Midas

On-the-fly-Schema-Migration Tool

For

EQUAL EXPERTS

Schema Migration Problems

- Applications have to hand-roll their own schema migration infrastructure or use some third-party tool

- Difficult to migrate TBs of data without downtime
  - unacceptable from SLA stand-point!

- How about: on-the-fly schema migration - *a Midas Touch*?

Zero-downtime Deployment

- Expansion Scripts
  - Apply changes to the documents safely that do not break backwards compatibility with existing version of the application.
    - e.g Adding, copying, merging, splitting fields in a document.
- Contraction Scripts
  - Clean up any database schema that is not needed after the upgrade.
  - e.g. removing fields from a document.

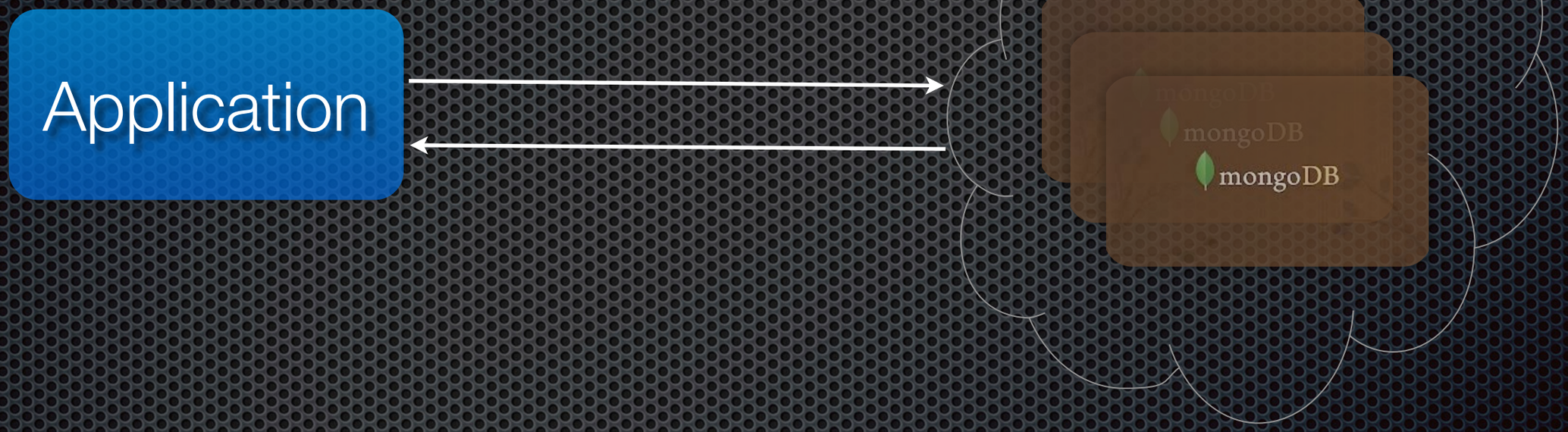
The Mechanics

- Run Expansion scripts before upgrading the application
- Upgrade the cluster, a node at a time
- Run Contraction Scripts
  - Once the system has been completely upgraded and deemed stable.
  - Typically, contractions can be run, say days/weeks after complete validation.

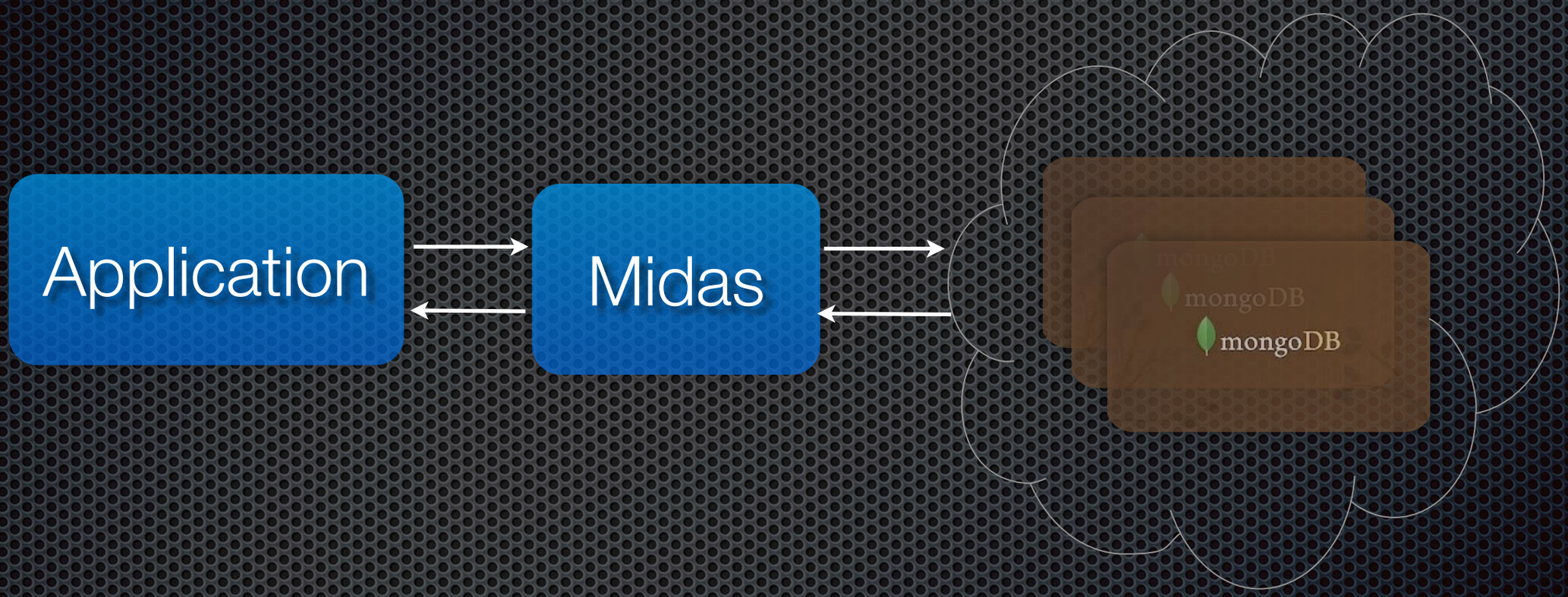
Do we need DB rollback?

- Short Answer
  - No
- Long Answer
  - Reversing DB changes can lead to potential loss of data or leave it in an inconsistent state.
  - Its safer to rollback application without needing to rollback DB changes as expansions are backward compatible.

