Use Case: Scalable applications

1. Introduction

A lot of companies are running (web) applications on a single machine, self-hosted, in a datacenter close-by or on-premise. The hardware is often bought or leased and the infrastructure (network, cabling) managed by themselves.

Using Cloud technologies already gives you certain benefits:

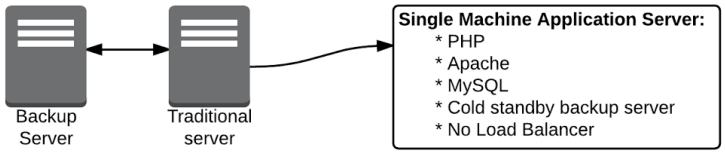
- Pay as you go model
- Reduced capital expenditures
- Managed hardware, network, and software (excluding OS and app)
- Global presence (US, South America, EU, Asia, Australia regions)

To optimally benefit from all cloud features, application design should be adapted to the cloud:

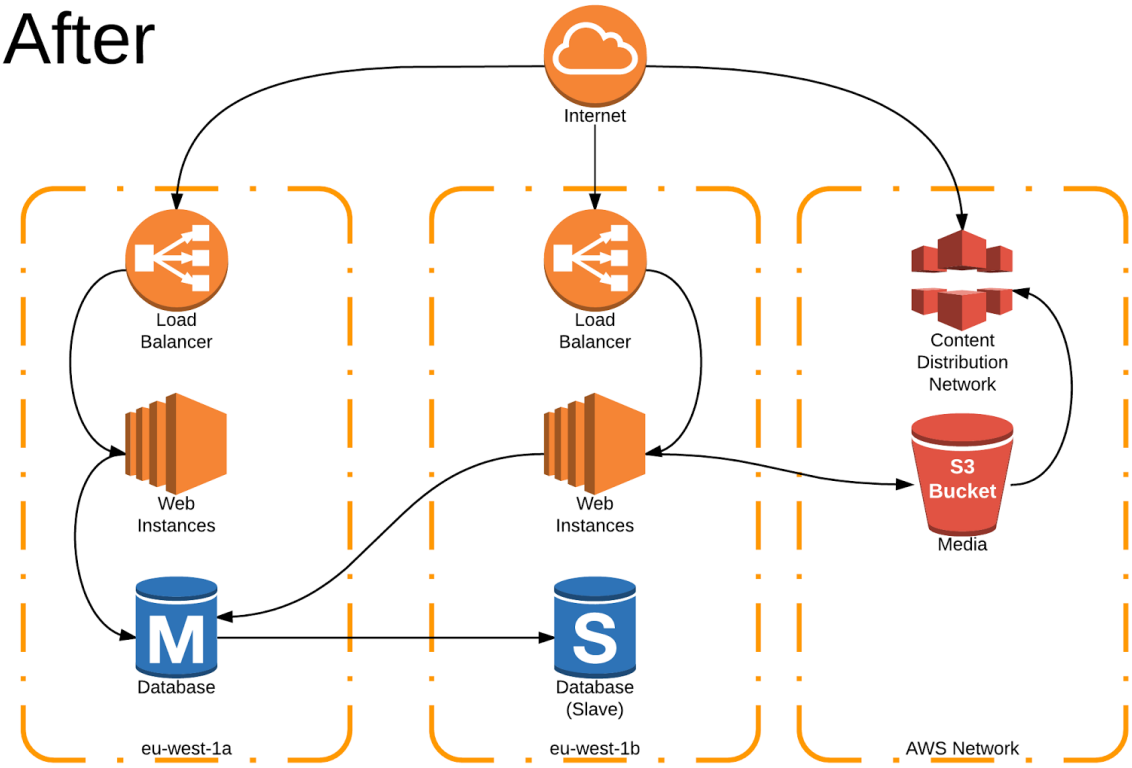
- Multi-datacenter availability
- Serving static data closest to the customer
- On-demand, inexpensive storage using Amazon S3
- Horizontal scaling capabilities using on-demand server resources
- Managed Relational database services using Amazon RDS
    - Automatic security updates
    - Managed backups / snapshots
    - Automatic failover
    - Vertical scaling with minimal downtime
- Managed Load balancer
- Green/Blue deployments and zero-downtime deployments
- Integrated Continuous Integration and delivery workflow using AWS CodePipeline
    - Test application code on every code change
    - Automatic deployments to dev, test and production

2. Architecture Diagram

Before



After



3. Setup

Infrastructure setup

The infrastructure is set up by in4it. After gathering the customers requirements, the infrastructure is provisioned using the customer's AWS credentials. The infrastructure blueprint (documentation) is issued and any relevant credentials are handed over to the customer.

