Degree Thesis, Åland University of Applied Sciences, Degree Programme in Information Technology

ÅLAND'S LUNCH SCHNITZEL OF THE DAY

Patrik Heimdahl

46:2017

Publishing date: 15.12.2017
Supervisor: Agneta Eriksson-Granskog

DEGREE THESIS
Åland University of Applied Sciences

Study program:	Information Technology
Author:	Patrik Heimdahl
Title:	Åland's lunch schnitzel of the day
Academic Supervisor:	Agneta Eriksson-Granskog
Technical Supervisor:	

Abstract

This degree thesis describes the steps involved in developing a simple web application and deploying it using public cloud services.

The application is written in Java using libraries like Spring Boot, Jsoup and Thymeleaf.

The infrastructure is set up using third-party services like Amazon Web Services, Cloudflare, GitHub, Travis CI and Pingdom.

The result of this project was a web service running at near zero daily cost and no active maintenance requirements with, at the time of writing, 100-150 unique users per week and around half that being returning visitors.

Keywords

Java, Spring, Thymeleaf, CI, AWS, CloudFlare

Serial number:	ISSN:	Language:	Number of pages:
46:2017	1458-1531	English	26 pages

Handed in:	Date of presentation:	Approved on:
15.12.2017	01.12.2017	15.12.2017

EXAMENSARBETE
Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Patrik Heimdahl
Arbetets namn:	Dagens lunchschnitzel på Åland
Handledare:	Agneta Eriksson-Granskog
Uppdragsgivare:	

Abstrakt

Detta examensarbete beskriver utvecklingen av en simpel webbapplikation och provisioneringen av denna i publika molntjänster.

Applikationen är skriven i Java och använder bibliotek som Spring Boot, Jsoup och Thymeleaf.

Infrastrukturen är konfigurerad med tredjepartstjänster som Amazon Web Services, Cloudflare, GitHub, Travis CI och Pingdom.

Resultatet av detta projekt är en webbtjänst med en månadskostnad på nära noll utan behov av daglig drift med 100-150 unika besökare per vecka och ungefär hälften av dem återkommande besökare.

Nyckelord (sökord)

Java, Spring, Thymeleaf, CI, AWS, CloudFlare

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
46:2017	1458-1531	Engelska	26 sidor

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
15.12.2017	01.12.2017	15.12.2017

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this degree thesis is to illustrate how I chose to implement a simple web application and deploy it to a public cloud. The application in question is *Dagens lunchschnitzel på Åland* (schnitzel.ax), a website for looking up which restaurants on Åland serve a schnitzel based dish for lunch on the current weekday.

Cloud computing has become immensely popular in the last decade and with several vendors providing free tier service levels it was the obvious choice for my degree thesis project instead of hosting services on my own servers.

The idea of a website like schnitzel.ax, as well as schnitzel being the national dish of Åland, has been a long-running joke among my coworkers. When starting this project I had recently joined a company where Amazon's cloud services were being used on a day-to-day basis and I wanted to set up something from scratch on the platform. Several cloud-based services are used in this project and cost efficiency is a continuous concern throughout the planning and implementation phase. Section 3 (*The Infrastructure*) will introduce most services and third-party products used in this project.

1.2 Method

The programming language and tools used in the implementation were all previously known to me, so a prototype was developed and deployed to AWS in one weekend. Improvements were made continuously thereafter. Some of the cloud services used were previously known to me. Official manuals and guides, as well as trial and error was used for products and services that I was not previously familiar with. I also drew on my 6 years of experience working in software development and application hosting.

1.3 Scope & delimitations

The goal of this project has been to launch a website where the public can easily go to find out where they can get a schnitzel for lunch.

The schnitzel.ax MVP (minimum viable product) has from the start had these requirements:

- Near zero daily cost
- Day-to-day tasks fully automated
- Application hosted in a public cloud
- Source code open-sourced through a public Git repository

These features were also implemented:

- Usage analysis with Google Analytics
- CI (continuous integration) to verify code changes
- Encrypted web traffic between end users and schnitzel.ax
- DDoS (distributed denial of service) protection
- Facebook page plugin integration

Some features were cut out due to time limitations:

- Centralized logging
- Deployment pipeline where the application is automatically re-deployed after a successfully tested code change
- Static code analysis through e.g. SonarQube

Ideas for further development are presented in section 4 (*Results*).

