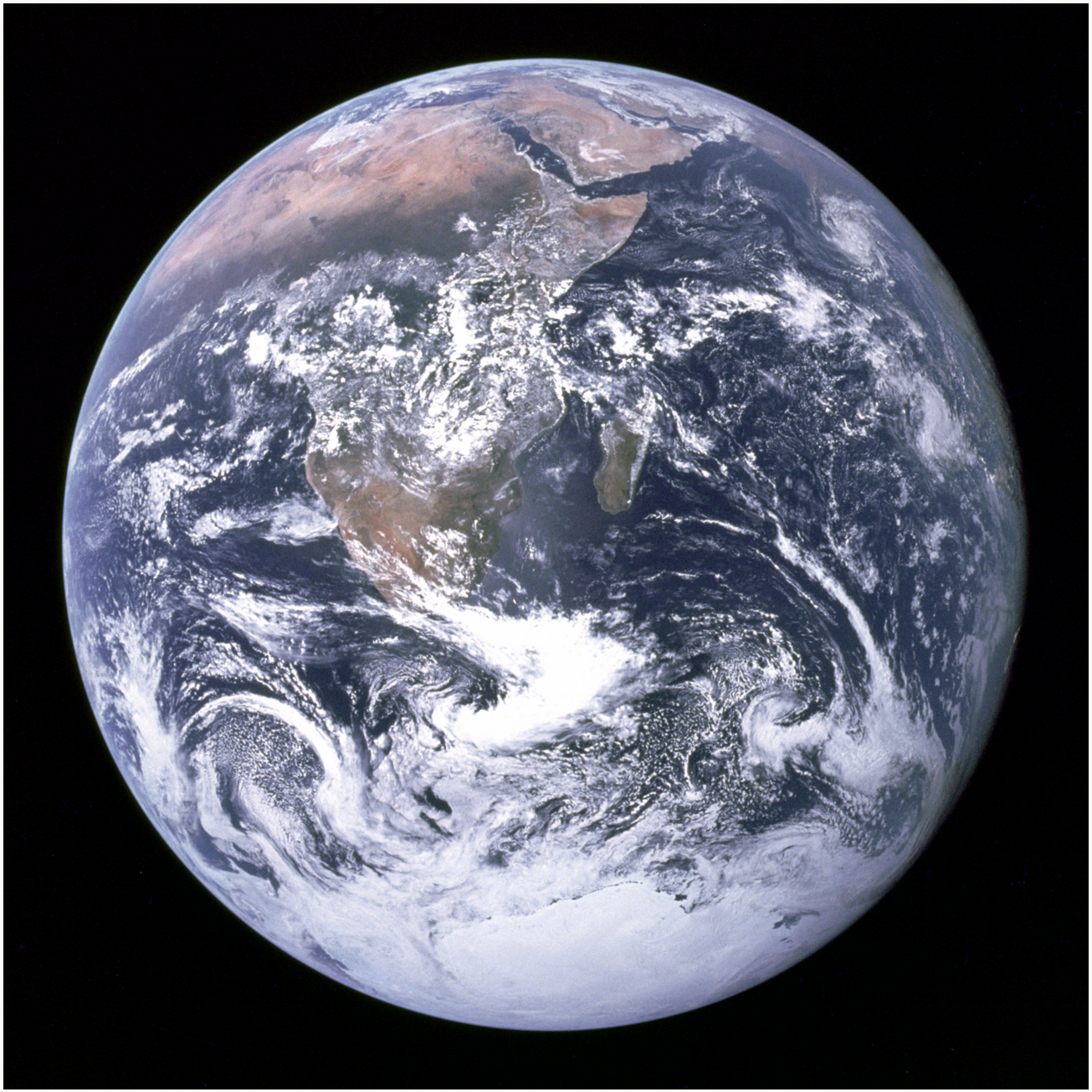
Cassandra: Consistency & Tolerance (a morality guide for databases)