How it works?
An Architectural Overview



How it works?
An Architectural Overview

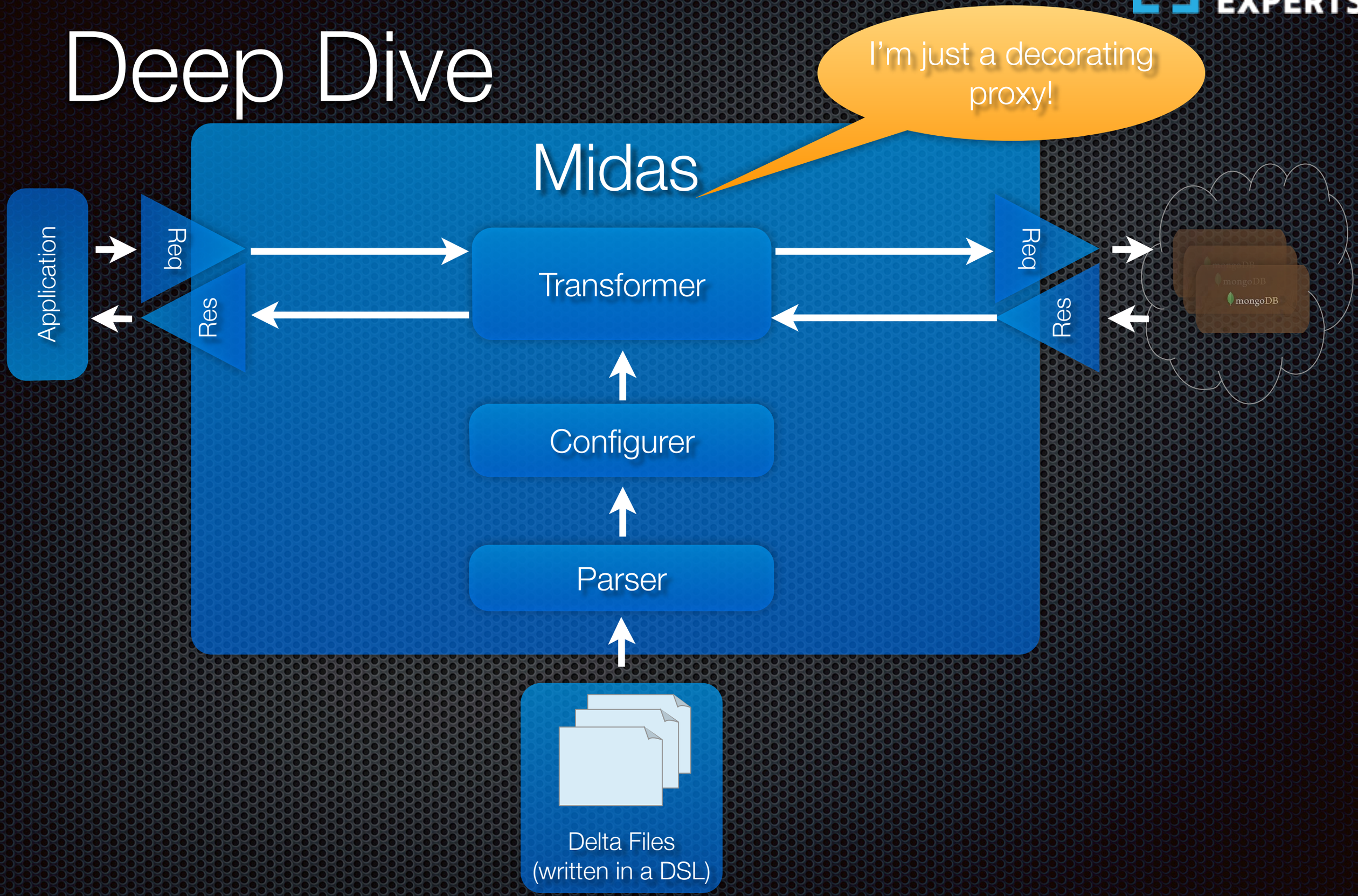


- Intercepts Responses at Protocol Level
- Upgrades/Downgrades Document schema in-transit

Protocol Level brings Transparency

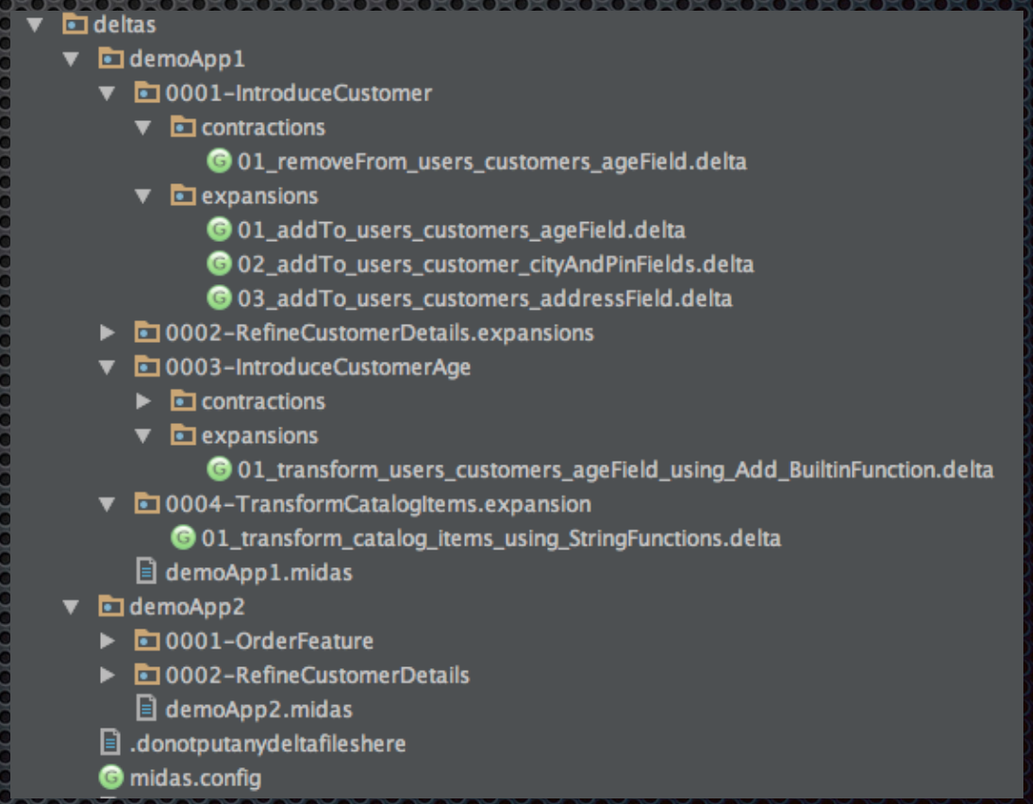
- From the App perspective
  - Midas is Agnostic of Language specific drivers and versions within those languages
    - Works with versions of Ruby, Python, C# and Java drivers.
- From MongoDB perspective
  - Midas is Agnostic of the MongoDB configurations
  - Works with Standalone, Replica Sets, Sharded environments.

Deep Dive


I'm just a decorating proxy!
Midas
Application
Req
Res
Transformer
Req
Res
mongoDB
mongoDB
Configurer
Parser
Delta Files
(written in a DSL)

Delta scripts

- Group ChangeSet as Directories
- Group deltas as Expansion and/or Contraction deltas as Directories within ChangeSet
- Write delta scripts using the `.delta` extension
- Midas relies on the order specified by you
  - Ordering info is embedded by you within the change set directory name and delta files



Delta Scripts - Convention

- Must begin change set directories with a number and subsequent change set directories are numbered in ascending order.
  - Example: use `<changesetNumber>-<featureName>` as convention for naming change set directory.
- Within a change set, expansion and contraction scripts are grouped by folders - **`expansions`** and **`contractions`**.
- Always begin delta scripts with a number and subsequent delta scripts are numbered in ascending order.
  - Example: use `<changeNumber>_<WhatTheChangeIs>.delta` as convention for naming delta files

Sample Delta Script

- Each Delta script is written using a DSL
- Very close to MongoDB lingo, virtually no learning curve.

```
use users
db.customers.remove('["address.line1"]')
db.customers.merge([“lname”, “fname”], “ “, “name”)


use transactions
db.orders.add('dispatch : { status: 'NOT DELIVERED'}')
```

EQUAL EXPERTS

Agile App Delivery & DevOps

- Inject Midas into Architecture
  - Start or Middle of project
- Supports Development of Application in small-steps
  - Add Application ChangeSets/Deltas on-the-fly
- Copes with Load
  - Add/Remove Application Nodes on-the-fly
- Supports Multiple Applications
  - Add/Remove Application(s) on-the-fly

