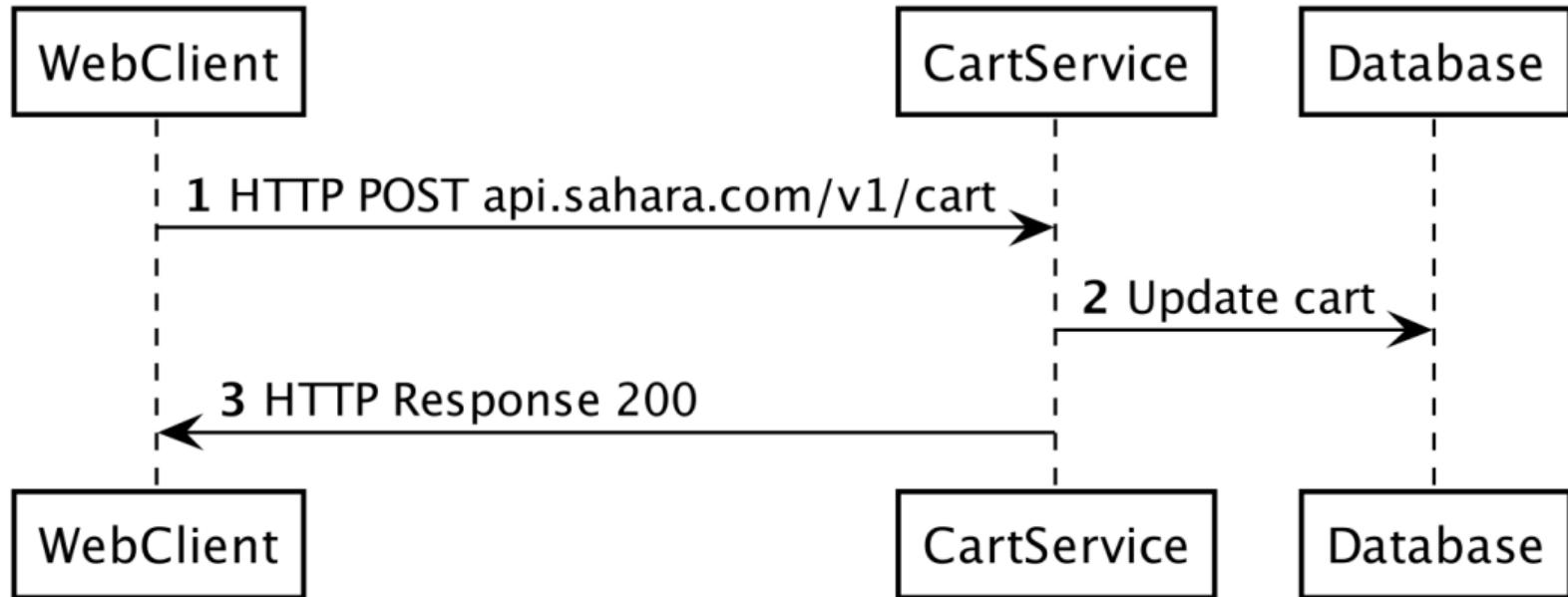
Something that is believed or assumed to be true but is not.

A few reasons for complexity

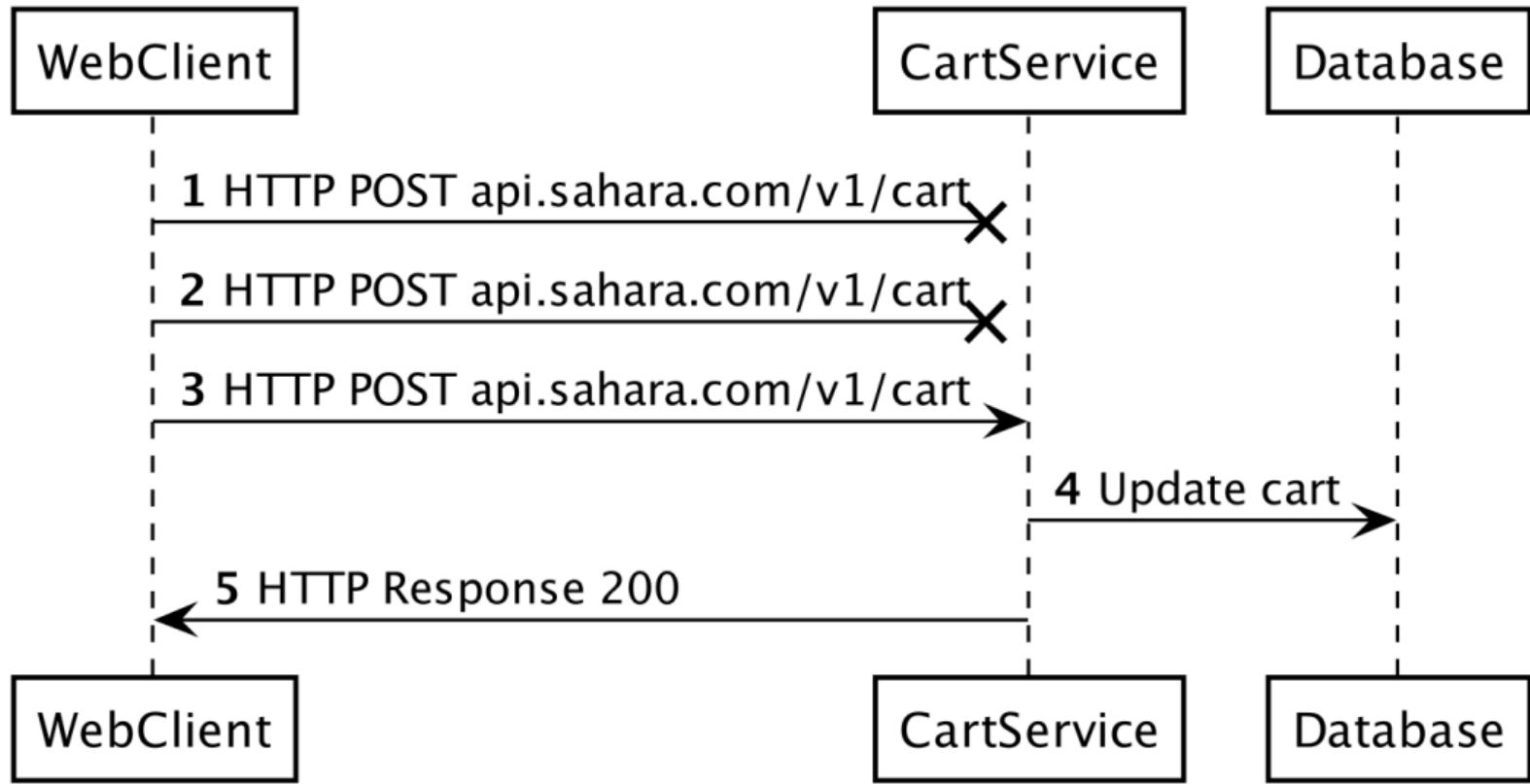
The Fallacies of *Distributed Computing*