Matt Kennedy, Architect

@thetweetofmatt

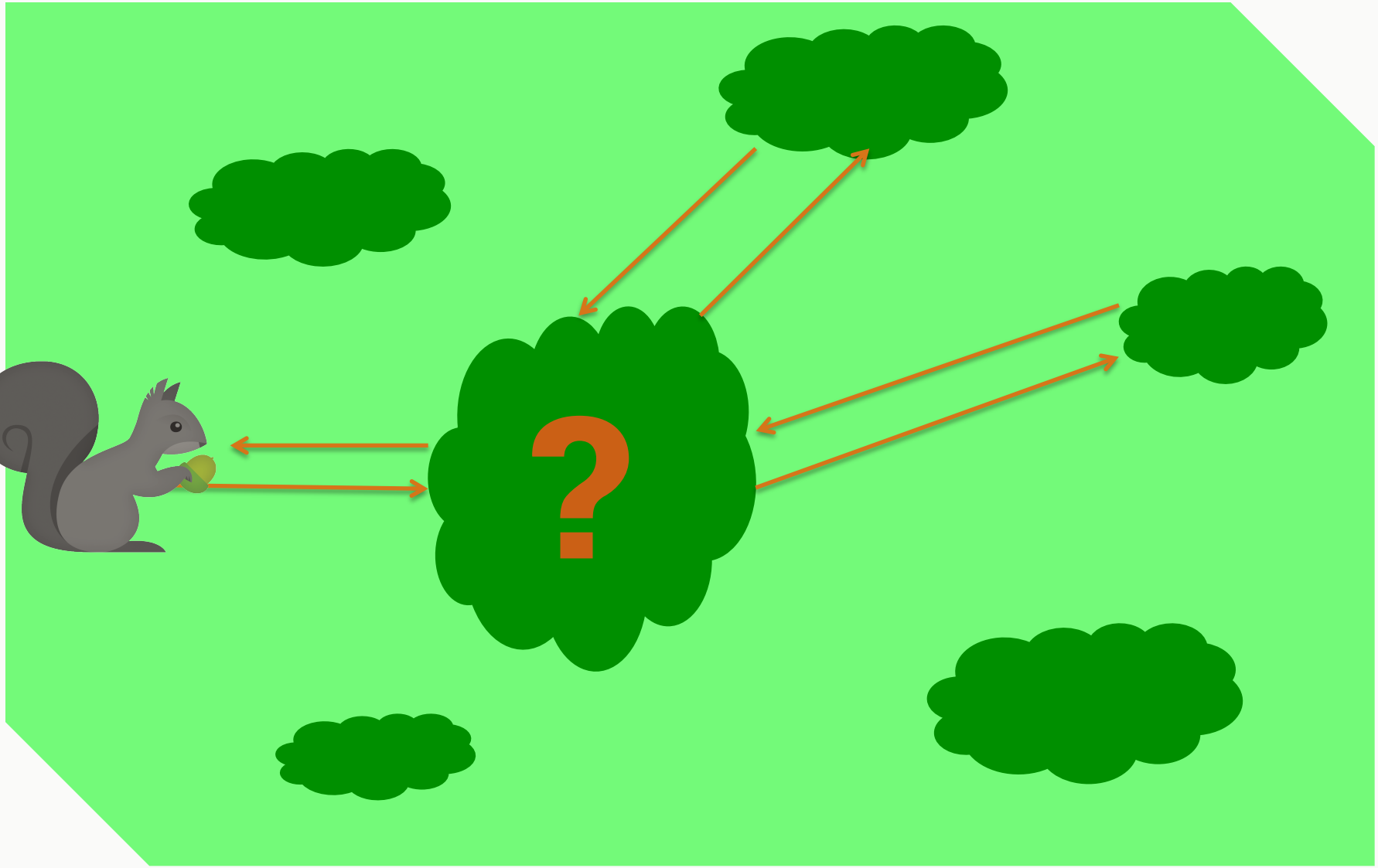
DATASTAX

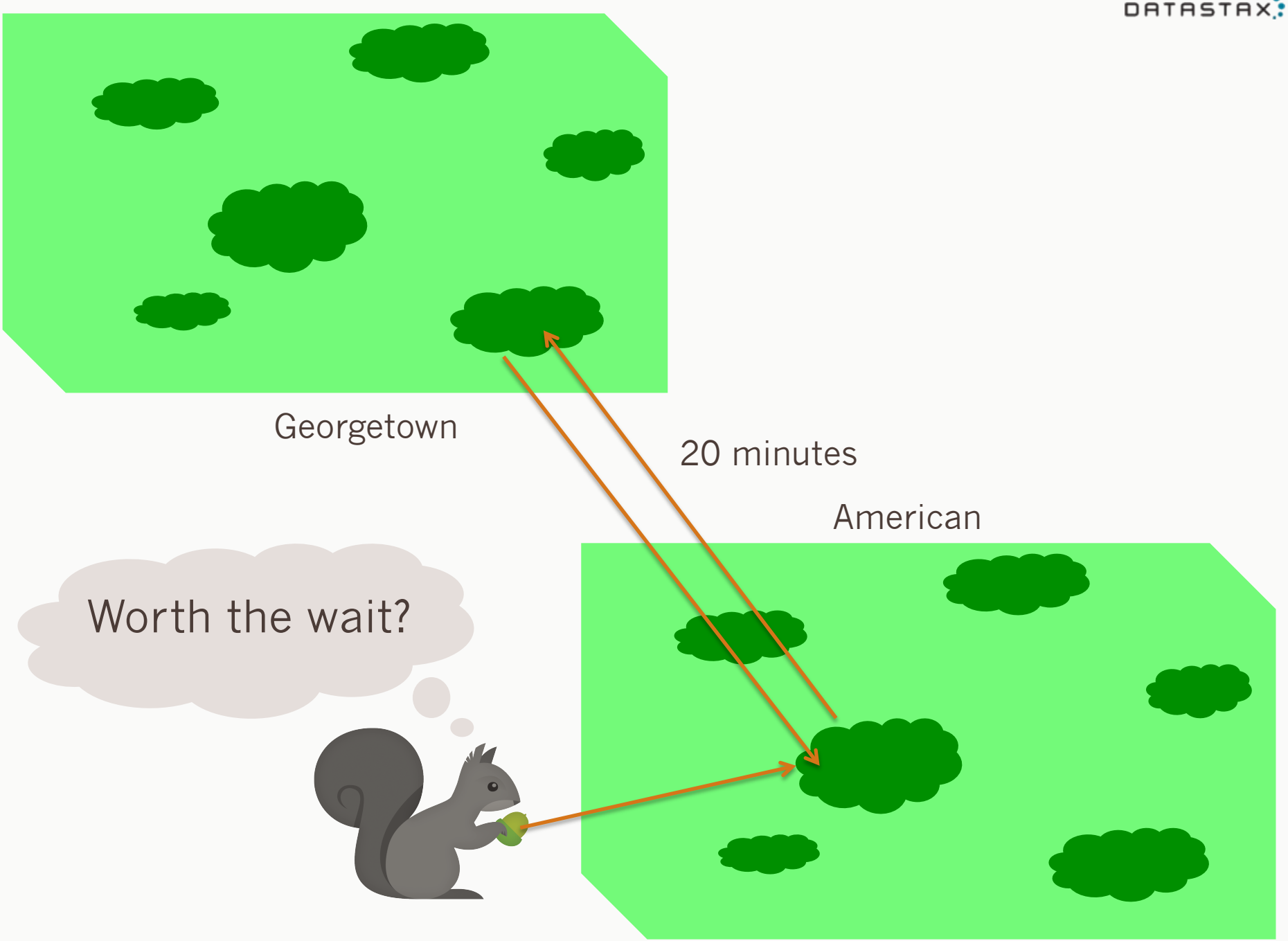


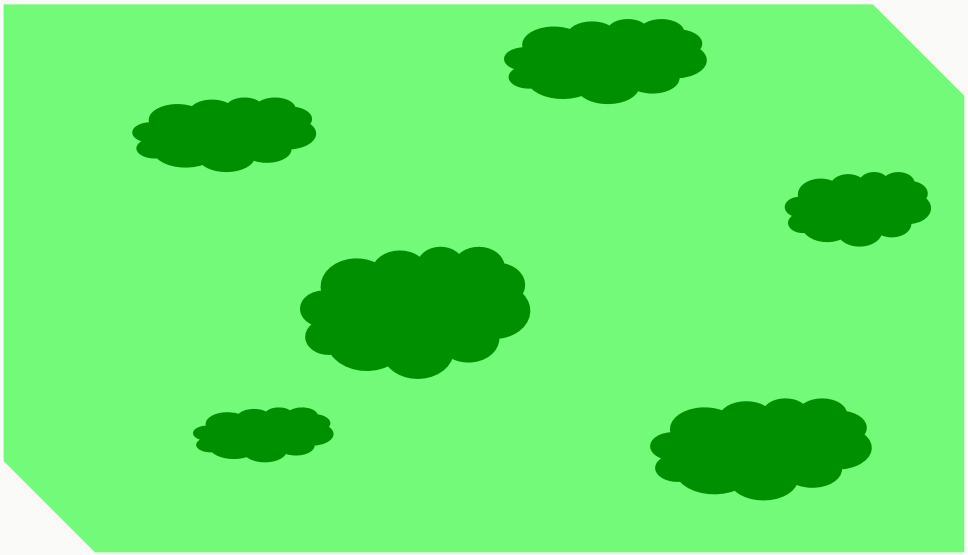

GEORGETOWN


GEORGETOWN
55


HOYAS
3







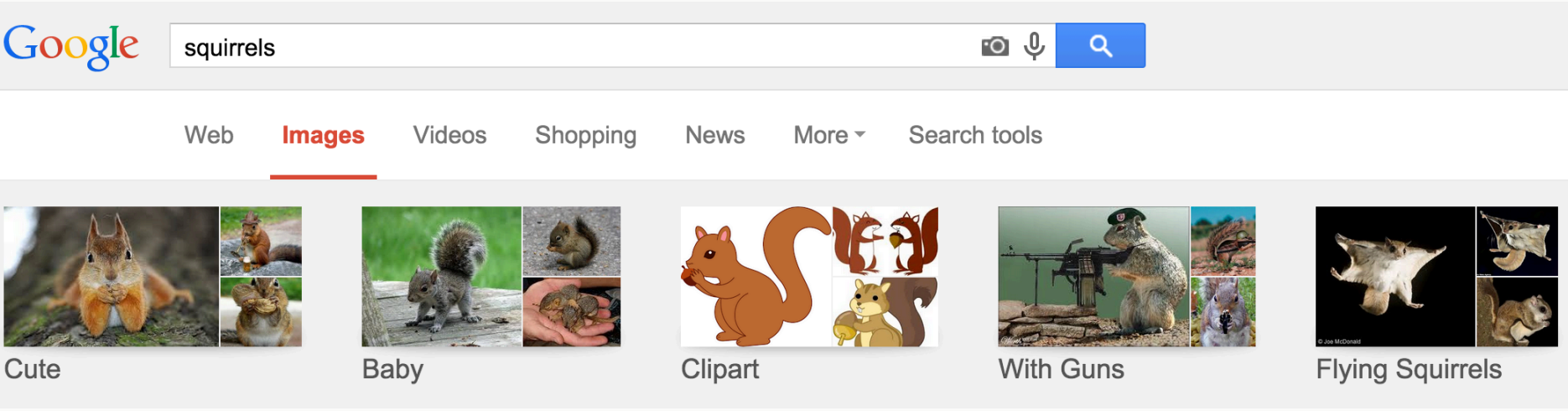
Georgetown

Lessons?

- High Availability means:
  - Have to maintain some redundancy
  - Need to distribute your stuff

- Distributed Stuff means:
  - Need to accommodate unreliable networks
  - Need to have strategies for different path costs
  - Plan for outright failures
  - Cooperation!

Also, don't underestimate a squirrel with a Jesuit education.



Eventual Consistency

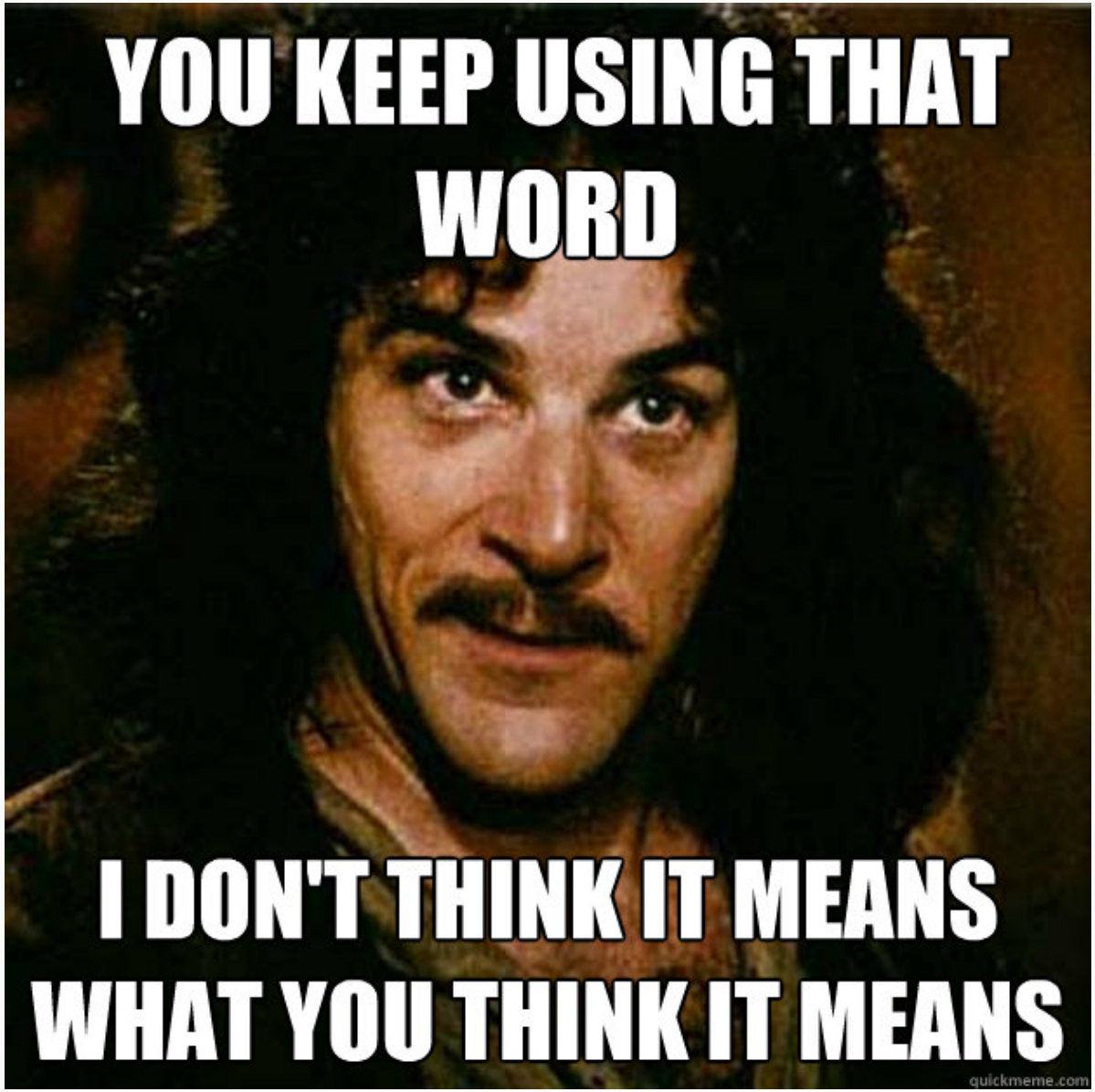
Meh.

http://nekrocodile-izumi.deviantart.com/art/Lazy-college-student-322186707

PRAGMATISM

No matter how trustworthy he may appear, do not give the monkey your pistol.