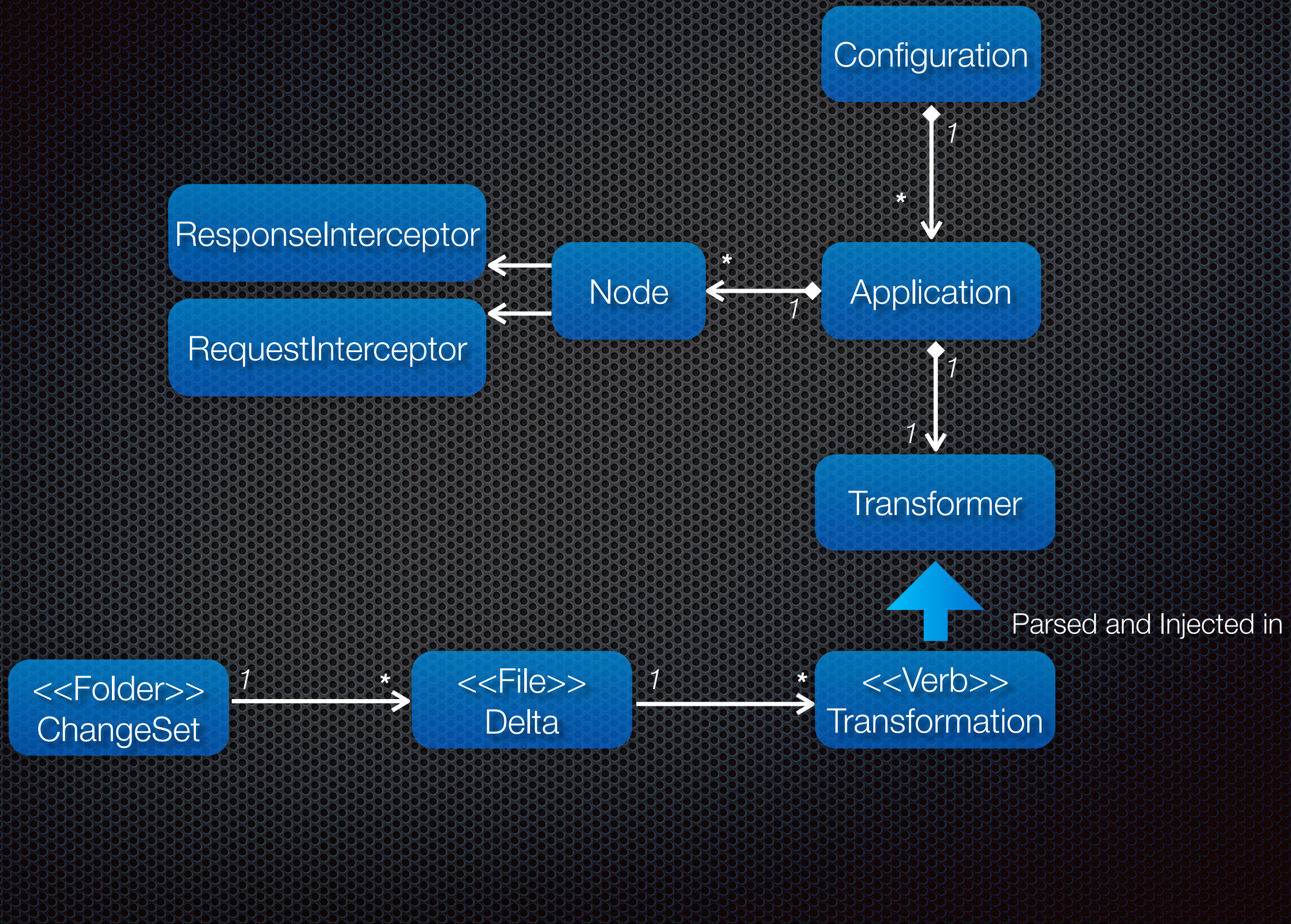
No Leaky Abstractions

- Does not expect the Domain Model to be aware of versioning.
  - Allows developers to focus on the domain while freeing them from versioning concerns.
    - If you wish to take charge, Midas will not come in your way.
- Midas maintains versioning information within the document itself.
  - _expansionVersion and _contractionVersion fields are part of the documents “touched” by Midas.
  - Updates them during request and response.

Caveats

- Never ever change a delta that has been applied to production.
  - Always move forward in time.
  - Reverse a change by a counter-change.
- Force-update migration on documents that are not
  - expanded by App demand, and then proceed to contraction.
  - contracted by App demand, and then proceed to next App-upgrade cycle.
- Currently Midas supports Zero Downtime Configuration - 1 (see later slides), others will be supported in subsequent releases.