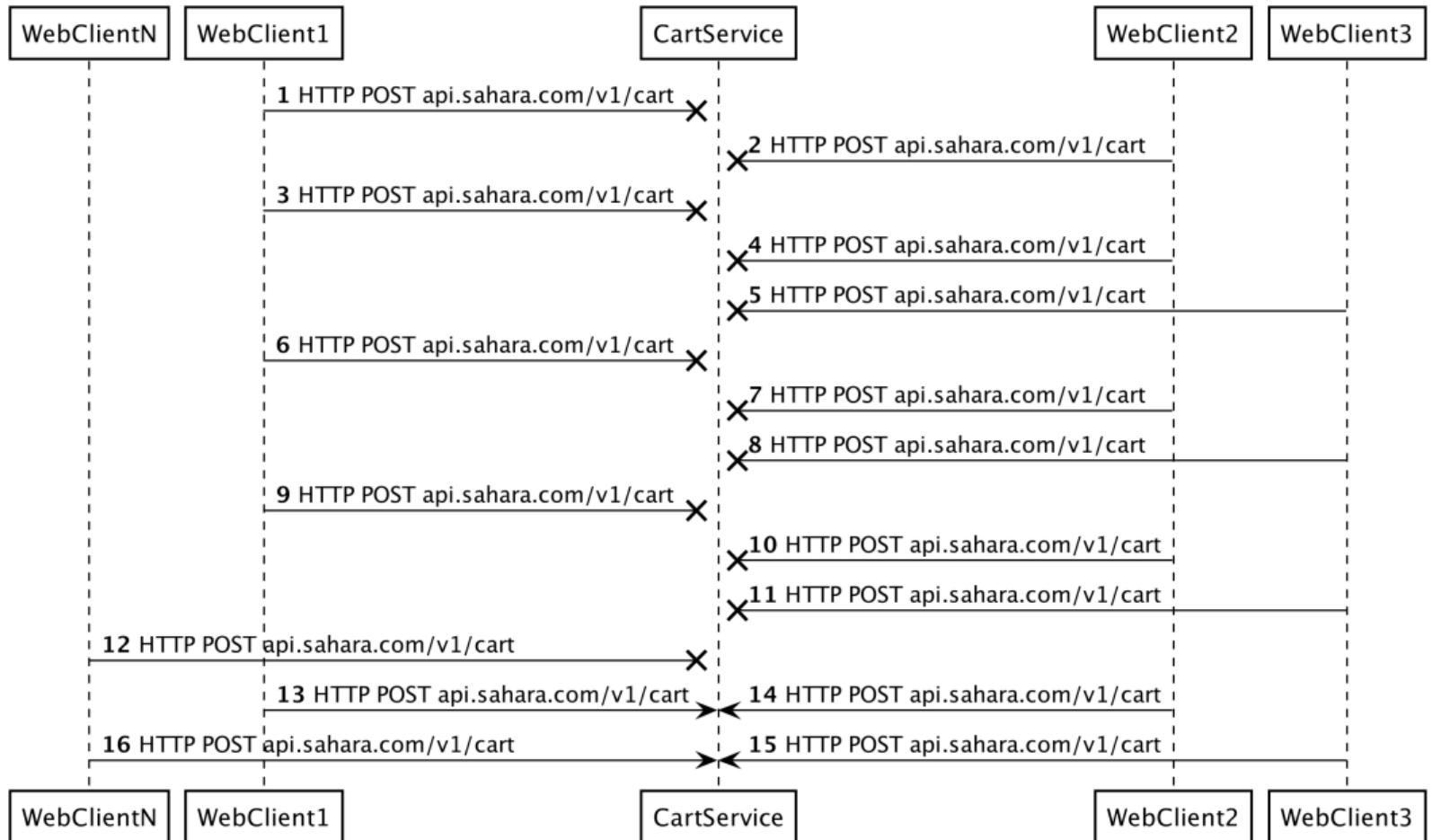
Fallacy #1

The network is reliable



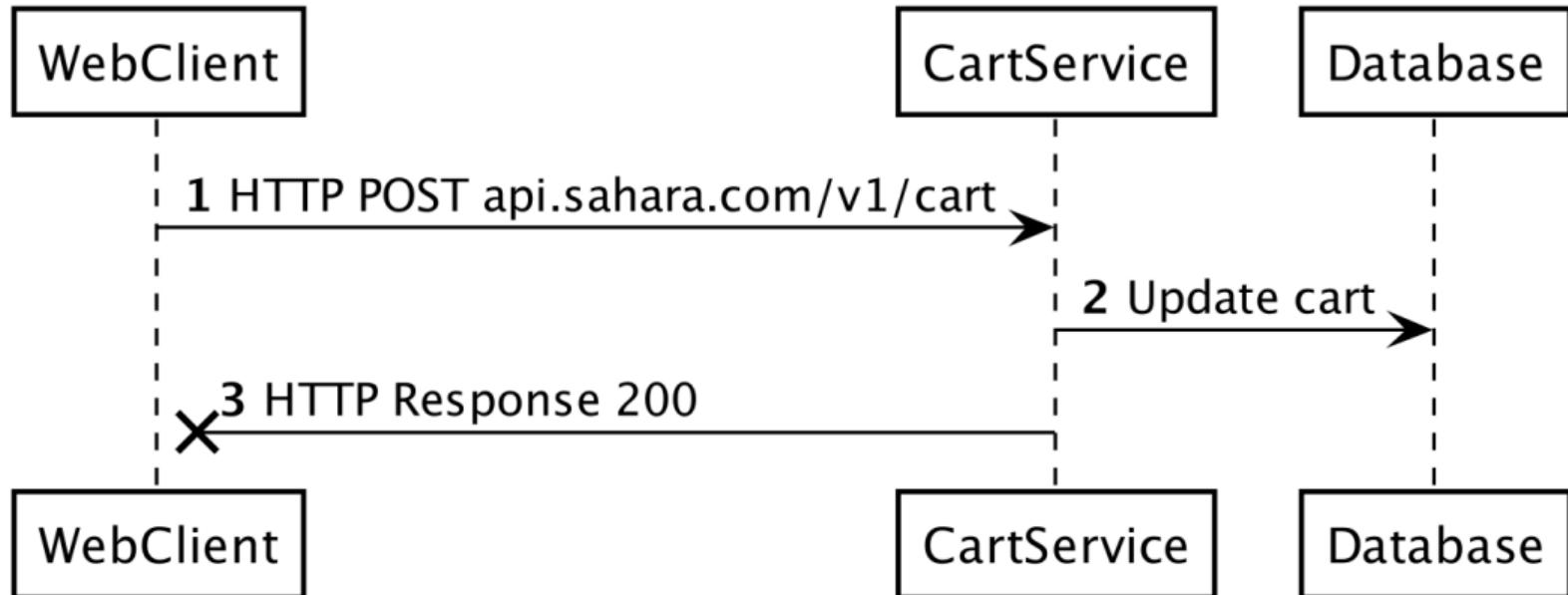


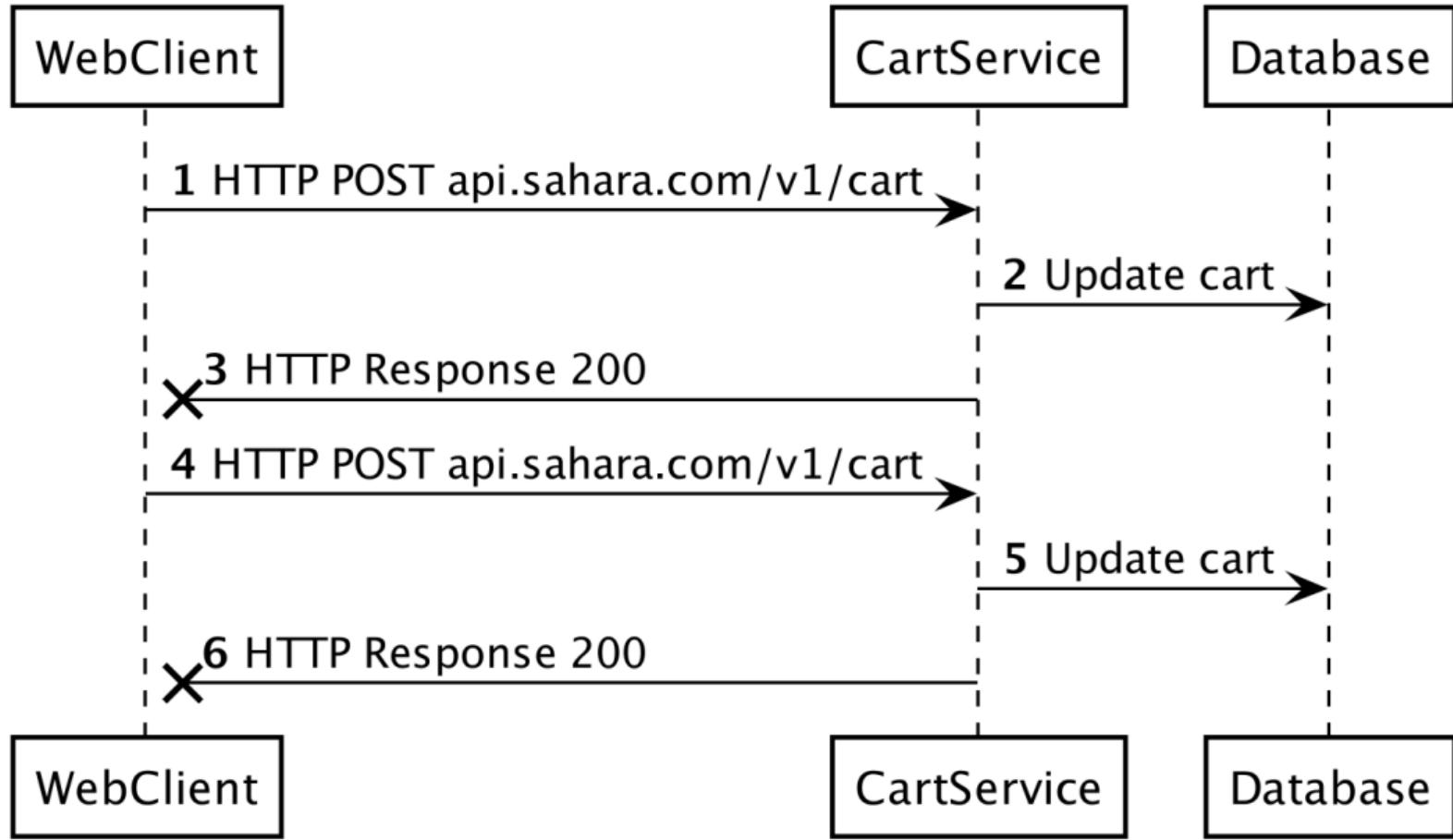


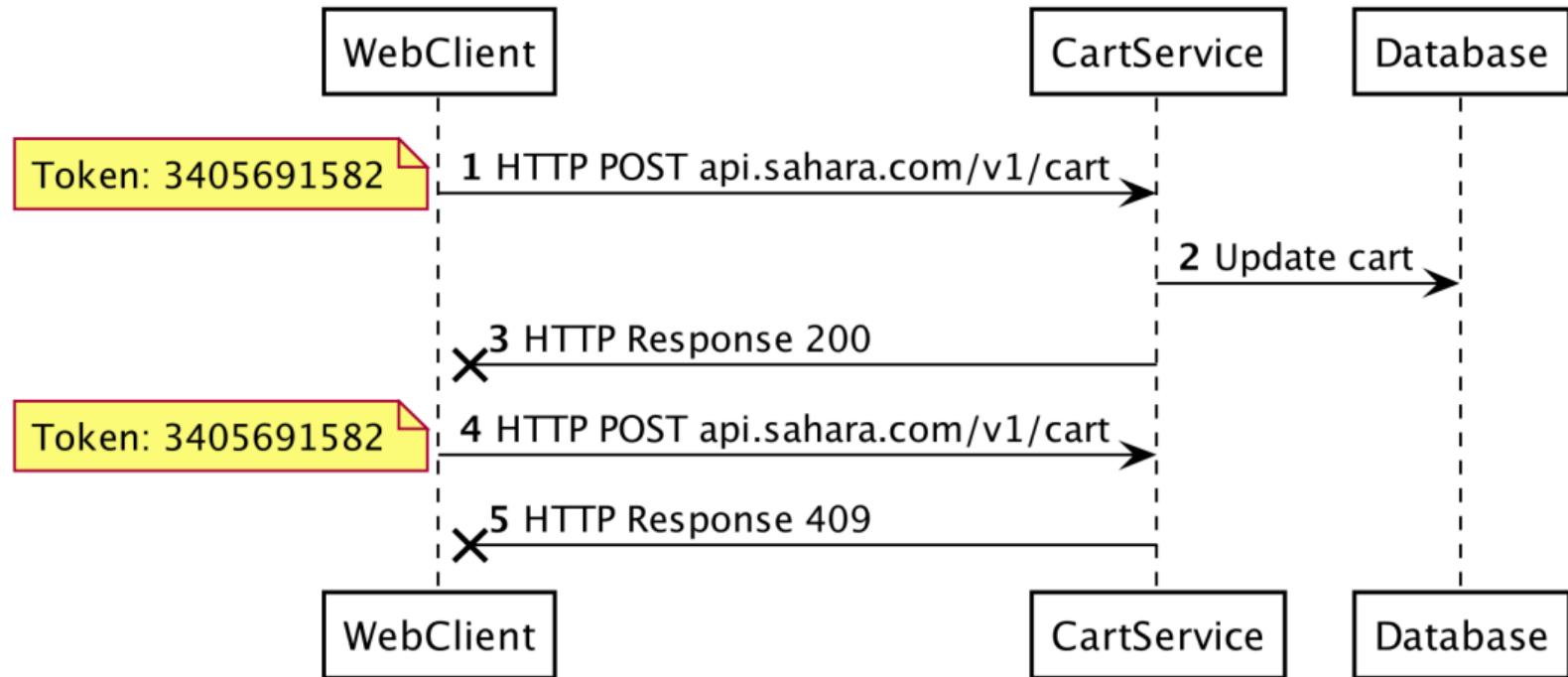


Exponential backoff

```
1  retry = True
2  do:
3      status = service.request()
4
5      if status != SUCCESS:
6          wait(2 ** retries)
7      else:
8          retry = False
9  while (retry and retries < MAX_RETRIES)
```