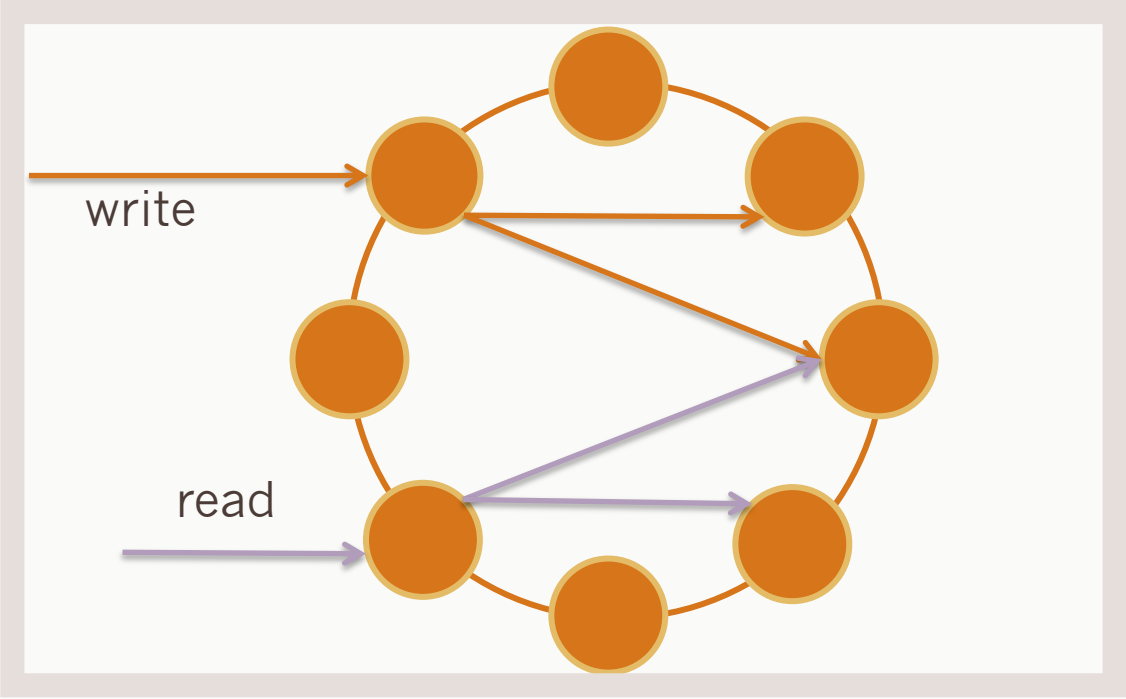
Consistency vs. *Consistency*



Read Consistency Levels

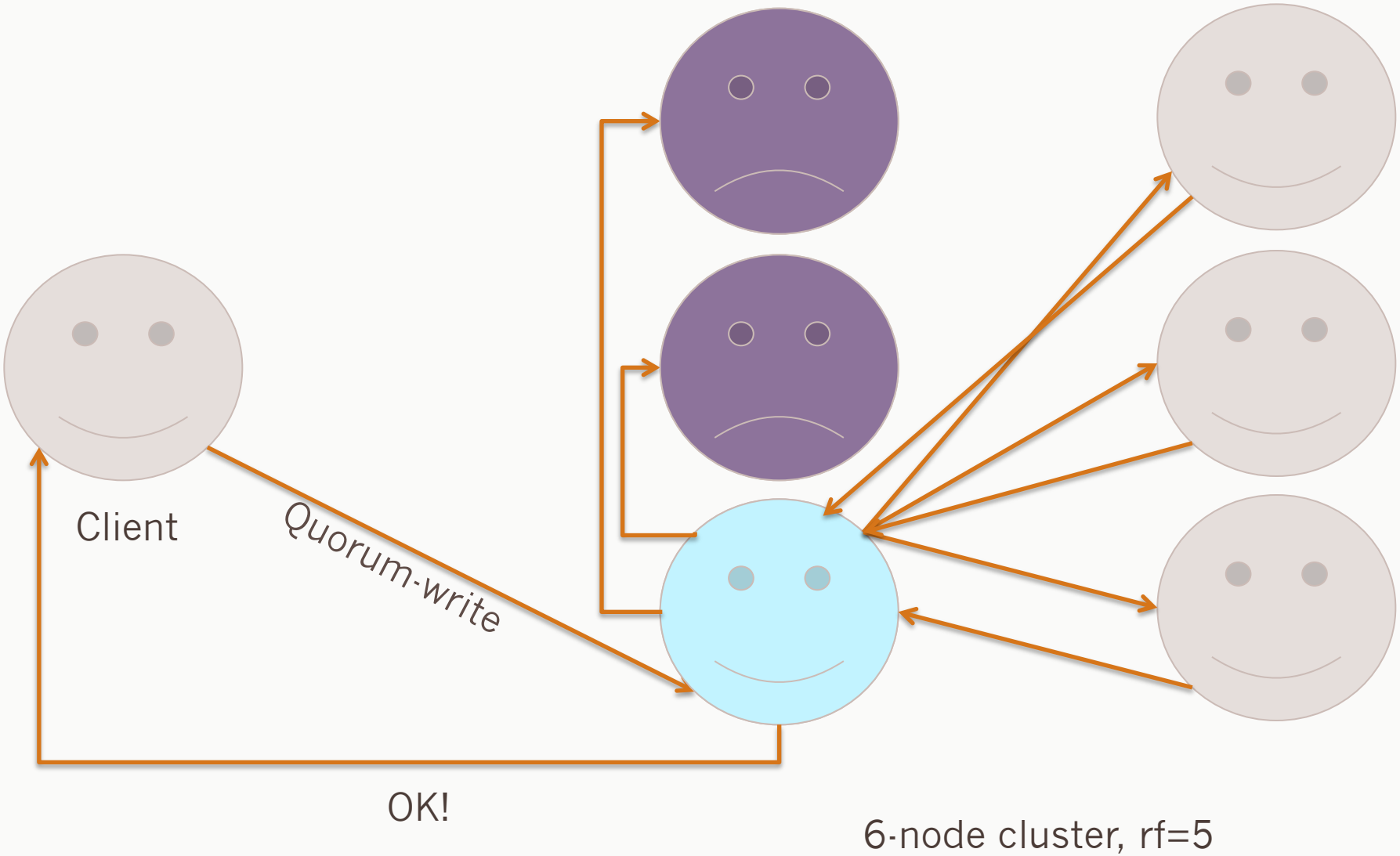
Level	Description	Usage
ALL	Returns the record with the most recent timestamp after all replicas have responded. The read operation will fail if a replica does not respond.	Provides the highest consistency of all levels and the lowest availability of all levels.
EACH_QUORUM	Returns the record with the most recent timestamp once a quorum of replicas in each data center of the cluster has responded.	Same as LOCAL_QUORUM
LOCAL_SERIAL	Same as SERIAL, but confined to the data center.	Same as SERIAL
LOCAL_QUORUM	Returns the record with the most recent timestamp once a quorum of replicas in the current data center as the coordinator node has reported. Avoids latency of inter-data center communication.	Used in multiple data center clusters with a rack-aware replica placement strategy (NetworkTopologyStrategy) and a properly configured snitch. Fails when using SimpleStrategy.
LOCAL_ONE	Available in Cassandra 1.2.11 and 2.0.2 and later. Returns a response from the closest replica, as determined by the snitch, but only if the replica is in the local data center.	Same usage as described in the table about write consistency levels.
ONE	Returns a response from the closest replica, as determined by the snitch. By default, a read repair runs in the background to make the other replicas consistent.	Provides the highest availability of all the levels if you can tolerate a comparatively high probability of stale data being read. The replicas contacted for reads may not always have the most recent write.
QUORUM	Returns the record with the most recent timestamp after a quorum of replicas has responded regardless of data center.	Ensures strong consistency if you can tolerate some level of failure.
SERIAL	Allows reading the current (and possibly uncommitted) state of data without proposing a new addition or update. If a SERIAL read finds an uncommitted transaction in progress, it will commit the transaction as part of the read.	To read the latest value of a column after a user has invoked a lightweight transaction to write to the column, use SERIAL. Cassandra then checks the inflight lightweight transaction for updates and, if found, returns the latest data.
TWO	Returns the most recent data from two of the closest replicas.	Similar to ONE.
THREE	Returns the most recent data from three of the closest replicas.	Similar to TWO.