Midas Domain Model



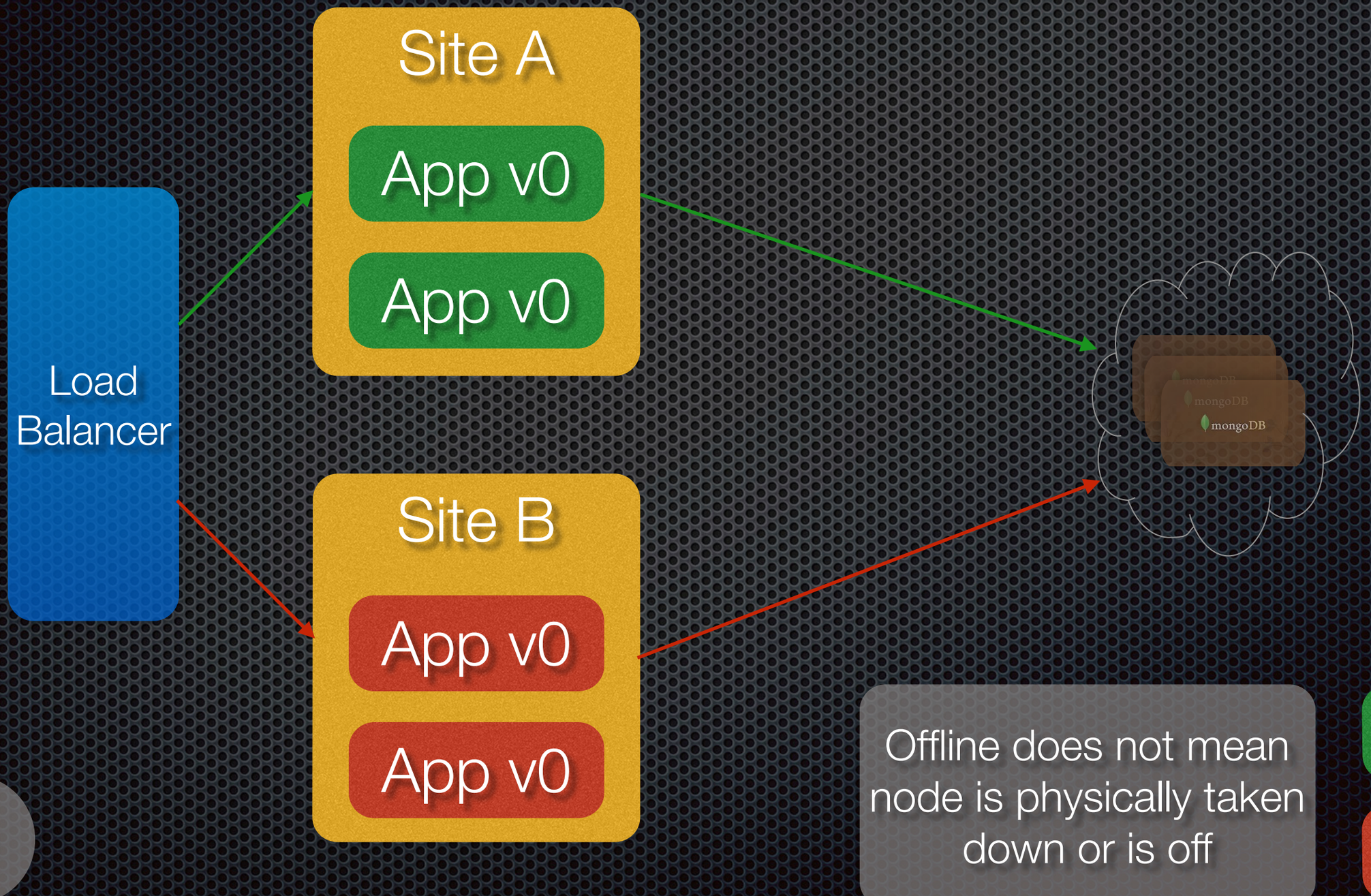
EQUAL EXPERTS

Zero-Downtime Deployment

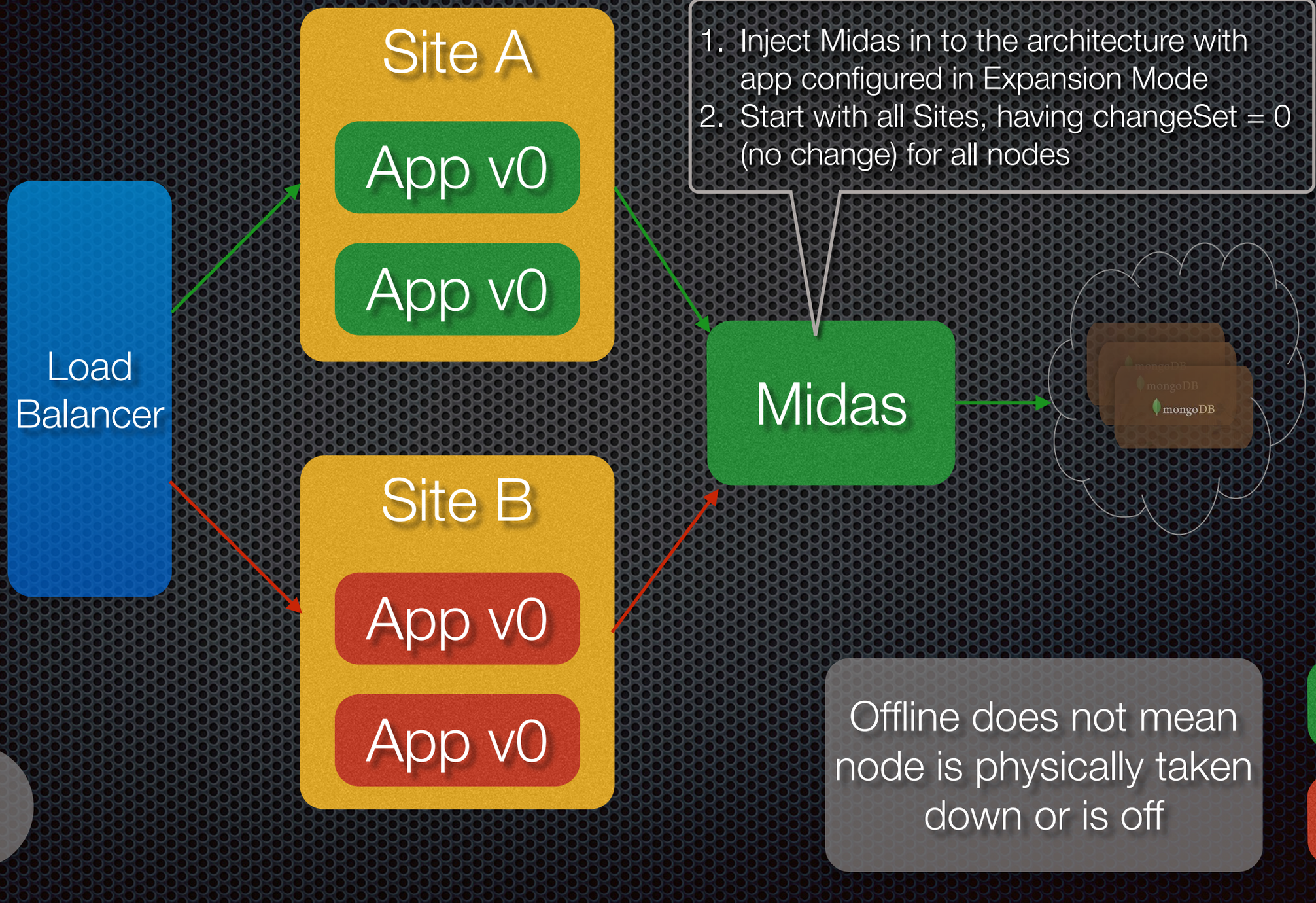
Configuration - 1

Injecting and Using Midas

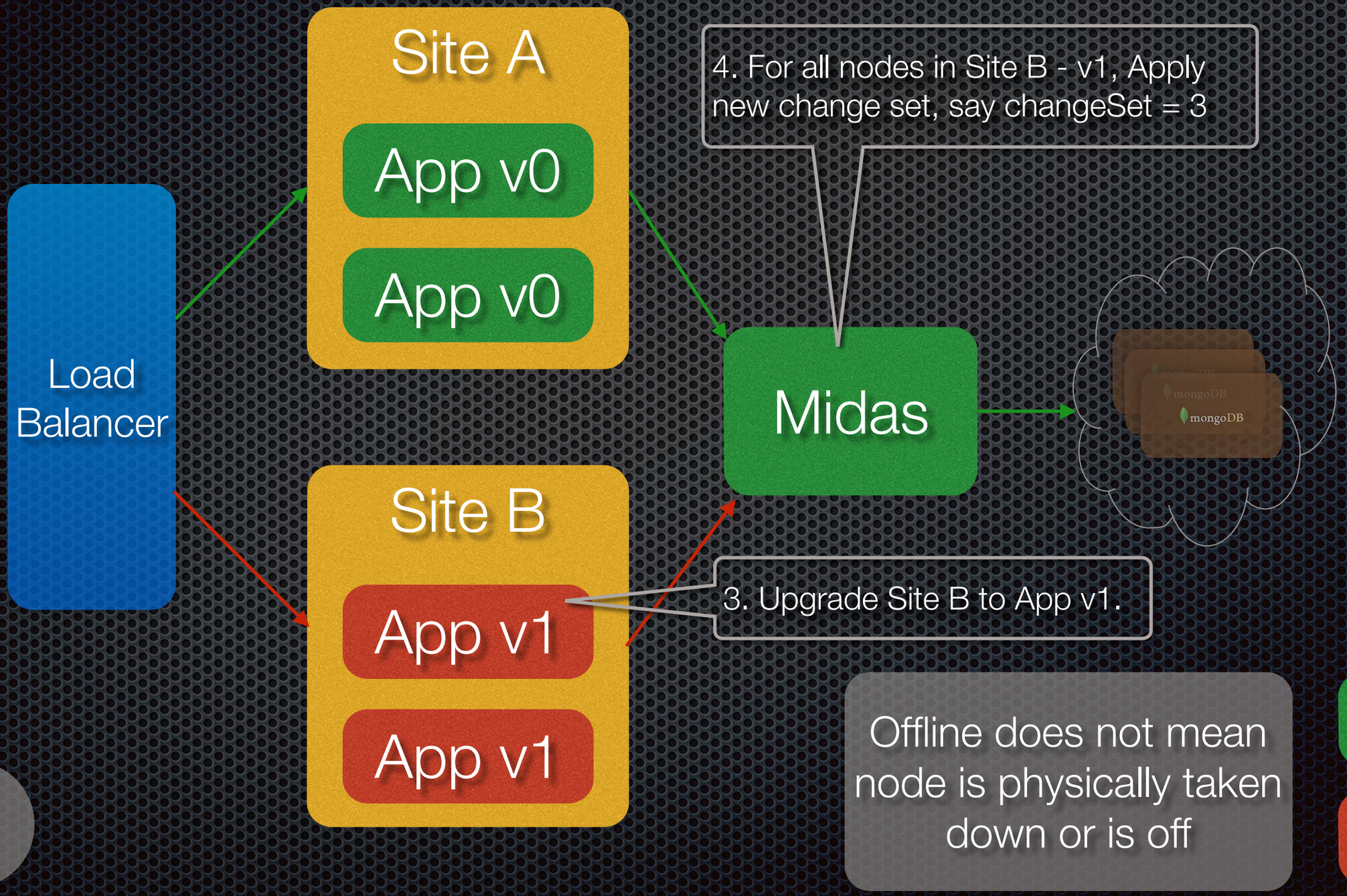
Deployment config - 1



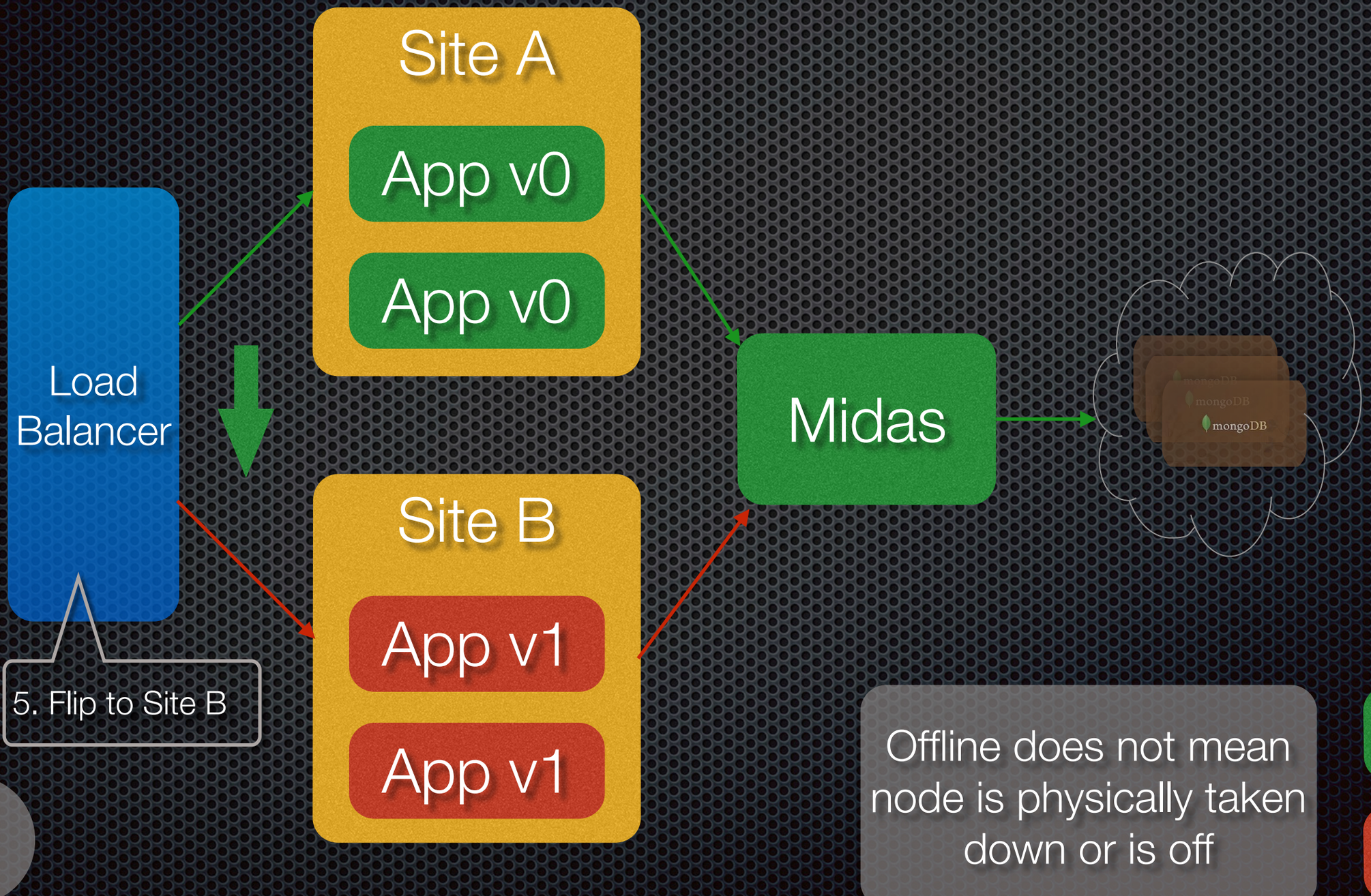