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

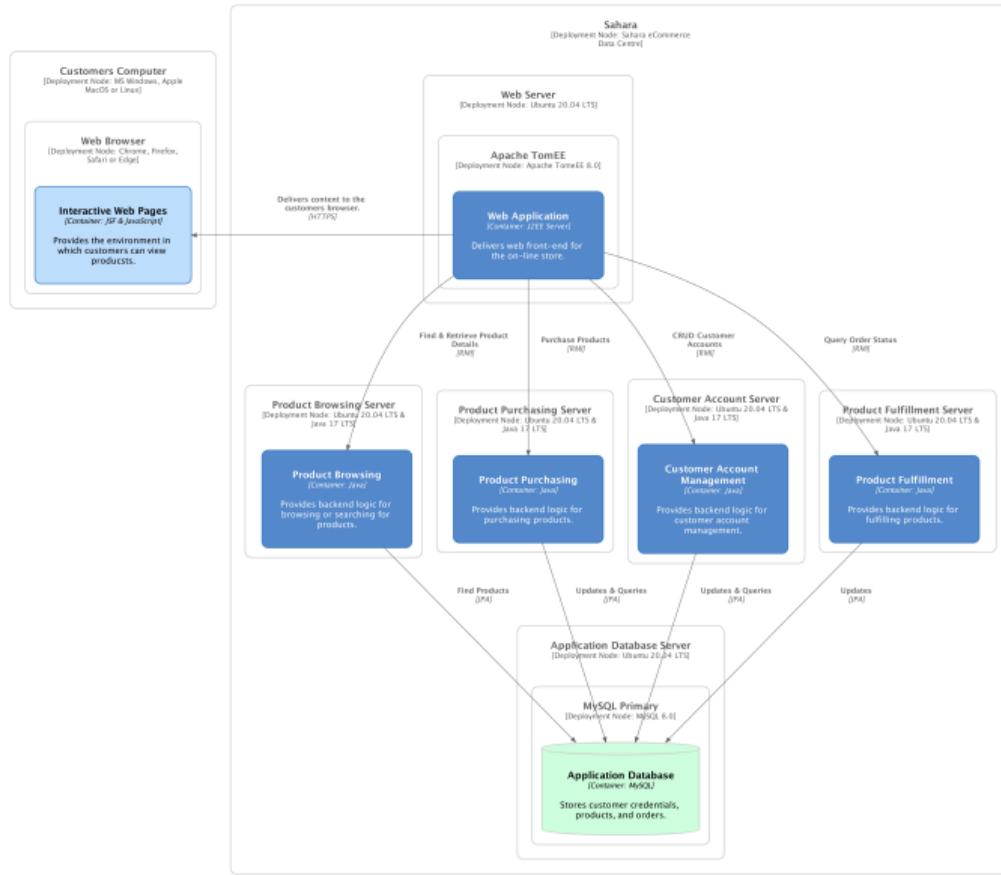
EC2 to EC2 0.662ms

Fallacy #3

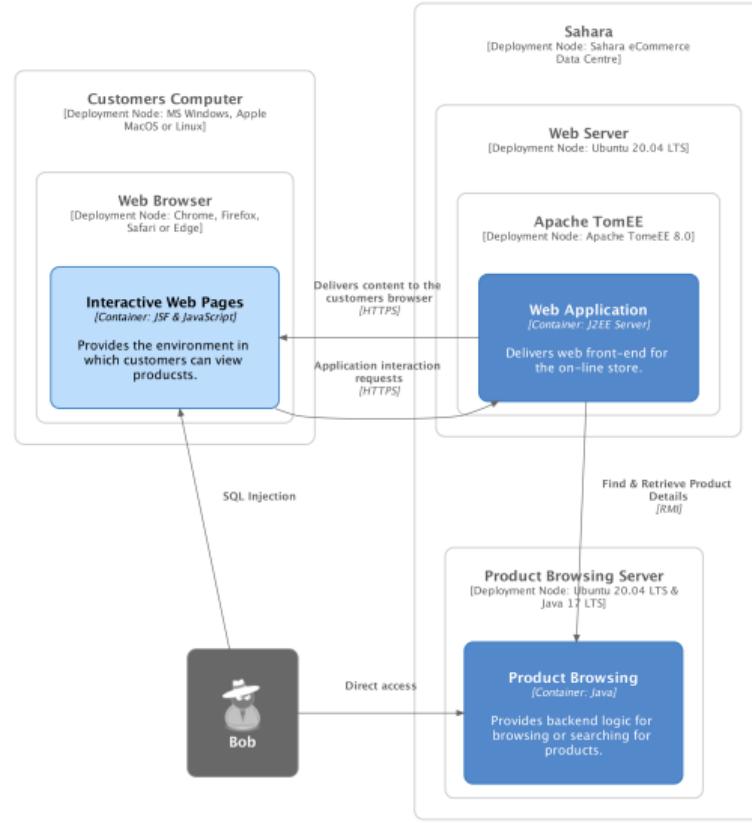
Bandwidth is infinite

Fallacy #4

The network is secure



Legend



Fallacy #5

The topology never changes

Fallacy #6

There is only one administrator

Scenario

- Deployments are banned on the weekend.

Scenario

- Deployments are banned on the weekend.
- Sunday night users start complaining.

Scenario

- Deployments are banned on the weekend.
- Sunday night users start complaining.
- There have been no deployments since Friday.

Scenario

- Deployments are banned on the weekend.
- Sunday night users start complaining.
- There have been no deployments since Friday.
- You can still access the system.

Scenario

- Deployments are banned on the weekend.
- Sunday night users start complaining.
- There have been no deployments since Friday.
- You can still access the system.
- Who do you talk to?

Fallacy #7

Transport cost is zero

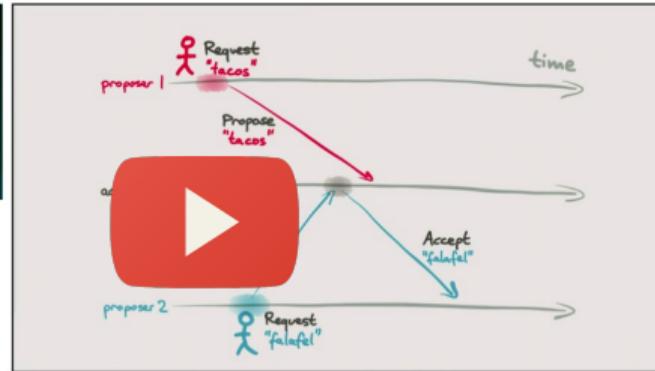
Remember

Distributed systems are *hard*.