Tunable Consistency & High Availability

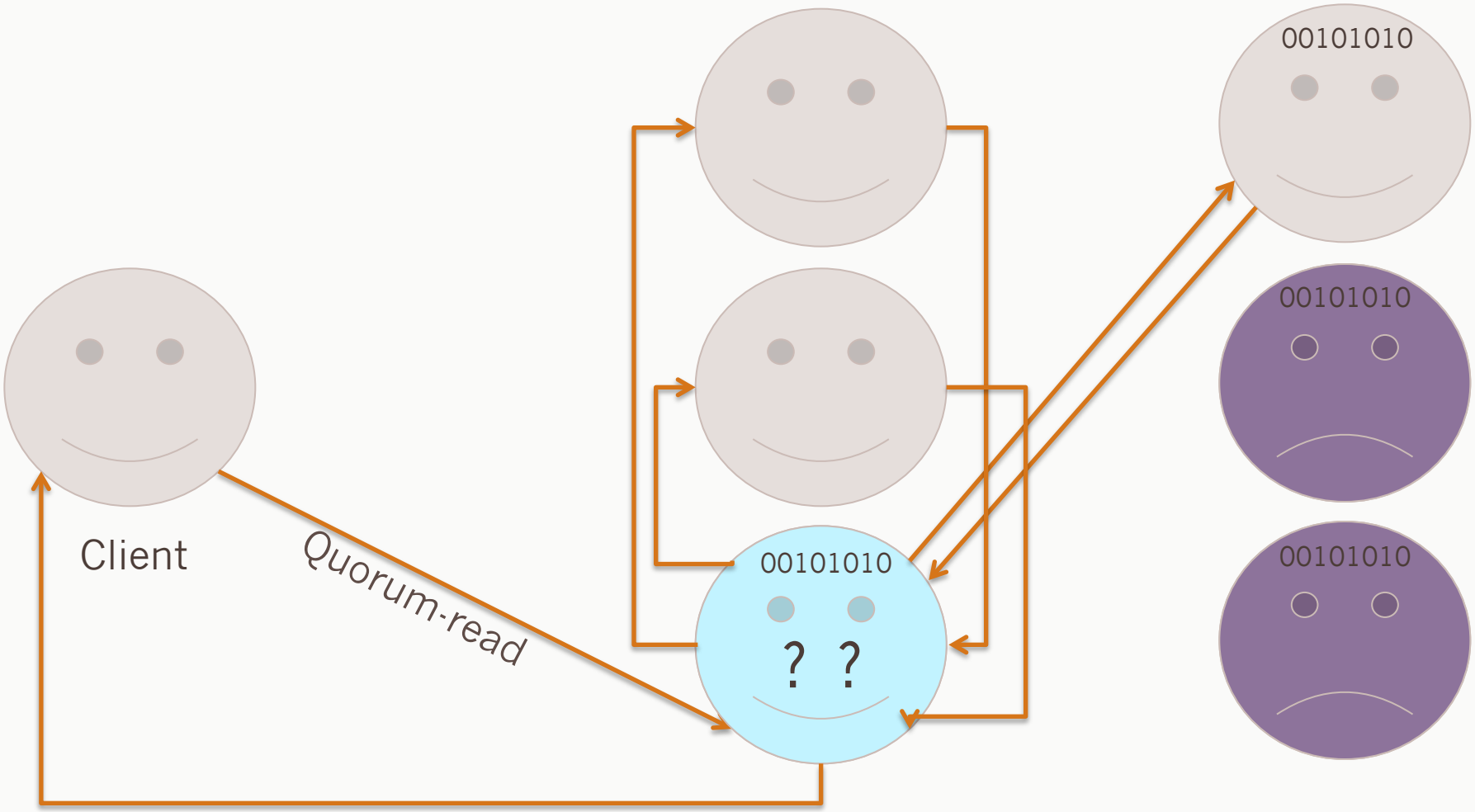


Example assumes replication factor = 3

Quorum-write, RF=5



Quorum read RF=5



Read value: 00101010

6-node cluster, rf=5

Linearizable Consistency

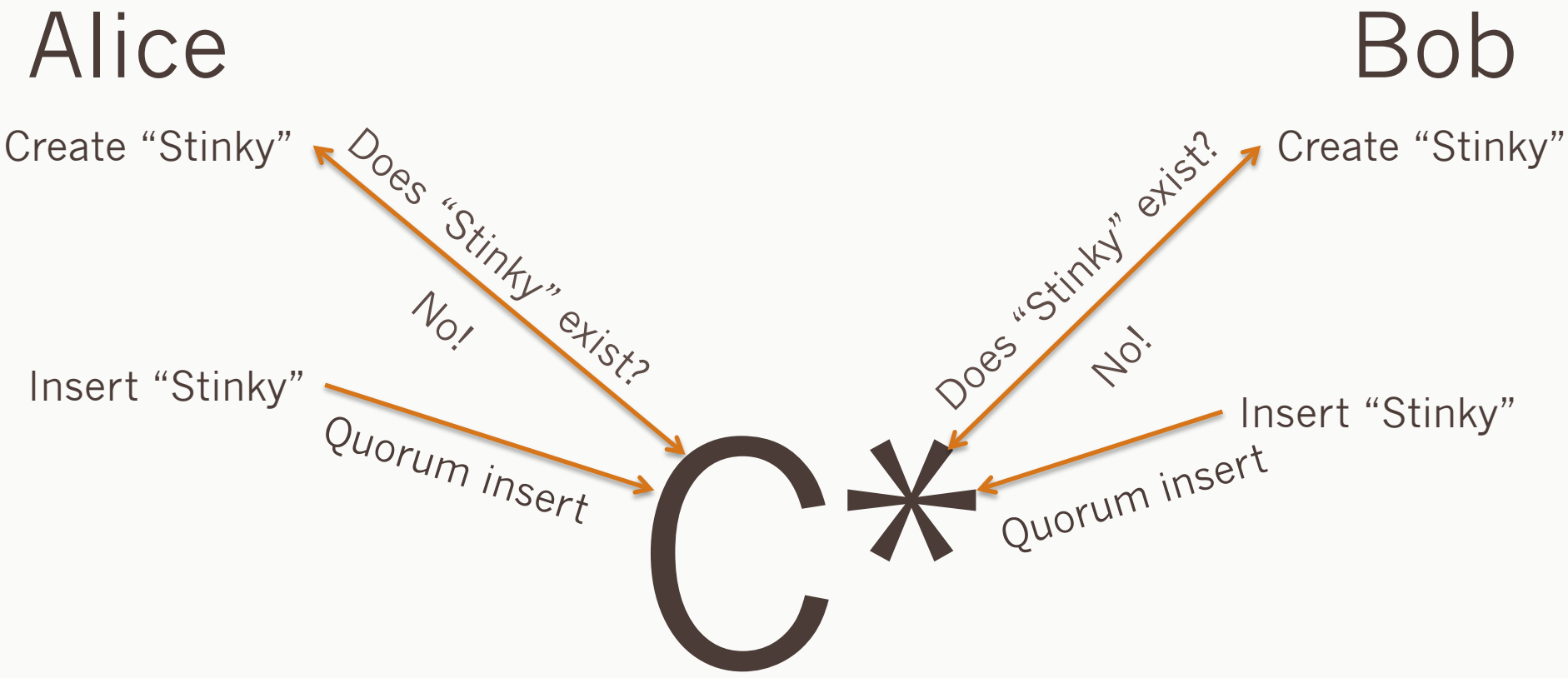
- AKA:
  - Compare & Set
  - Check & Set
  - Lightweight Transactions
  - CQL “IF” Statements

- Uses Paxos
  - Can specify local datacenter or all datacenters
  - (CASSANDRA-5797)