2. THE APPLICATION

Dagens lunchschnitzel på Åland is written in Java (Oracle, 2017) and runs in a Spring Boot managed container. The code is built with Gradle (Gradle Inc., 2017), hosted on GitHub and verified in Travis CI. More on GitHub and Travis CI in section 3 (*The Infrastructure*).

Spring (Pivotal Software, Inc., 2017) is a collection of open source application frameworks and libraries for the Java programming language. Two of their over 20 projects (Pivotal Software, Inc., 2017) are used in this project. *Spring Framework* provides dependency injection and a model-view-controller (MVC) framework, among other things. *Spring Boot* allows the developer to auto-configure a lot of Spring Framework components and embed a web server, meaning that there is no need for a separate application server to actually run the application.

Jsoup (Hedley, 2017) is an open source Java library used to parse and modify HTML documents. The library also bundles a web client that can be used to read HTML over HTTP, making it relatively easy to build a web scraper, spider or crawler. This library is used to download and parse *Lunchguiden*, the data source of schnitzel.ax.

2.1 Application backend

Dagens lunchschnitzel på Åland's backend is developed in Java with a Domain Driven Design (DDD) inspired structure. The application source is divided into four sections; application, domain, infrastructure and interfaces, as seen in figure 1.

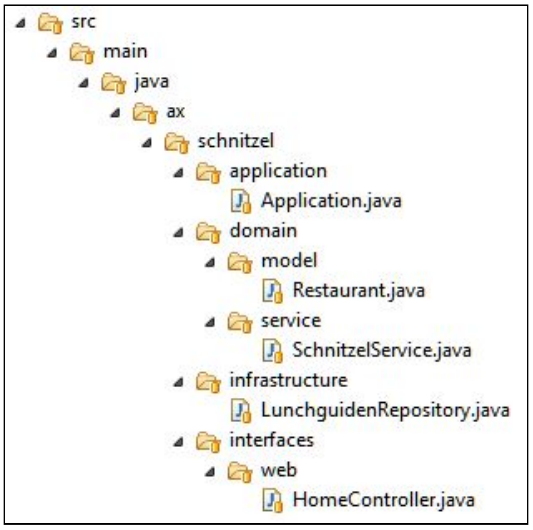


Figure 1. Java code file structure

The domain layer is divided into domain models and domain services. The domain model describes data models used in the software and the domain service layer houses application logic and is an extension and abstraction of the infrastructure layer. See how Spring's "Scheduled" annotation is used to schedule a cache update in figure 2.

```
/** Scheduled method for updating the local cache of restaurants. */
@PostConstruct
@Scheduled(cron = "0 0 0,9,10,11,12,13 * * MON-FRI")
public void updateRestaurants() {
    try {
        List<Restaurant> newlist = lunchguiden.getRestaurants();
        synchronized (cachedRestaurants) {
            cachedRestaurants.clear();
            cachedRestaurants.addAll(newlist);
            this.lastUpdate = new Date();
        }
    }
    catch (Exception e) {
        LOG.error(e.getMessage(), e);
    }
}
```

Figure 2. Scheduled update of the local cache

The infrastructure layer creates domain models by sending HTTP requests to and parsing the response of aland.com's *Lunchguiden* site. A HTML parsing library called Jsoup is used to traverse the HTML markup and extract text from the website. See figure 3 for an example of how Lunchguiden's HTML markup can be traversed and have data extracted from it.

```
List<Restaurant> parse(Document lunchguiden) {

    List<Restaurant> restaurants = new ArrayList<Restaurant>(0);

    // TODO: Convert to Java 8 Lambdas?
    for (Element restaurantElement : lunchguiden.getElementById("restaurants").getElementsByClass("restaurant")) {
        Element menuElement = restaurantElement.getElementsByClass("restaurant_menu").get(0);

        if (StringUtils.containsIgnoreCase(menuElement.toString(), "schnitzel")) {

            final Restaurant restaurant = new Restaurant();
            restaurant.setId(restaurantElement.id());
            restaurant.setName(restaurantElement.getElementsByClass("header_left").get(0).text());

            LOG.info("Setting up {}", restaurant.getName());

            List<String> dishes = new ArrayList<String>(0);

            for (Element menuItem : menuElement.getElementsByTag("li")) {

                // TODO: Use fuzzy searching in case "schnitzel" is misspelled in the menu?
                // See https://en.wikipedia.org/wiki/Approximate_string_matching

                if (StringUtils.containsIgnoreCase(menuItem.toString(), "schnitzel")) {
                    dishes.add(menuItem.text());
                    LOG.info("Adding \"{}\" to {}", menuItem.text(), restaurant.getName());
                }
            }

            restaurant.setDishes(dishes);
            restaurants.add(restaurant);

        }
    }
    return restaurants;
}
```