Remember

Distributed systems are often *not your friend.*

When you need to, prove it



Sept 13-14, 2019
thestrangeloop.com

Previously in CSSE6400...

Simplicity For a distributed system



Reliability Independent services



Scalability Coarse-grained services



Previously in CSSE6400...

Reliability Independent services



Question

What makes software *reliable*?

‘Working’ software

Satisfies the functional requirements

Definition 2. Reliable Software

Continues to work, even when things go wrong.

Definition 3. 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*

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.

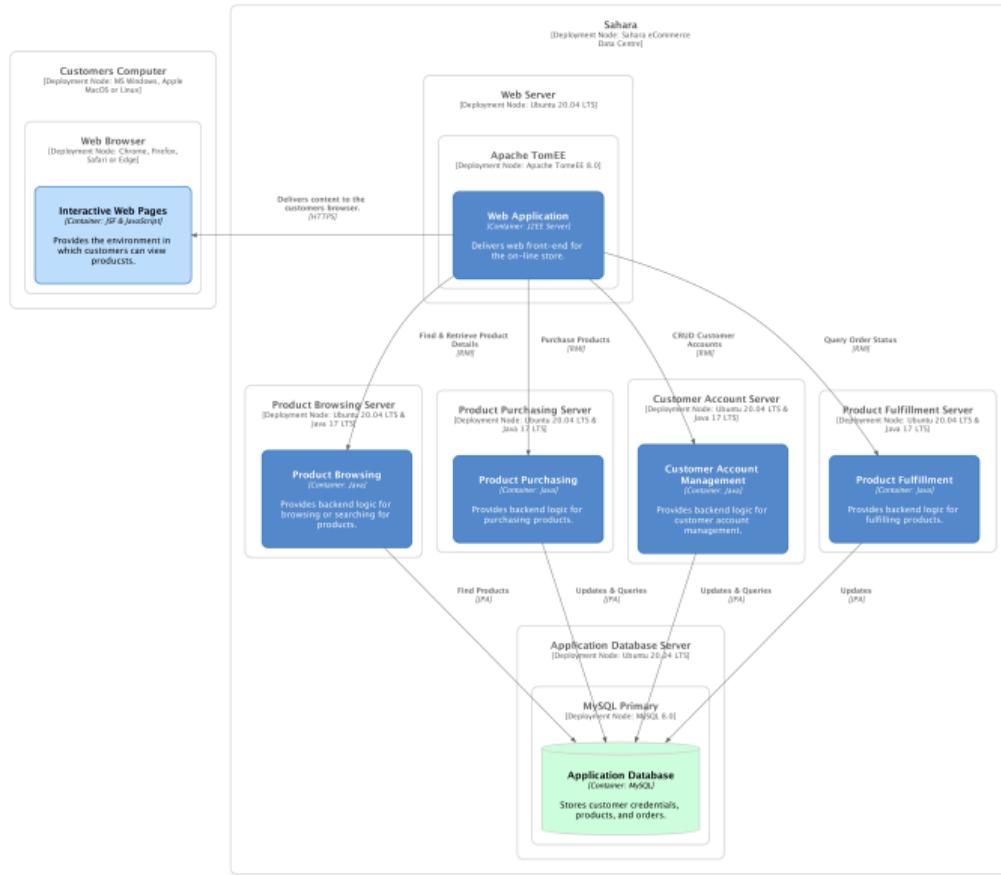
Adding *100* computers reduces the cuts the risk by *100*.

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 cuts the risk by *100*.

Of course, there are other reasons you might want run software on multiple computers.



Legend

Previously in CSSE6400...

Simplicity For a distributed system



Reliability Independent services



Scalability Coarse-grained services



Previously in CSSE6400...

Scalability Coarse-grained services



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 minimal and maximum amount of instances.

What we really want

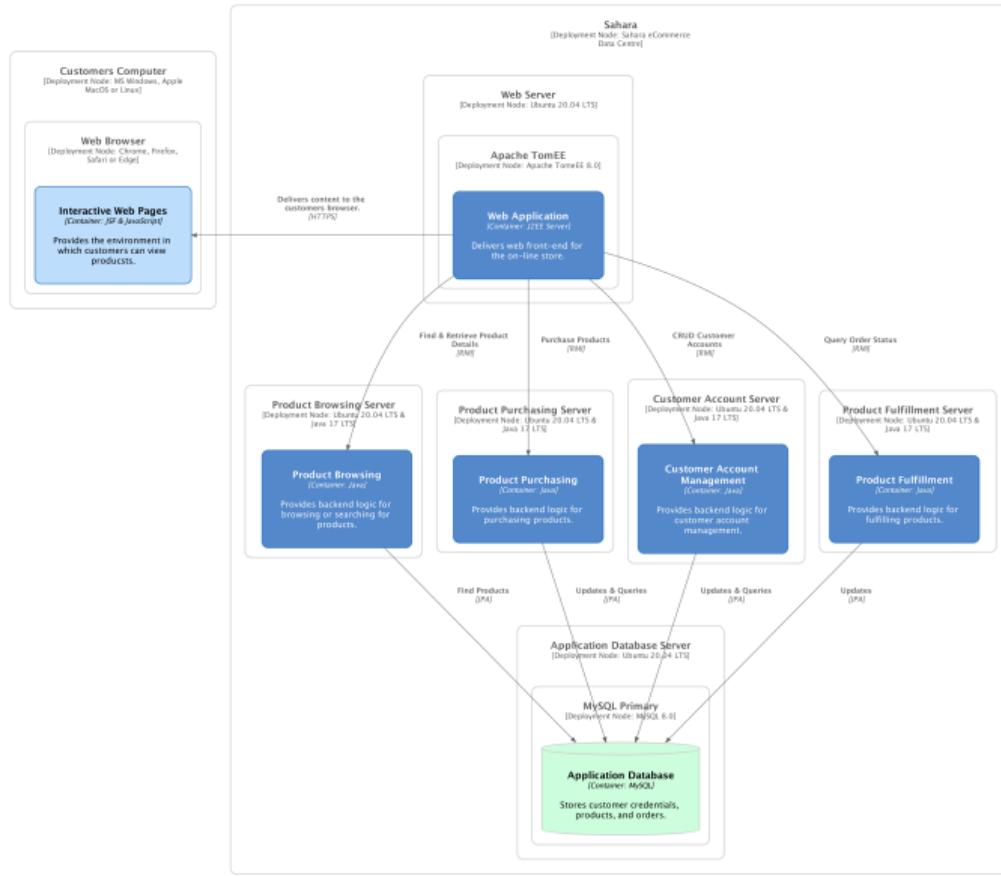
Desired Capacity Amount of *healthy* instances we want to have in an auto-scaling group.