Linearizable Consistency

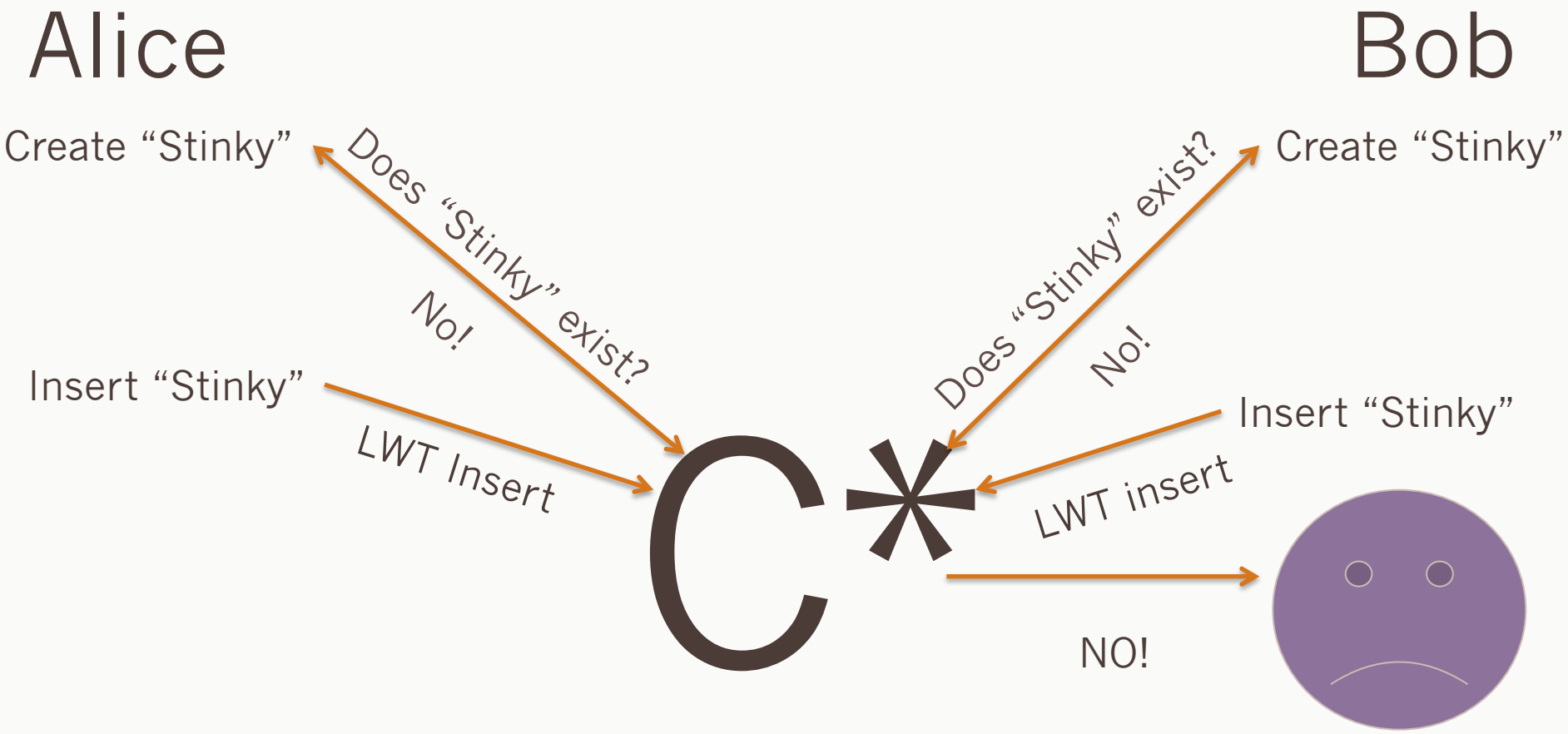
- Put X in the database if X is not already in the database
- Set foo=bar if foo=baz
- Set foo=bar if boo=faz

MONSTER MASH!!!



Now Bob owns “Stinky” and Alice doesn’t know why she can’t see the monster she just made

MONSTER MASH!!!



Consistency & Cassandra

DATASTAX

- Tunable Consistency
- Linearizable Consistency
- *Your App can tell C* what it needs*

THIS IS IMPORTANT

- The *Consistency Mechanisms* are King
- Cassandra's core strengths…
  - High Availability
  - Tolerance of **node** failures
  - Tolerance of **datacenter** failures
  - Tolerance of **network** failures between datacenters

- All due to the *Consistency Mechanisms*
- Cassandra isn't Cassandra without Eventual Consistency

Consistency is a Key Value!

DATASTAX

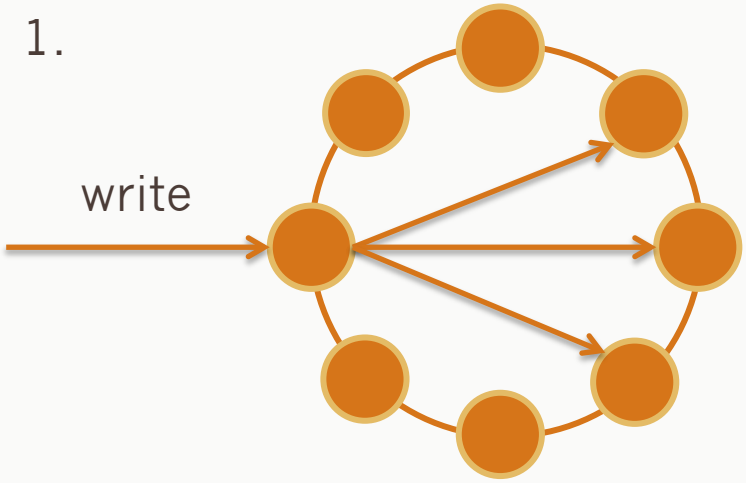
- Get it? Key-Value?
- It's a NoSQL joke.

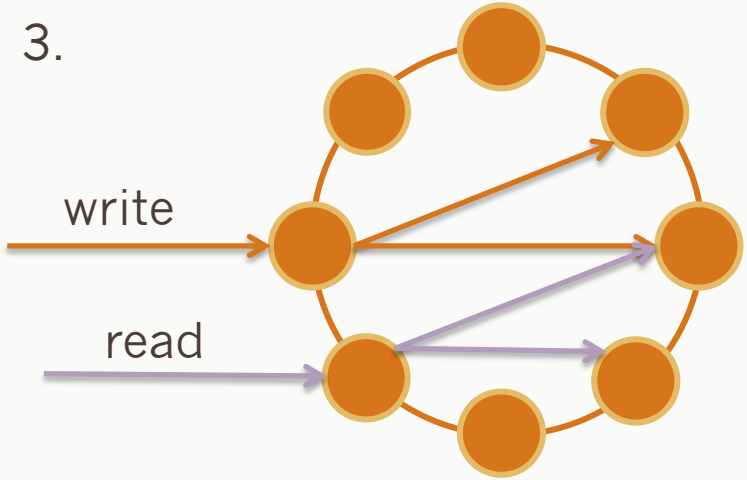
- Always-on Big Data demands multi-region DCs
- Multi-region DCs must face the reality of network partitions

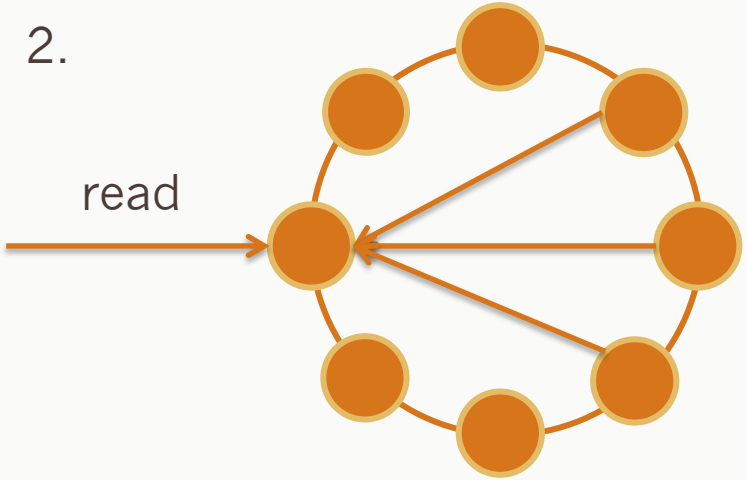
- Eventual Consistency enables applications built on Cassandra to do what they need to do correctly and flexibly while remaining constantly available.

