KIDLY Software Engineer take-home test, part 2.

KIDLY's customer facing website is an excellent example of a software product that has a very long life and which changes frequently, and significantly over time. Its only purpose is to serve KIDLY's customers, and it needs to be available 24/7/365. We aim for zero downtime. For the sake of this task, you need to know that the KIDLY website environment consists of a web application, a document database, and few background services. The web applications and services communicate with other applications in the KIDLY retail ecosystem using queues and pub-sub topics.

1. Describe what principles or practices you would recommend in order to ensure that all deployments can happen during daytime.

2. Imagine the following: Delivery details for orders are captured on an `Order` class as a series of independent properties:

```
{
    public decimal DeliveryCharge{get;set;}
    public decimal DeliveryDiscount{get;set;}
    public DeliveryMethod ChosenDeliveryType {get;set;}
    public Address DeliveryAddress{get;set;}
}
```

The database that holds order information stores the data as JSON documents which mirror the structure of the `Order` class. If a property is removed from the order JSON document, then the corresponding property on the `Order` class cannot be populated when an order is loaded from the database.

Your task is to encapsulate the four properties listed above into a new class called `DeliveryDetails`, and to expose this property on the `Order` class like so:

```
{
    public DeliveryDetails Delivery {get; private set;}
}
```

The four existing properties should be removed.

Given that we can have no downtime during releases, and that all systems must continue to function normally while data and code is being modified, can you imagine how you might go about making a change like this?

Hints:

- You do not need to constrain yourself to a single release for this change.
- You can assume that the existing data access layer takes care of all the work of mapping JSON data to properties/members on classes both during load and persistence. If you have a property on a class, the DAL will persist it to JSON. Conversely, if a property exists on a JSON document the DAL will attempt to map it to a property with the same name during a load operation (this is, roughly, how RavenDb works).
- Provide your answer as written text, in whatever format you prefer. Write as much or as little as you like, but do know that we love getting into the detail of things. We don't expect you to spend hours on this task, but we *do* expect a well considered answer.