Health check

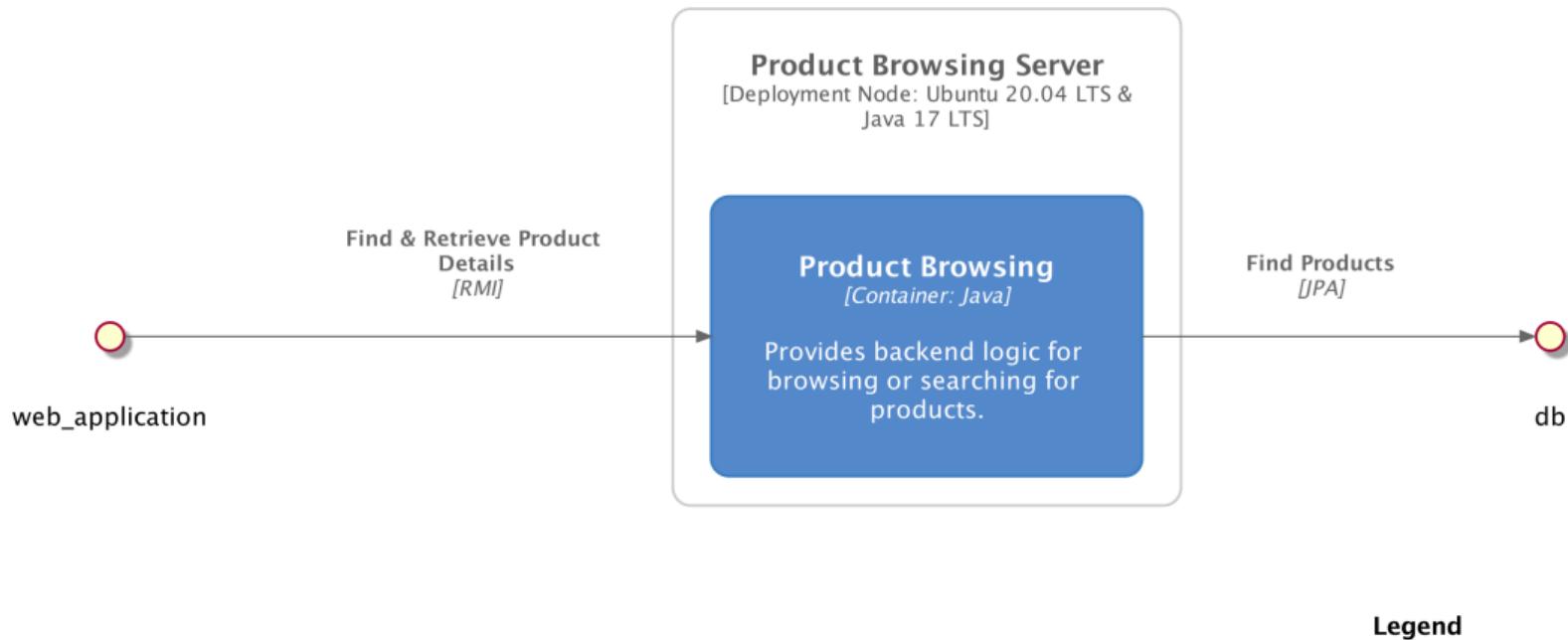
User defined method to determine whether an instance is *healthy*.