Typically dev and optionally QA/Test are setup first, so the customer can prepare the application migration. Typical migration timelines go from a couple of weeks to months.

Application setup

Changes to the applications will be necessary to get all the benefits described in the introduction of this document. In4it works together with the customer to make sure the migration happens flawlessly. Application changes can include, but are not limited to:

- The application code needs to be stored in version control, preferably git
    - Using services like Bitbucket, GitHub or AWS CodeCommit
- The application layer needs to be stateless
    - No files can be stored on the local disk
    - No sessions can be saved in the temporary directory
    - No cache files that cannot be automatically regenerated should not be stored
- The application needs to be disposable
    - If one instance of the application shuts down, it shouldn't impact the application
- The application should use environment variables to use external services
    - External services can be a Database, Caching or Search services
    - Credentials are saved into the environment, and are not saved in version control
- Any static data need to be stored on shared storage (AWS S3 / EFS)
- Static data can be served the CDN (Content Distribution Network - AWS CloudFront)
- Any existing periodical scheduled processes (like crontab) should be moved to a worker instance
    - Ideally a queuing system should be used, if not, the worker processes cannot be scaled out horizontally, only vertically (1 instance with more cpu / memory)
    - AWS SQS can be used, or any other distributed queuing system prefered by the customer
- Outgoing e-mail must use an SMTP service (for example AWS SES - Simple Email Service)
    - Bounce/Complaints handling needs to be implemented
    - Credentials are provided using environment variables

Typical Applications/Workloads

The typical applications stacks are (but are not limited to):

- ASP.NET, C#, with IIS
- Java Tomcat
- Ruby on Rails (Passenger or Puma)
- NodeJS
- PHP
- Python
- Any generic Docker application

The typical databases are:

- Microsoft SQL Server (express, standard or web edition)
- Oracle
- MySQL
- MariaDB
- PostgreSQL
- MongoDB
- ElasticSearch
- Redis / Memcache

Employee workstation support:

- Active Directory Synchronization (e.g. for single sign-on capabilities)
- Email mailboxes (Amazon Workmail)

DevOps

The architecture allows for a DevOps approach:

- A git commit to develop branch can trigger a build workflow
    - The application can be automatically compiled (if applicable)
    - The application can run its tests: unit tests, regression tests, etc
    - If all tests succeed, the application can automatically be deployed
- Zero-downtime deployments are included in the standard production build
- Green/Blue deployments can be enabled, to make sure new deployments are healthy before they come live
- The deployment tools allow for a very short deployment cycle (multiple deployments per day are possible)

4. Support & Maintenance

Package:

Basic Support package	Max 1 codebase, max 1 application layer within development, QA, production
	€ 250 Total fee per month per application payable to in4it

All prices are excluding Belgian VAT (21%)

Includes:

Name	**Description**
Operating System Security updates	All Linux / Windows servers will receive operating system updates. Optionally, an agent can be installed which checks periodically for security vulnerabilities
Stand-by “best effort” SLA	The best effort Service Level Agreement doesn’t have any formal response times. You get a 24/7 telephone number to call to alert an engineer in case of any issues
Capacity Management	In4it will monitor the infrastructure and take action if any capacity issues arise.
Availability Management	In4it will monitor the production application for availability. If an unavailability event occurs, in4it will be notified and take action (using the “best effort” SLA)
Pro-active monitoring & alerts	Alerts will be set on some key metrics to make sure in4it is notified when inconsistencies would occur

Excludes:

Name	**Description**
Amazon AWS Services	Amazon AWS Services are charged directly to the customer