Deployment config - 1



Deployment config - 1



Deployment config - 1

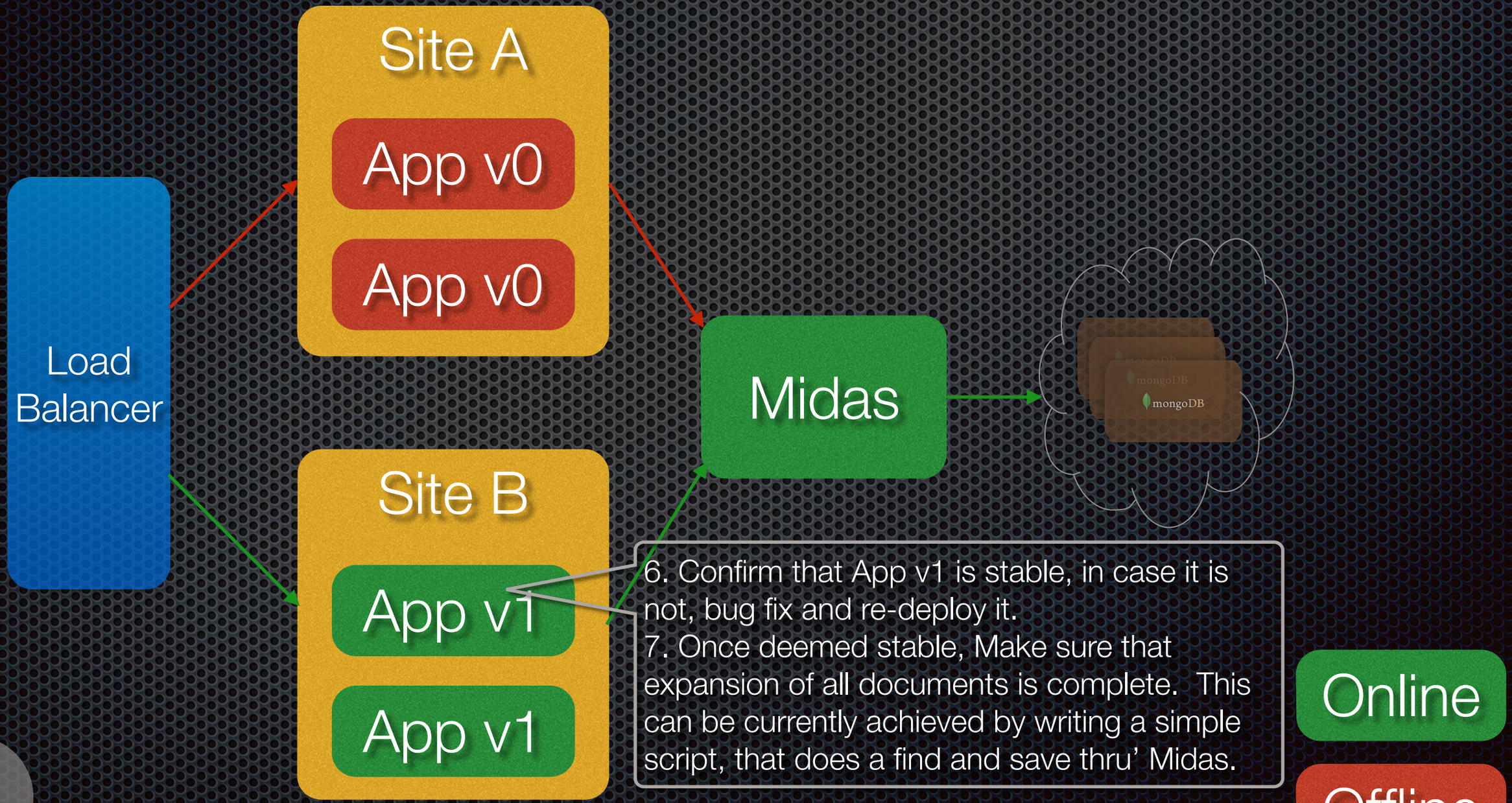


Offline does not mean node is physically taken down or is off

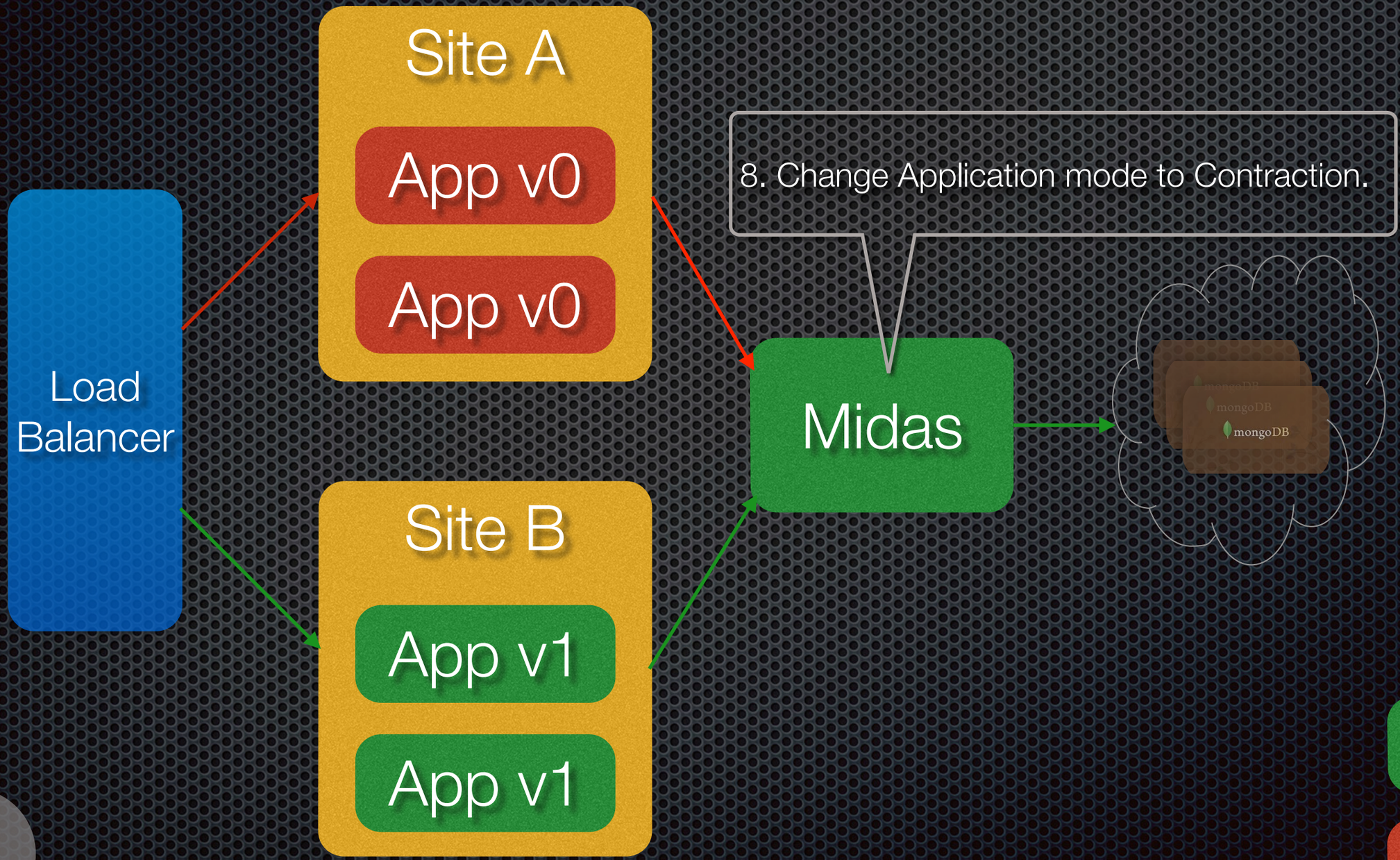
Online

Offline

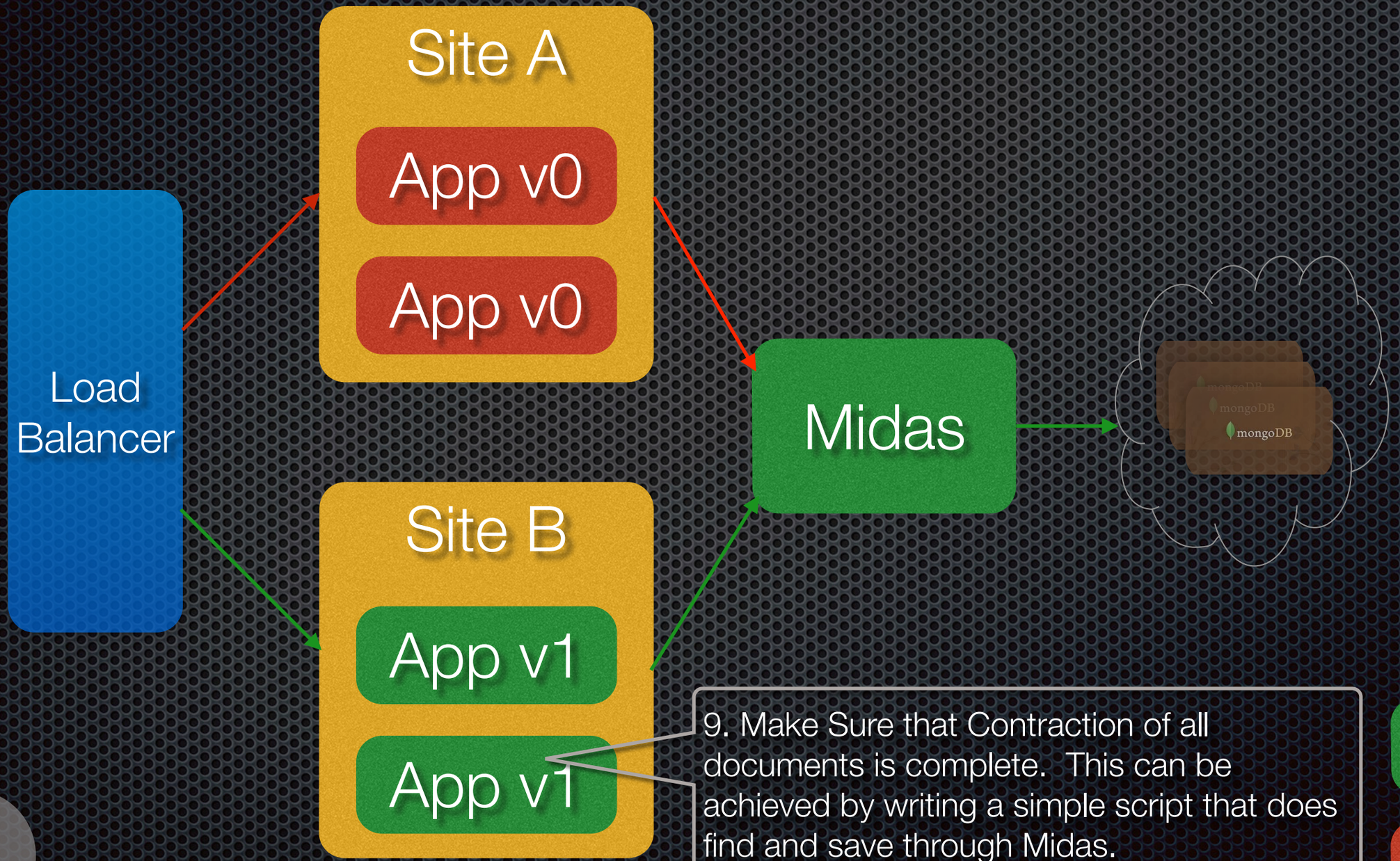
Deployment config - 1