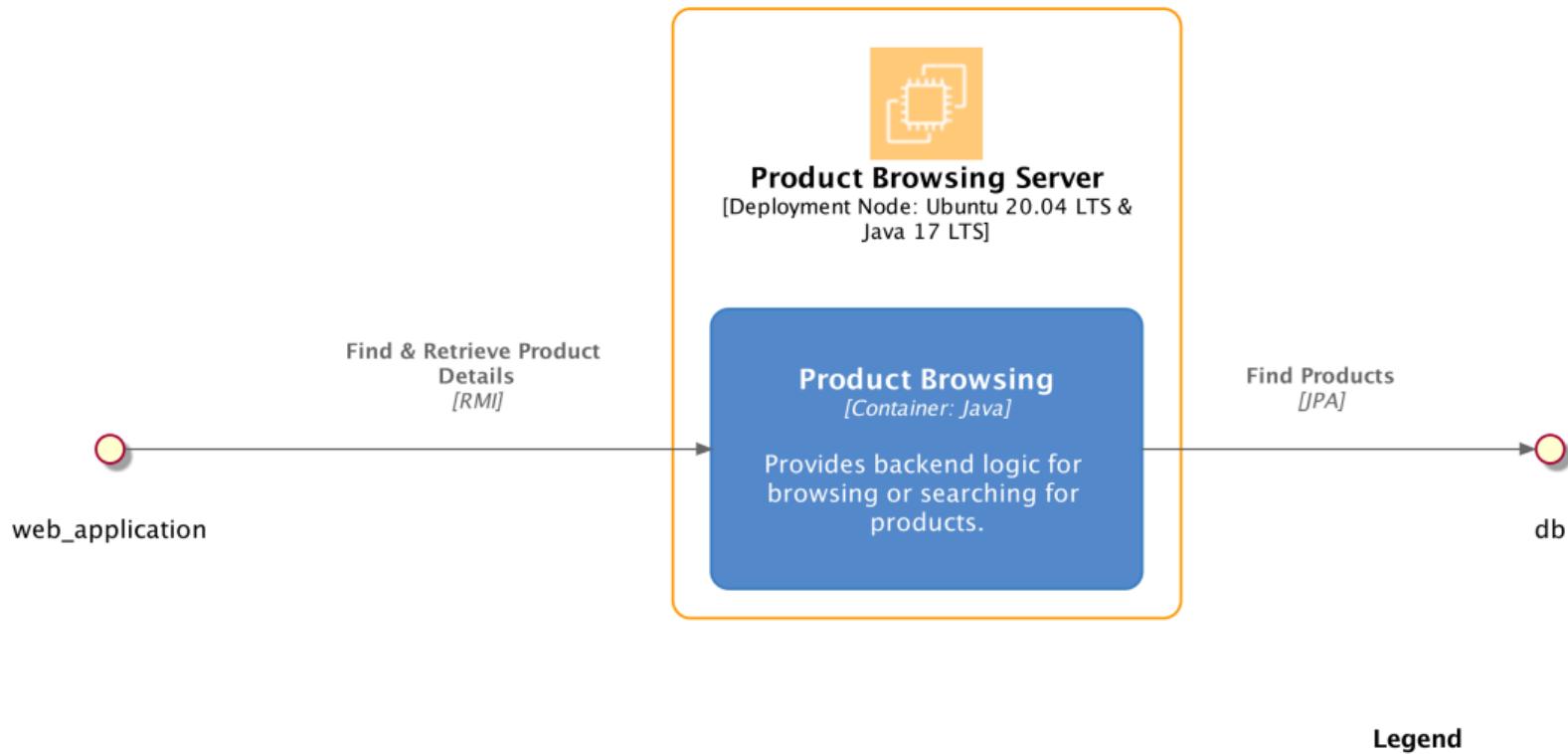
Auto-scaling

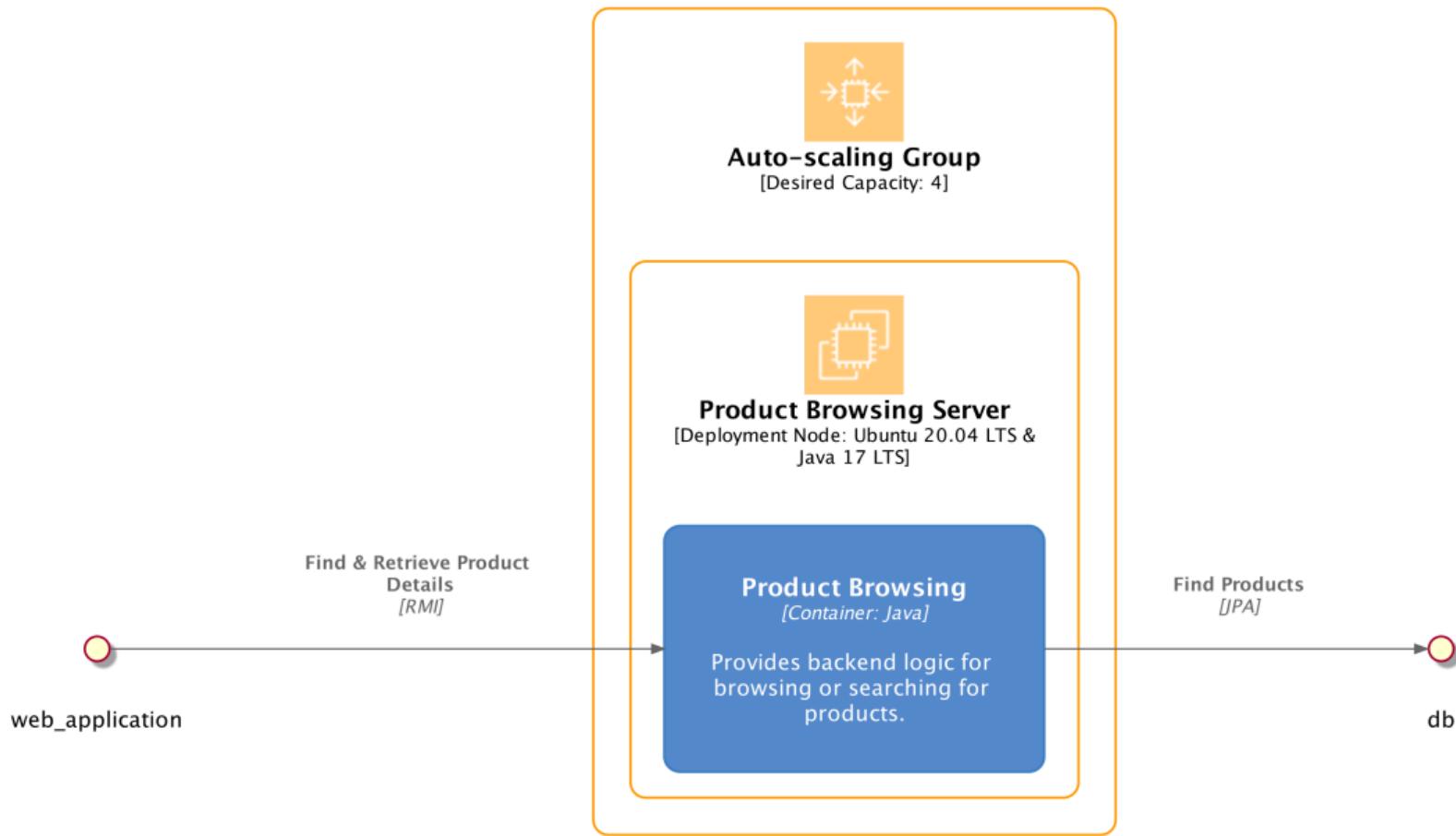
An example

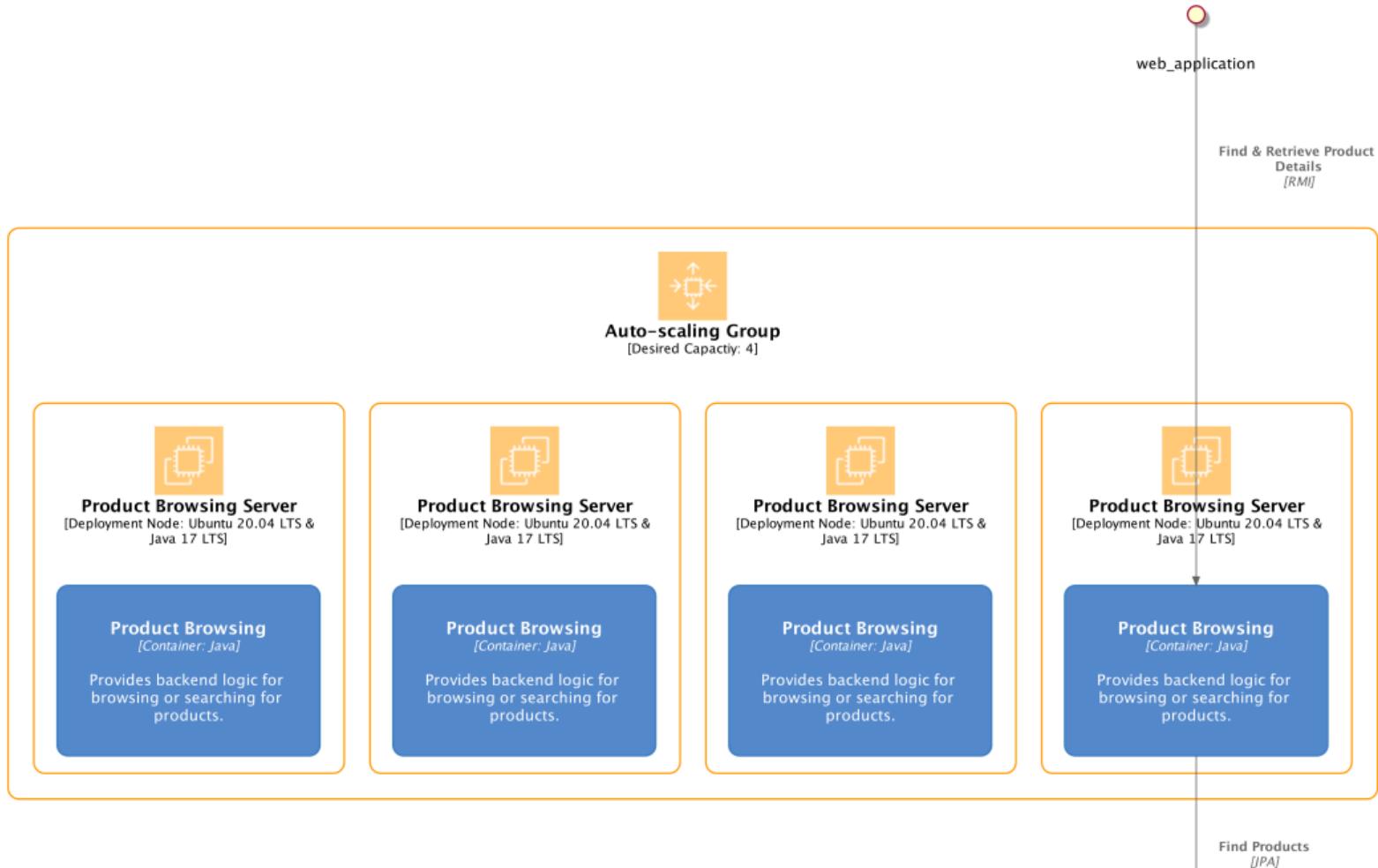