Thank You

Tunable Consistency & High Availability





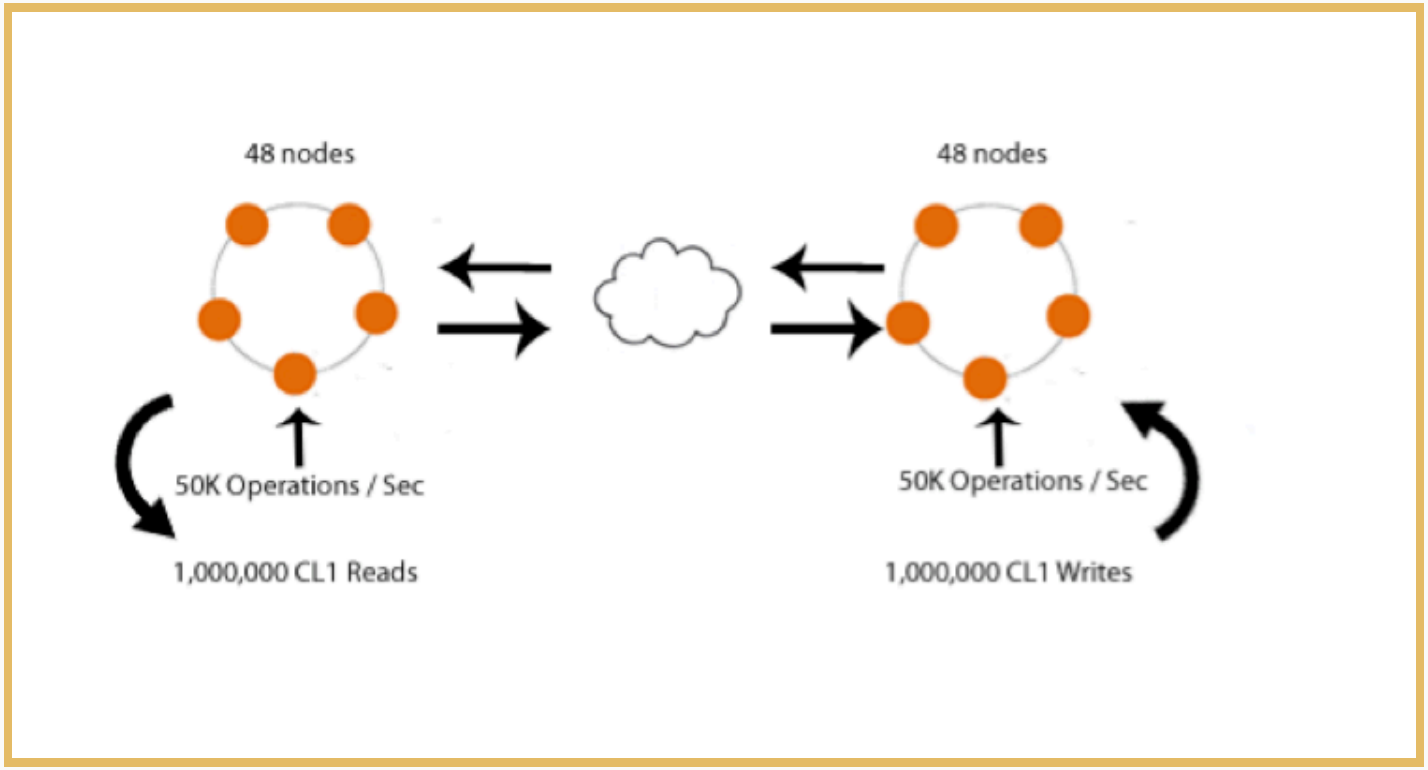


Options 1 & 2 achieve strong consistency at the expense of availability. Option 3 requires only 2 replicas on write and 2 on read, so it achieves high availability & strong consistency.

Example assumes replication factor = 3

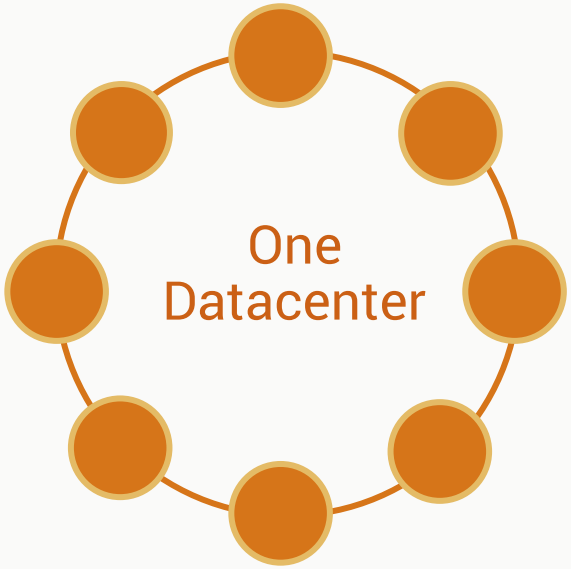
Eventual Consistency The Netflix Experiment

- "Eventual Consistency != Hopeful Consistency"
  - Netflix, big C* user, performed an experiment



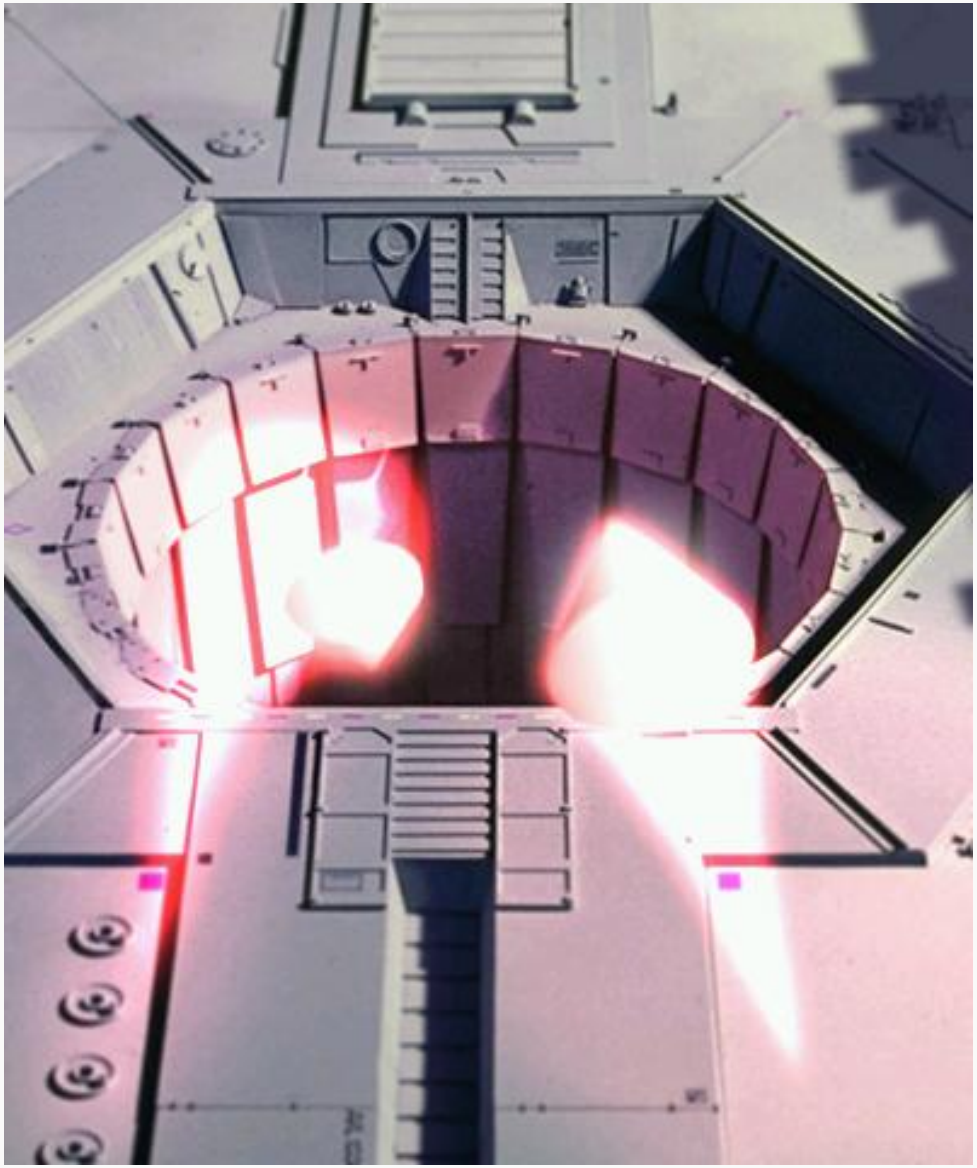
http://planetcassandra.org/blog/post/a-netflix-experiment-eventual-consistency-hopeful-consistency-by-christos-kalantzis/