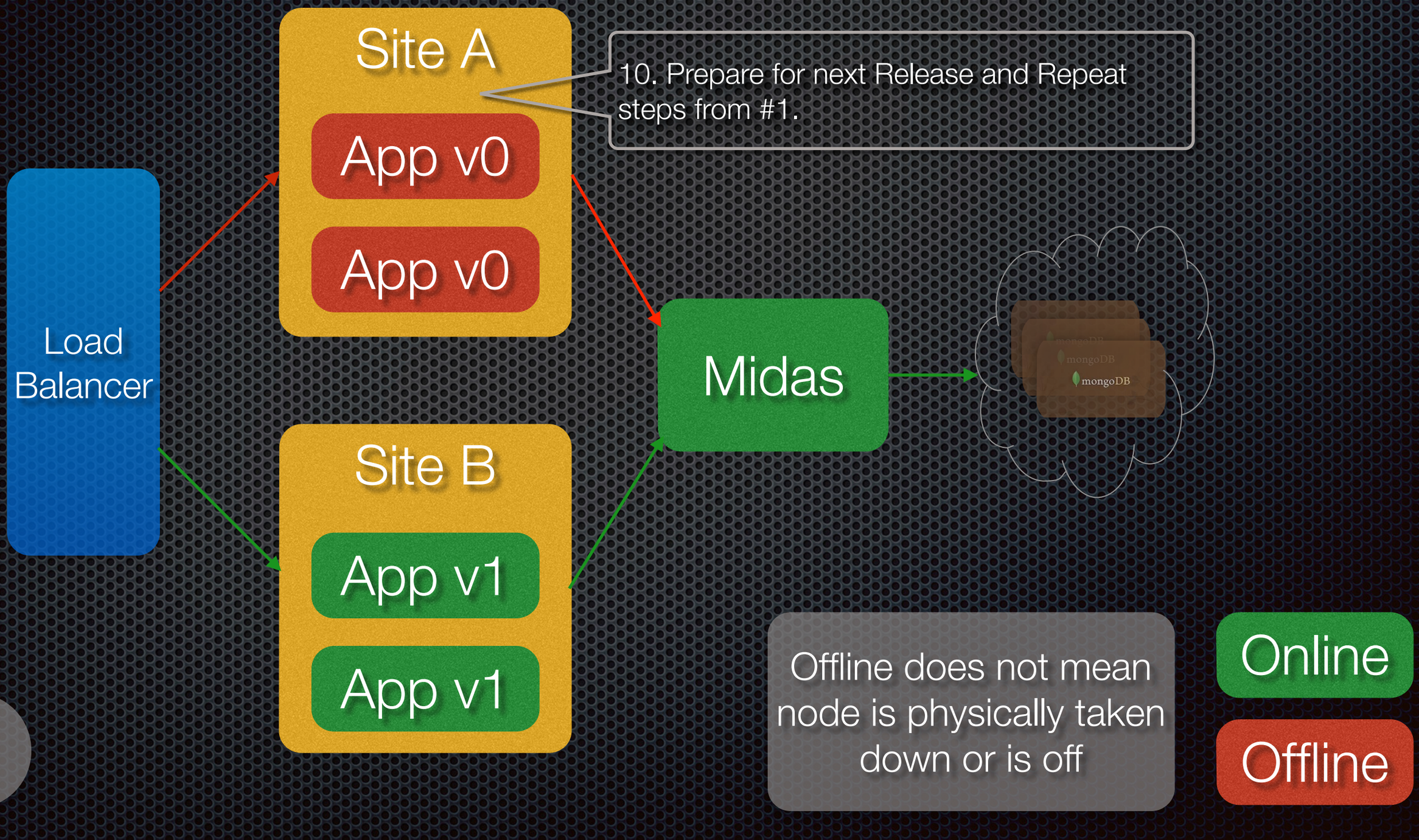
Deployment config - 1



Deployment config - 1



Deployment config - 1



EQUAL EXPERTS

Zero-Downtime Deployment

Configuration - 1

Node Removal at Runtime

EQUAL EXPERTS

Say, due to some problems, you want to remove Node0 from service:

1. Remove Node0 from LB.

2. Remove Node0 from **app.midas**

```
demoAppV1 {
  mode = contraction
//  siteANode0 {
//    ip = x.x.x.x
//    changeSet = 3
//  }
  siteANode1 {
    ip = y.y.y.y
    changeSet = 3
  }
}
```

Load Balancer

Site A

Node0

Node1

Midas

mongoDB

mongoDB

Online

Offline

2. Midas will sever all connections with Node0

Load Balancer

Site A

Node0

Node1

Midas

Offline does not mean node is physically taken down or is off

Online

Offline

EQUAL EXPERTS

Zero-Downtime Deployment

Configuration - 1

Node Injection at Runtime

You want to inject new node for App in service

1. Add Node2 to **app.midas**.

```
demoAppV1 {
  mode = expansion
  siteANode0 {
    ip = x.x.x.x
    changeSet = 3
  }
  siteANode1 {
    ip = y.y.y.y
    changeSet = 3
  }
  siteANode2 {
    ip = z.z.z.z
    changeSet = 3
  }
}
```

Load Balancer

Site A

Node0

Node1

Node2

Midas

mongoDB

mongoDB

2. Midas is ready to receive connections from Node2

Offline does not mean node is physically taken down or is off

Online

Offline

EQUAL EXPERTS

3. Add Node2 to LB.

Load Balancer

Site A

Node0

Node1

Node2

Midas

mongoDB

mongoDB

Online

Offline

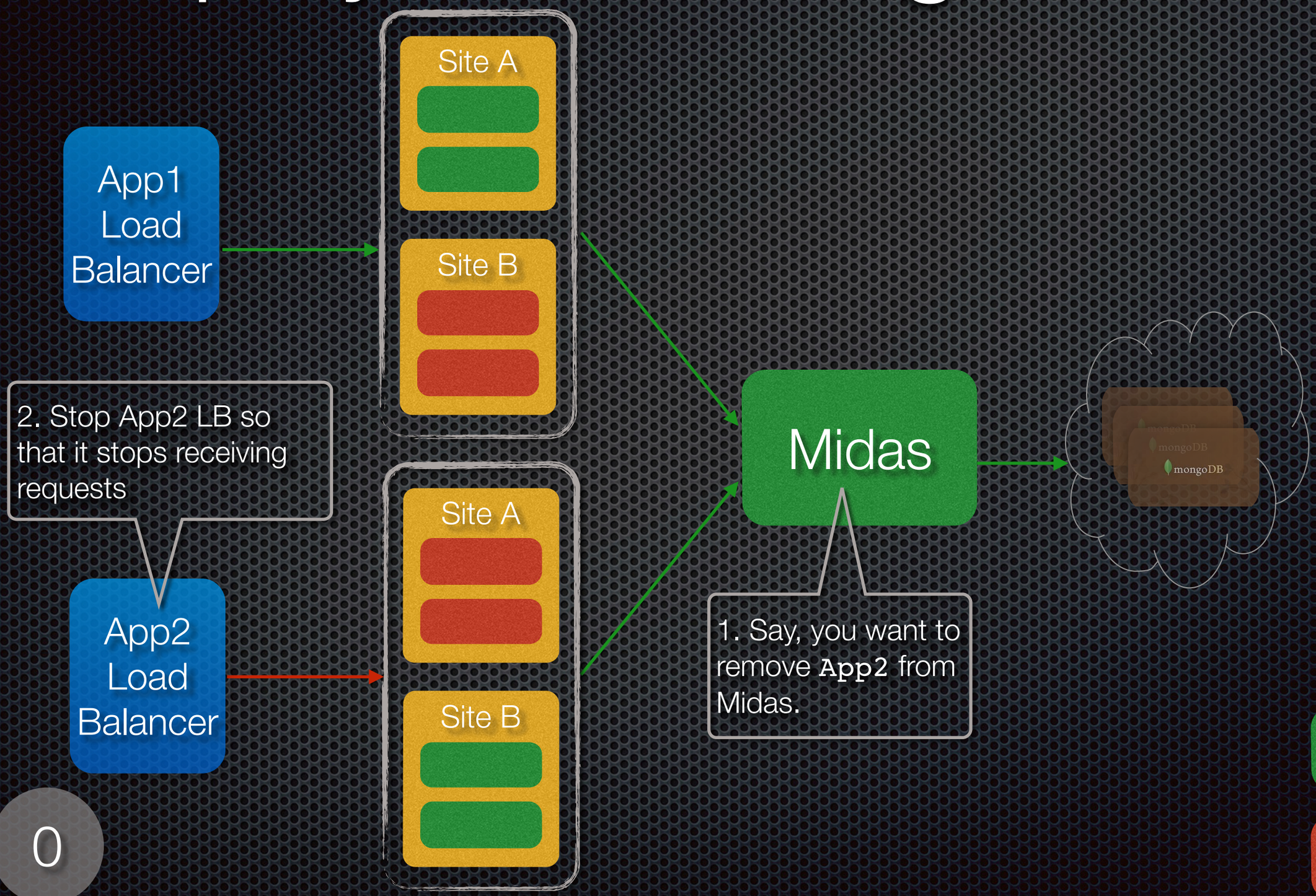
EQUAL EXPERTS

Zero-Downtime Deployment Configuration - 1

App Removal at Runtime

EQUAL EXPERTS

Deployment config - 1



EQUAL EXPERTS

1. Remove App2 from **`midas.config`**

```
apps {
  demoApp1
//  demoApp2
}
```

Note: *You don't have to remove* ***`demoApp2`*** *dir or* ***`demoApp2.midas`*** *to remove the application from Midas (this allows you to re-inject App2 should the need be)*

App1 Load Balancer

Site A

Site B

Midas

mongoDB

mongoDB

2. Midas will sever all connections from **`App2`**

App2 Load Balancer

Site A

Site B

3. You can now take down **`App2`**

Online

Offline

EQUAL EXPERTS

Zero-Downtime Deployment Configuration - 1

App Injection at Runtime

EQUAL EXPERTS

App1 Load Balancer

Site A

Site B

1. Say, you want to add `App2` to Midas.

Midas

App2 Load Balancer

Site A

Site B

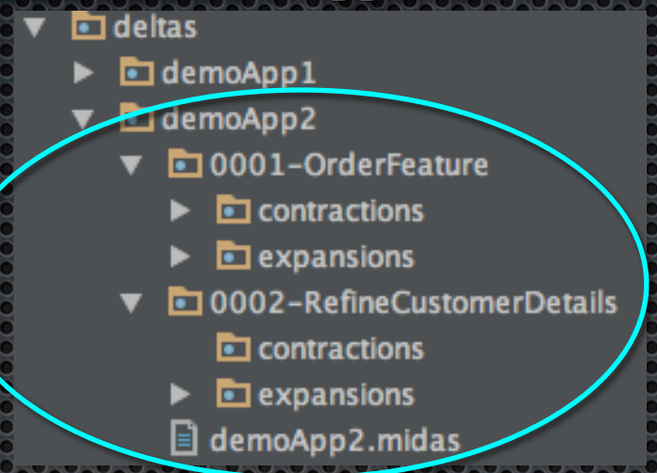
2. Setup the infrastructure for `App2` (Sites and Nodes).