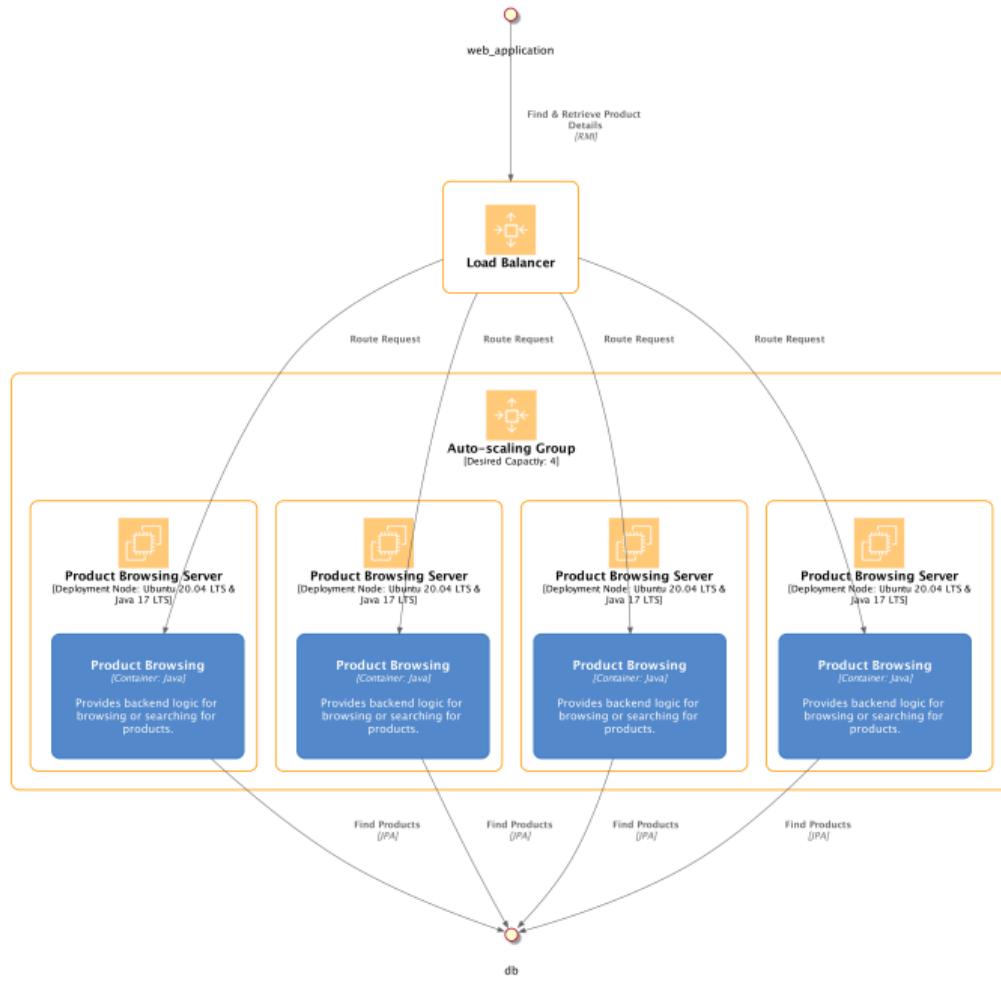
Legend

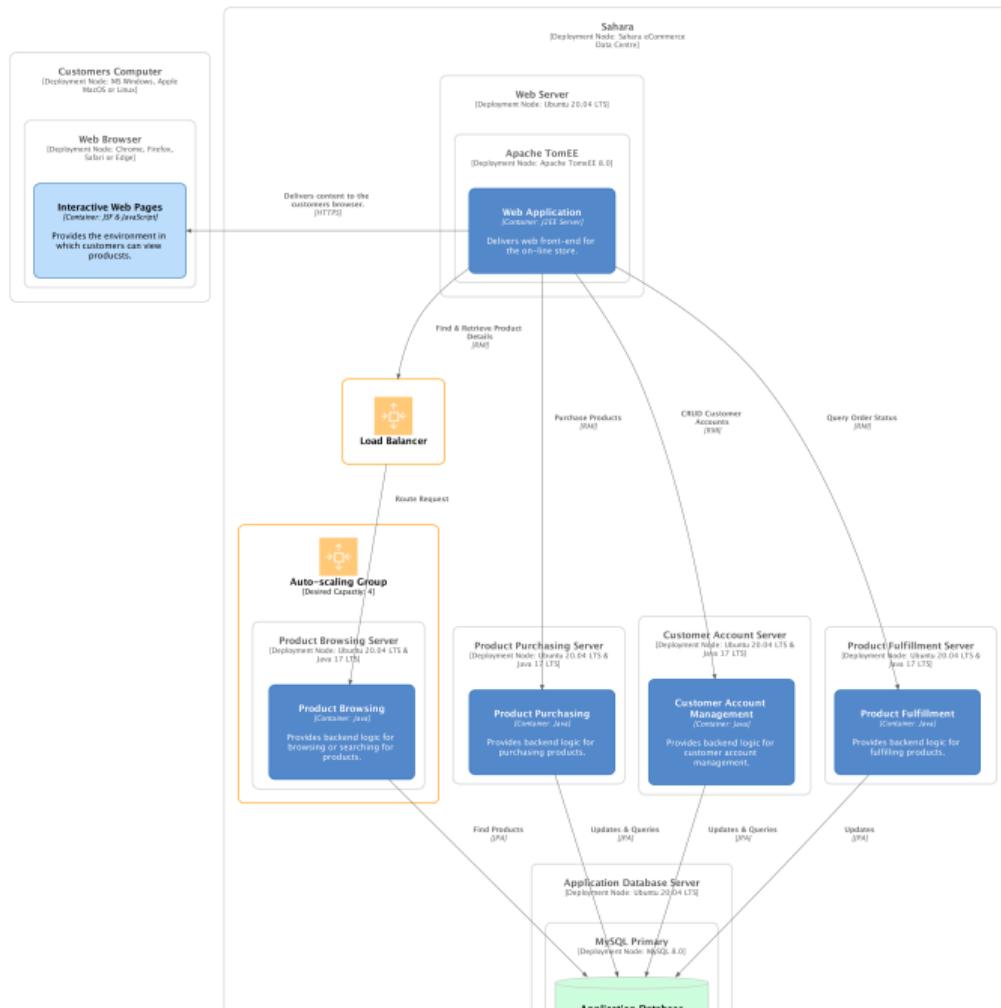












In Summary

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 bottle-neck*.