C* Datacenters A Single Cluster



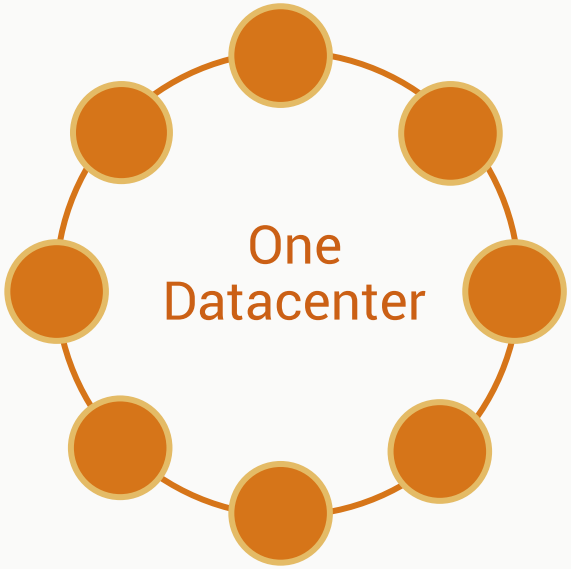
Peer-to-peer ring of nodes

Datacenters can go offline…



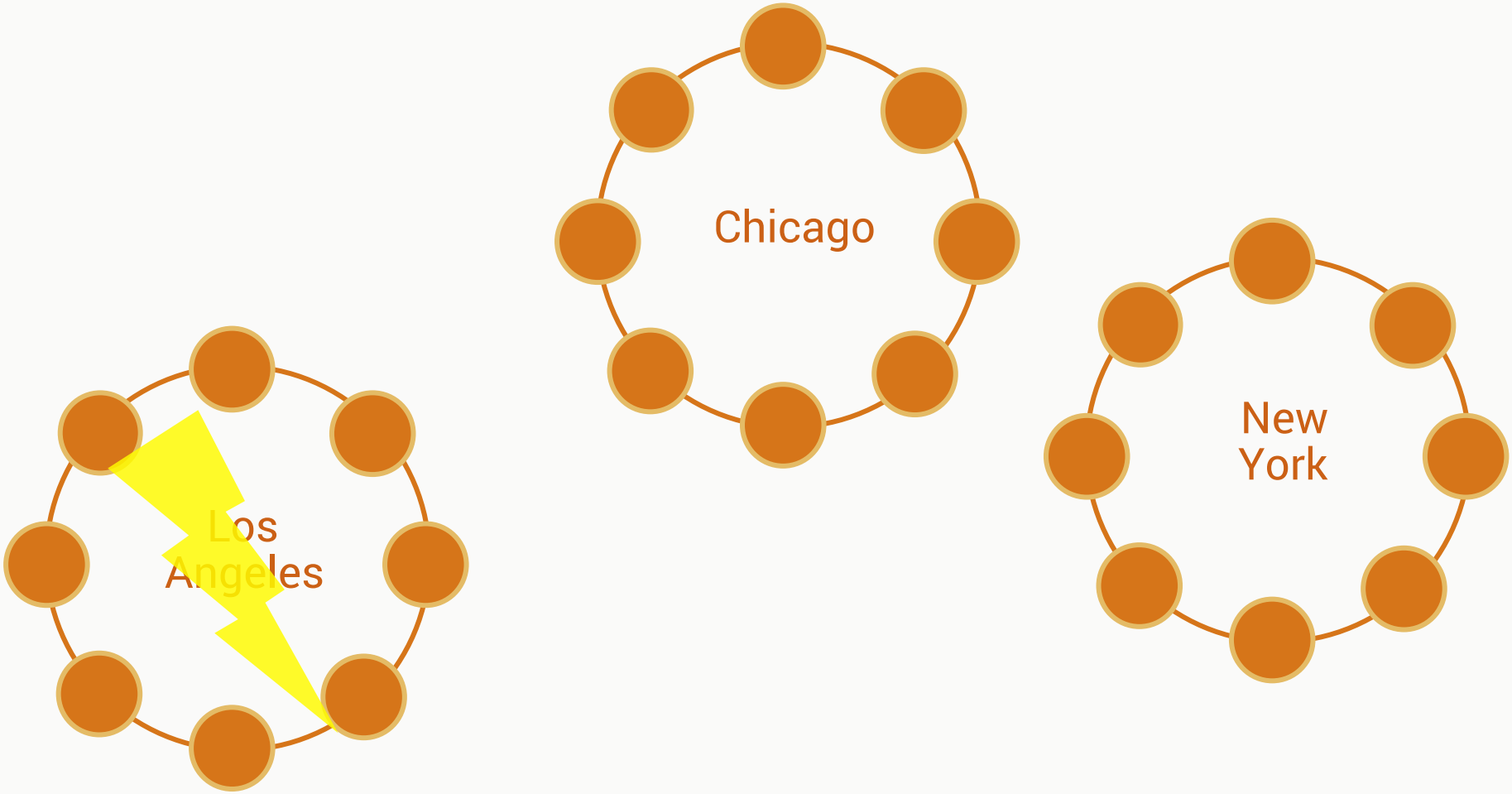
For a variety of reasons

C* Datacenters A Single Cluster

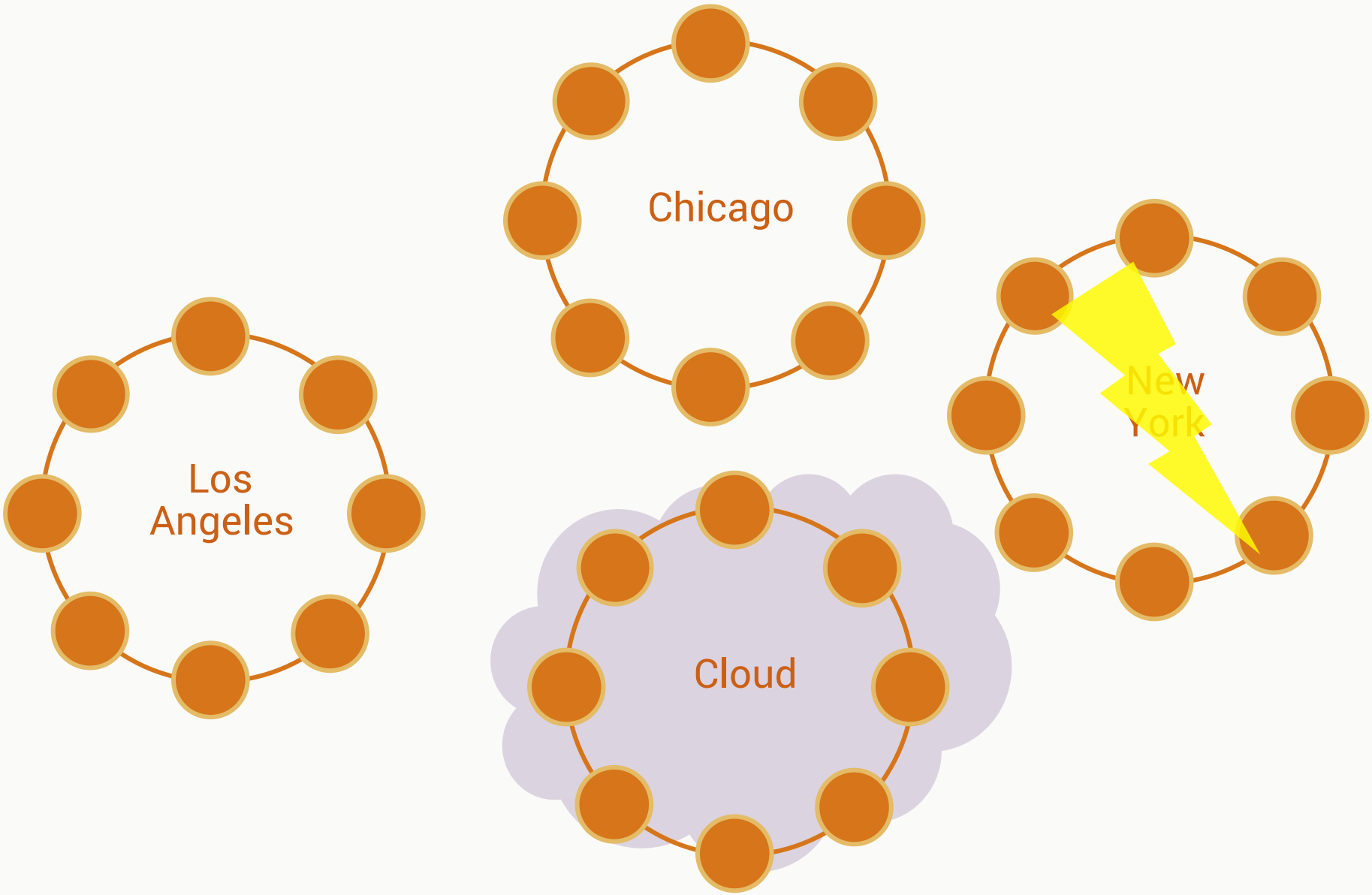


Peer-to-peer ring of nodes

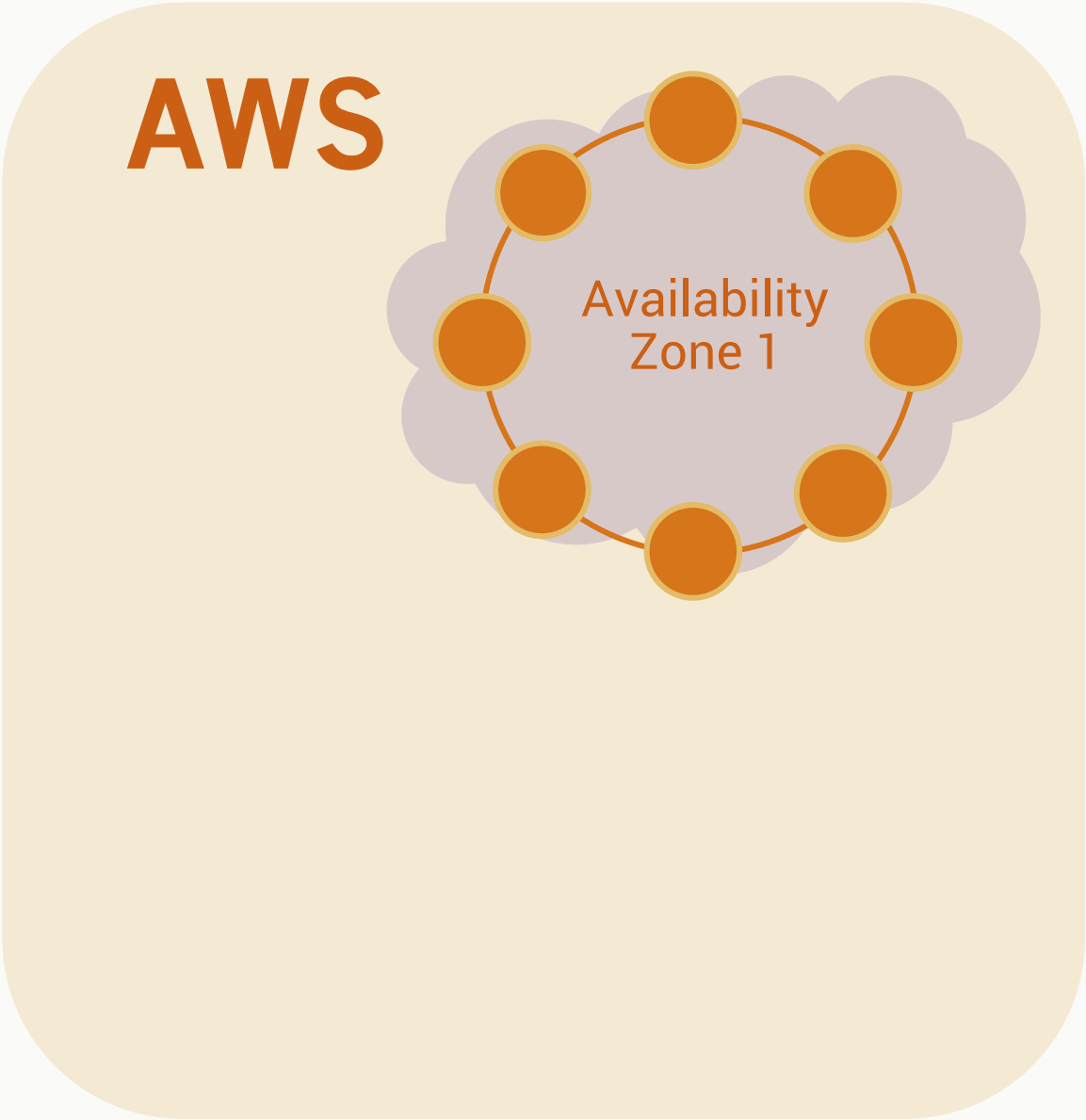
C* Datacenters What happens when I lose one?