Figure 3. Using Jsoup to traverse the HTML markup and extract restaurants and their menus

Unit tests for LunchguidenRepository are implemented in JUnit. There are two test cases that verifies the repository's parsing of *Lunchguiden*; one case with two restaurants serving schnitzels and one case with no restaurants serving schnitzel. See figure 4 for an example of JUnit unit tests verifying HTML parsing implemented in Jsoup.

```java
@Test
public void test_20171019() throws IOException {

    Document doc = Jsoup.parse(this.getClass().getClassLoader().getResourceAsStream("ax/schnitzel/lunchguiden_2017-10-19.html"), "UTF-8", url);
    List<Restaurant> restaurants = lunchguidenRepository.parse(doc);
    assertTrue(restaurants.size() == 2);

    // Validate Bistro Savoy
    assertTrue(StringUtils.equals(restaurants.get(0).getId(), "restaurant_7"));
    assertTrue(StringUtils.equals(restaurants.get(0).getName(), "Bistro Savoy"));
    assertTrue(restaurants.get(0).getDishes().size() == 1);
    assertTrue(StringUtils.equals(restaurants.get(0).getDishes().get(0), "Fläskschnitzel med champinjonsås och pommes frites"));

    // Validate Sittkoffska
    assertTrue(StringUtils.equals(restaurants.get(1).getId(), "restaurant_53"));
    assertTrue(StringUtils.equals(restaurants.get(1).getName(), "Restaurang Sittkoffska Gården"));
    assertTrue(restaurants.get(1).getDishes().size() == 1);
    assertTrue(StringUtils.equals(restaurants.get(1).getDishes().get(0), "Schnitzel med vitlökssmör, citron och röstipotatis (H)"));
}

@Test
public void test_20171023() throws IOException {

    Document doc = Jsoup.parse(this.getClass().getClassLoader().getResourceAsStream("ax/schnitzel/lunchguiden_2017-10-23.html"), "UTF-8", url);
    List<Restaurant> restaurants = lunchguidenRepository.parse(doc);
    assertTrue(restaurants.isEmpty());
}
```

Figure 4. Unit tests verifying that the repository can correctly parse Lunchguiden's HTML.

The interfaces layer uses the domain service layer to get domain models and prepare them for presentation to the end user. See figure 5 for an example of a Spring MVC controller classing storing objects in the request scope.

```java
@RequestMapping("/")
public ModelAndView home(final Model model) {

    model.addAttribute("restaurants", schnitzelService.getRestaurants());

    Date lastUpdate = schnitzelService.getLastUpdate();
    model.addAttribute("lastUpdated", lastUpdate);
    model.addAttribute("updatedToday", lastUpdate != null && DateUtils.isSameDay(lastUpdate, new Date()));

    DayOfWeek dayOfWeek = LocalDate.now().getDayOfWeek();
    model.addAttribute("isWeekend", (dayOfWeek == DayOfWeek.SATURDAY || dayOfWeek == DayOfWeek.SUNDAY));

    return new ModelAndView("home");
}
```

Figure 5. Spring MVC controller class storing objects in the request scope.

The application layer handles application configuration and configures Spring Framework through Spring Boot.