Online

Offline

EQUAL EXPERTS

3. Create `App2` dir within deltas dir
4. Set-up change sets with expansions and contractions in `App2` dir.



App1 Load Balancer

Site A

Site B

Midas

mongoDB

mongoDB

App2 Load Balancer

Site A

Site B

Online

Offline

5. Add **`demoApp2.midas`** (with mode and nodes configured)

```
demoApp2Ver1_0 {
  mode = expansion
  siteANode1 {
    ip = a.a.a.a
    changeSet = 2
  }
  …
  …
  siteBNode1 {
    ip = p.p.p.p
    changeSet = 2
  }
}
```

App1 Load Balancer

Site A

Site B

App2 Load Balancer

Site A

Site B

Midas

mongoDB

mongoDB

Online

Offline

EQUAL EXPERTS

App1 Load Balancer

Site A

Site B

6. Add App2 to **midas.config**

```
apps {
  demoApp1
  demoApp2
}
```

7. Midas is ready to receive connections from App2

Midas

9. Start the LB so that App2 is live

App2 Load Balancer

Site A

Site B

8. Start any Site that can go-live, say Site A here

Online

Offline

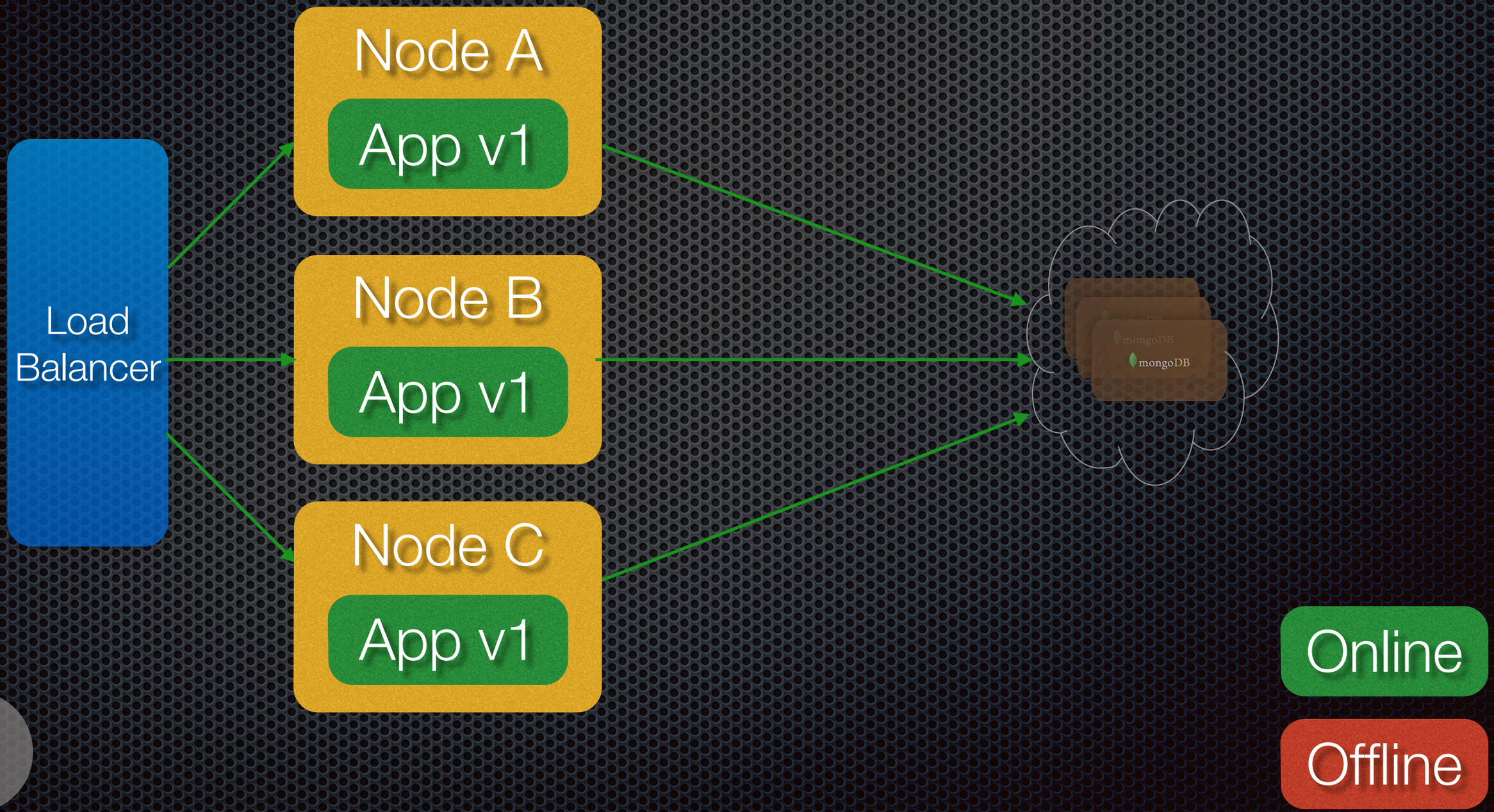
EQUAL EXPERTS

Zero-Downtime Deployment

Configuration - 2

Not Supported in this Release

Deployment config - 2

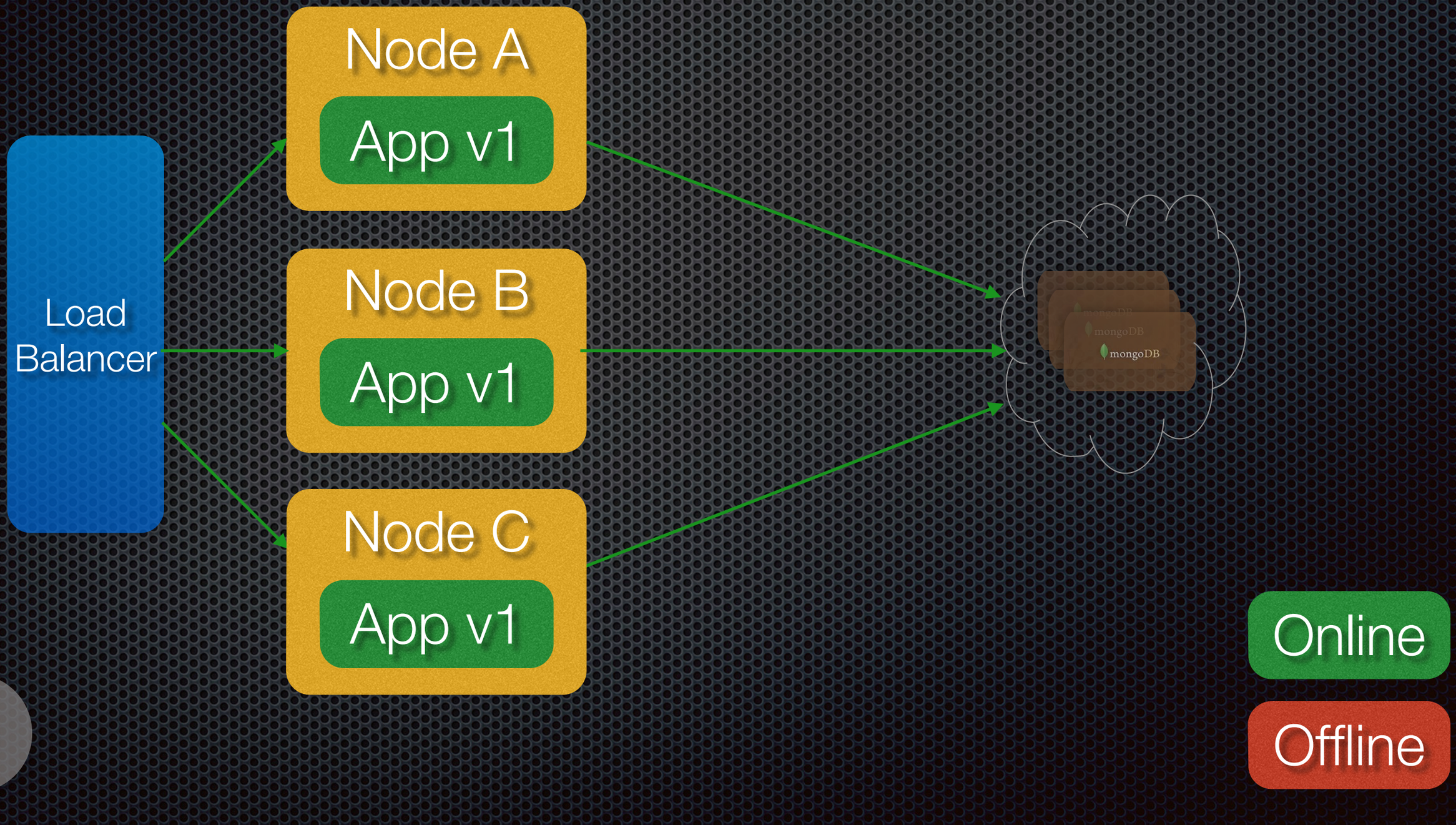


EQUAL EXPERTS

Zero-Downtime Deployment Configuration - 3 AWS

Not Supported in this Release

Deployment config - 3



The Team

- Brian Blignaut
  - bblignaut@equalexperts.com
- Dhaval Dalal [@softwareartisan]
  - ddalal@equalexperts.com
- Vivek Dhapola
  - vdhapola@equalexperts.com
- Komal Jain
  - kjain@equalexperts.com

References

- Owen Rogers
- http://exortech.com/blog/2009/02/01/weekly-release-blog-11-zero-downtime-database-deployment/

EQUAL EXPERTS

Thank-You