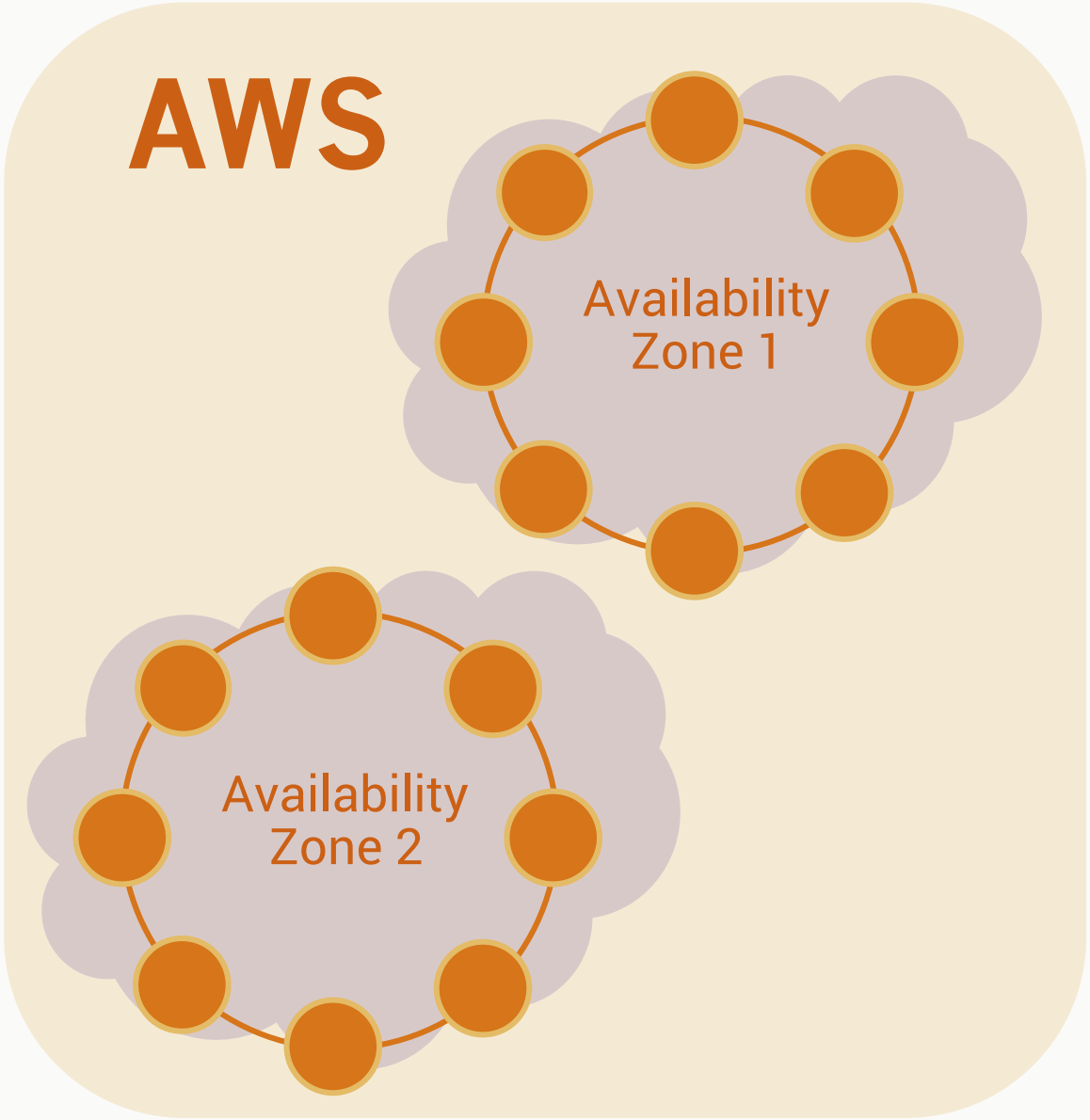
C* Datacenters What is possible?



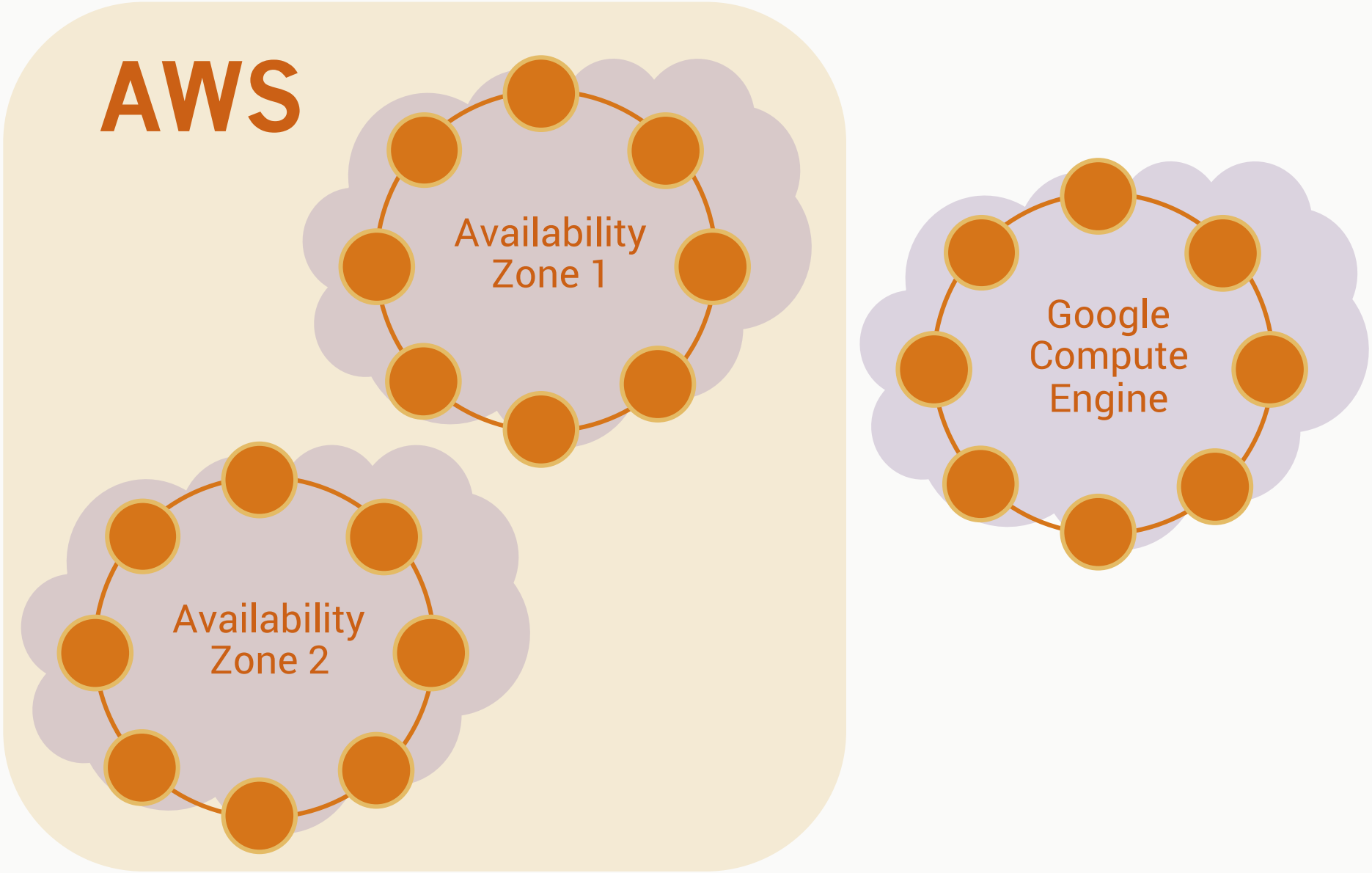
C* Datacenters Cloud Deployments



C* Datacenters Cloud Deployments



C* Datacenters Cloud Deployments



C* Datacenters Cloud Deployments