2.2 Application frontend

Dagens lunchschnitzel på Åland uses a templating engine called Thymeleaf to programmatically modify HTML markup based on the domain models and other helpful variables provided by the Spring Controller. See figure 6 for an example of Thymeleaf attributes on HTML tags for simple logic and iteration control.

```
<th:block th:unless="${restaurants.empty}">
    <dl>
        <th:block th:each="restaurant : ${restaurants}">
            <dt><a data-toggle="collapse" th:attr="href='#'+${restaurant.id}+'_menu'" th:text="${restaurant.name}" class="text-secondary"></a></dt>
            <div class="collapse" th:id="${restaurant.id}+'_menu'" style="padding-left: 10px">
                <dd class="text-muted font-italic" th:each="dish : ${restaurant.dishes}" th:text="${dish}" />
            </div>
        </th:block>
    </dl>
</th:block>
```

Figure 6. Thymeleaf attributes on HTML tags for simple logic and iteration control

The Bootstrap (Bootstrap, 2017) framework is used for UI elements and animations in this project. Other options for UI frameworks include Foundation and jQuery UI. Bootstrap was chosen for its familiarity. See figure 7 for the final result.

Figure 7. Screenshot of schnitzel.ax on a mobile device with all restaurants expanded to show the dish being served

3. THE INFRASTRUCTURE

"The cloud" is a term used to describe where cloud computing occurs, which is generally a term for IT services provided over the internet.

Common cloud computing service models include:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

The difference between these models is basically where the line of abstraction towards the customer is drawn, as seen in figure 8.

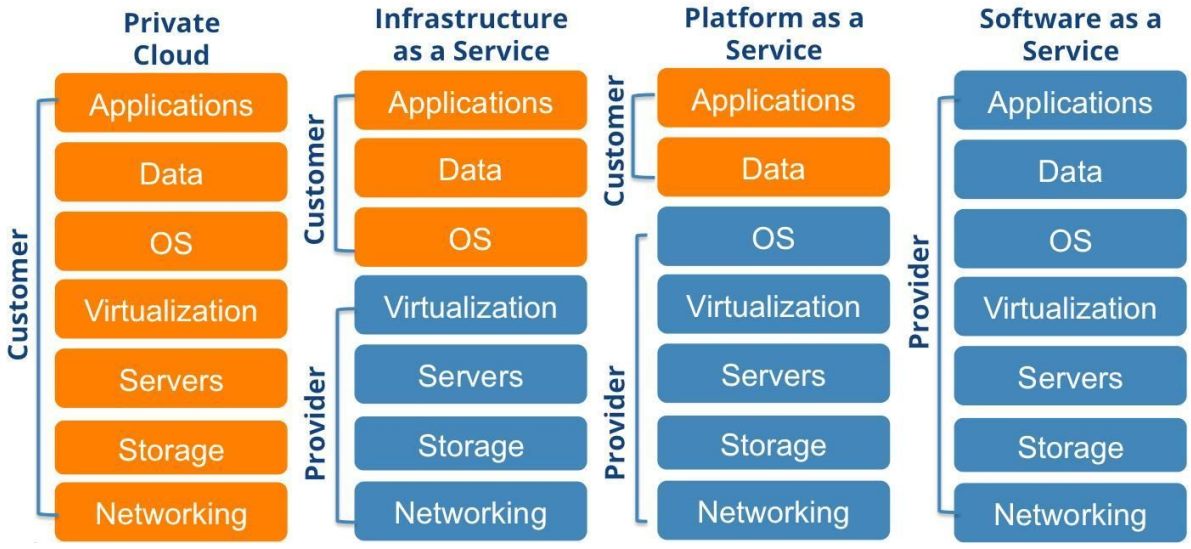


Figure 8. Illustration of different cloud service models and what they include (SevOne)

Cloudflare, Inc. is a US based company providing web performance and security services. This project uses uses Cloudflare in two ways: as a reverse proxy for security reasons as well as an authoritative name server for managing schnitzel.ax. See section 3.2 and 3.4 for more information on DNS and public endpoint security.

See figure 14 for a full map of this project's infrastructure.

3.1 Revision control and continuous integration

Revision control is used in software development to keep a log of changes made to the code base. There are several options when choosing which revision control software to use, but I went for the most popular software – Git. A Git repository can be hosted on the developer's machine or on a server. For this project I chose to host my Git repository on the cloud service GitHub (GitHub, Inc., 2017a), the largest version control repository in the world with ~20M users and ~57M repositories (GitHub, Inc., 2017b)

Continuous Integration (CI) is the concept of continuously merging changes made by developers to a common mainline. In the most basic form this means that all developers work on a single branch of the code base, so that when they commit changes to the revision control system the changes are automatically integrated. If a conflict arises it's up to the developer to fetch the latest changes from the mainline and manually merge the incoming changes with his outgoing changes.

CI also includes automation of the software build process, i.e. verifying the changes when they are integrated into the mainline. This is usually done by compiling the source code (when applicable) and running a set of unit tests. The build automation can also be extended to run different kinds of static code analysis and integration tests on each commit.

Travis CI (Travis CI, GmbH, 2017) was chosen to handle the automated builds and unit test executions in this project. Travis CI is free for open source projects and is easily integrated with GitHub.

3.2 Domain Name System (DNS)

The Domain Name System (DNS) is a semi-decentralized naming system most commonly used to map static human-friendly addresses to optionally dynamic IP addresses.

The first step in most web-based hobbyist projects is to register a domain name, especially if branding is an important factor. For example the domain name in this project, schnitzel.ax, was considered important for recognizability and rememberability.

There have been reports of *domain name front running* at some registrars, the practice of registering domain names that have been looked up by customers. This was however not considered a risk in this project for two reasons: the limited popularity of the .ax TLD and because domain availability searches are done on the domain registry's official WHOIS page.

This project's domain was registered through *Ålands IT-konsulter*, one of multiple .ax registrars. Schnitzel.ax was initially managed by AWS' *Route 53* service since Cloudflare requires new sites to be functional before migration to their DNS infrastructure. Once the domain had been added to Cloudflare new name servers were provided to whois.ax.

Cloudflare hosts one of the largest authoritative DNS networks in the world with a market share of over 38% (Cloudflare, Inc., 2017). Authoritative name servers is the part of the Domain Name System that manages individual domain names, as opposed to the domain name registry which manages top level domains. See figure 9 where all DNS zones involved are visualized through a DNS tracing utility.

```
patrik@dev01:~$ dig +trace schnitzel.ax | grep -Ev "RRSIG|DS"

; <<>> DiG 9.9.5-3ubuntu0.16-Ubuntu <<>> +trace schnitzel.ax
;; global options: +cmd
.                       236724  IN      NS      d.root-servers.net.
.                       236724  IN      NS      i.root-servers.net.
.                       236724  IN      NS      m.root-servers.net.
.                       236724  IN      NS      h.root-servers.net.
.                       236724  IN      NS      j.root-servers.net.
.                       236724  IN      NS      f.root-servers.net.
.                       236724  IN      NS      g.root-servers.net.
.                       236724  IN      NS      k.root-servers.net.
.                       236724  IN      NS      b.root-servers.net.
.                       236724  IN      NS      a.root-servers.net.
.                       236724  IN      NS      c.root-servers.net.
.                       236724  IN      NS      l.root-servers.net.
.                       236724  IN      NS      e.root-servers.net.
;; Received 1097 bytes from 192.168.1.1#53(192.168.1.1) in 20 ms

ax.                     172800  IN      NS      ns1.aland.net.
ax.                     172800  IN      NS      ns2.aland.net.
ax.                     172800  IN      NS      ns3.alcom.fi.
ax.                     172800  IN      NS      ns4.alcom.fi.
;; Received 585 bytes from 192.58.128.30#53(j.root-servers.net) in 23 ms

schnitzel.ax.           86400   IN      NS      ray.ns.cloudflare.com.
schnitzel.ax.           86400   IN      NS      rita.ns.cloudflare.com.
;; Received 819 bytes from 194.112.0.5#53(ns2.aland.net) in 7 ms

schnitzel.ax.           300     IN      A       104.18.33.202
schnitzel.ax.           300     IN      A       104.18.32.202
;; Received 73 bytes from 173.245.59.138#53(ray.ns.cloudflare.com) in 8 ms

patrik@dev01:~$
```

Figure 9. DNS trace look-up of schnitzel.ax using the dig utility

3.3 Amazon Web Services (AWS)

Dagens lunchschnitzel på Åland is hosted on Amazon Web Services (AWS). Amazon Web Services, a subsidiary of Amazon.com, Inc., is the largest cloud service provider with a market share of around 35% (Synergy Research Group, 2017). Offering over one hundred products (Amazon Web Services, Inc., 2017) AWS could provide all third-party services for this project. Alternative public cloud providers include Google (App Engine), Microsoft (Azure) and IBM (Bluemix). All of them provide free-tier services but AWS was chosen for its popularity and familiarity.

Due to cost limitations and products deemed inferior to other options only Elastic Compute Cloud (EC2) and Simple Storage Service (S3) are used.

3.3.1 Server instances and auto scaling

Dagens lunchschnitzel på Åland utilizes an AWS EC2 feature called *Auto Scaling group* (ASG). ASG allows the administrator to easily scale their pool of server instances in case of changes in demand.

Since the project is limited to AWS' free tier only one *t2.micro* instance is active in the ASG. The ASG desired capacity setting is increased and then decreased when there is a need to redeploy a new version of the service without causing any downtime for the end user. Figure 10 displays some of an ASG's configuration options.

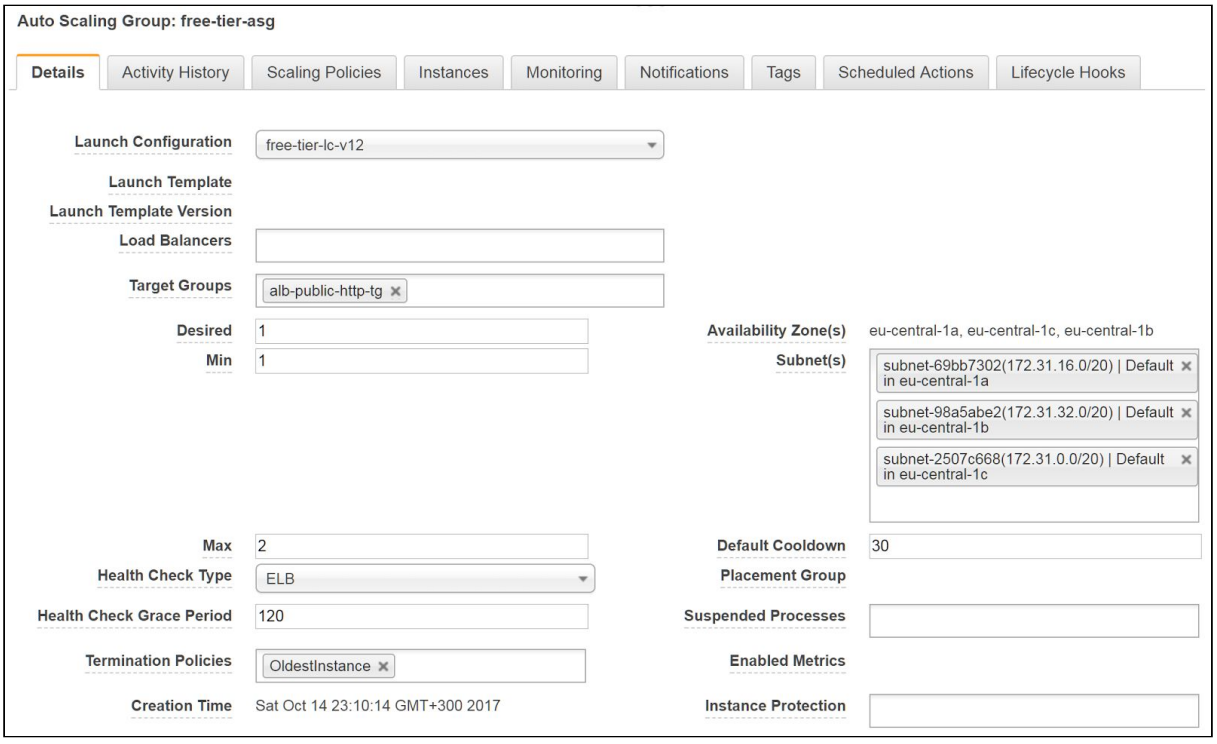
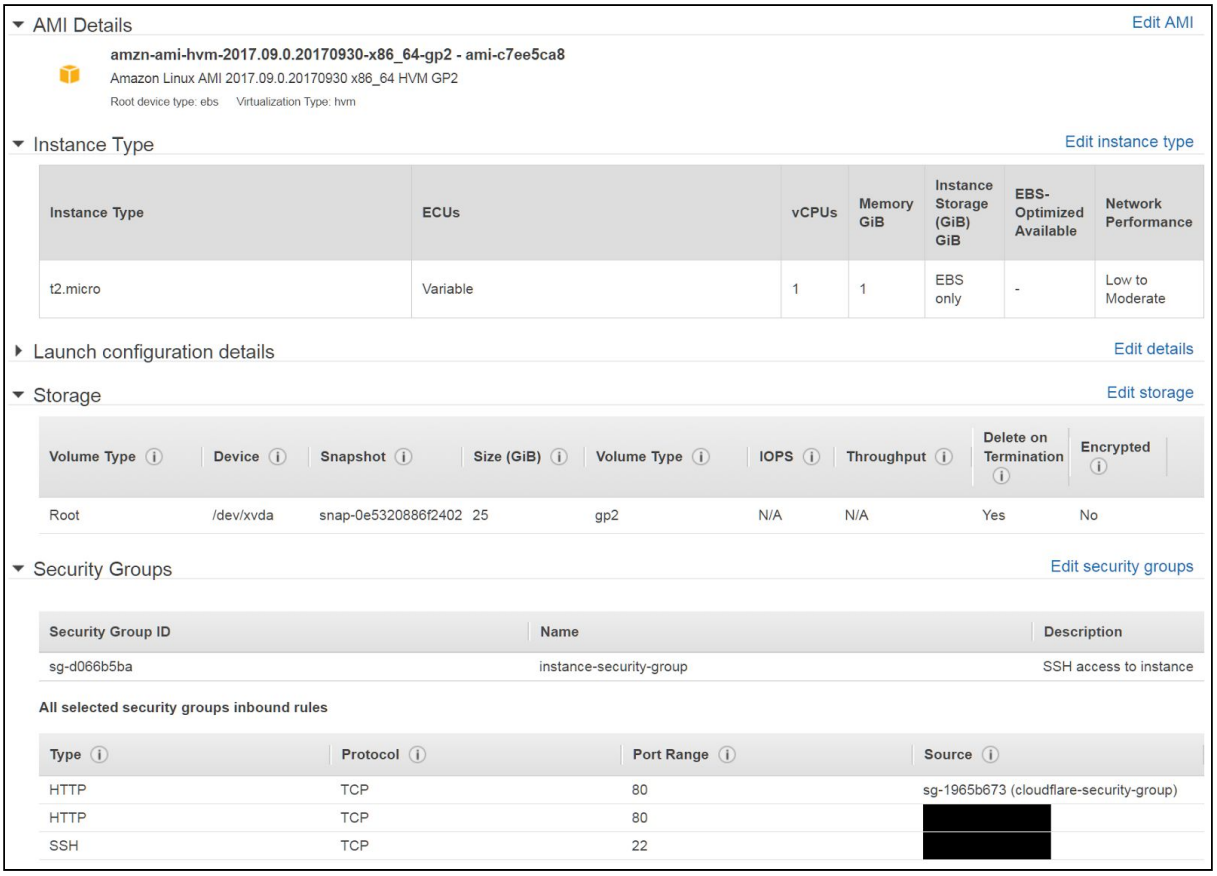


Figure 10. Auto Scaling group configuration

The ASG uses server instance definitions known as "Launch Configurations" (LC) when starting new instances. An LC contains information about the instance size (*t2.micro* in this case), which Amazon Machine Image (AMI) to use (Amazon Linux in this case), Elastic Block Storage (EBS) devices, which network Security Group (SG) to belong to and pre-set commands to run on installation (called "user data") and much more. See figure 11 for details on the LC used in this project.



AMI Details — Edit AMI

amzn-ami-hvm-2017.09.0.20170930-x86_64-gp2 - ami-c7ee5ca8
Amazon Linux AMI 2017.09.0.20170930 x86_64 HVM GP2
Root device type: ebs Virtualization Type: hvm

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory GiB	Instance Storage (GiB) GiB	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Launch configuration details — Edit details

Storage — Edit storage

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput	Delete on Termination	Encrypted
Root	/dev/xvda	snap-0e5320886f2402	25	gp2	N/A	N/A	Yes	No

Security Groups — Edit security groups

Security Group ID	Name	Description
sg-d066b5ba	instance-security-group	SSH access to instance

All selected security groups inbound rules

Type	Protocol	Port Range	Source
HTTP	TCP	80	sg-1965b673 (cloudflare-security-group)
HTTP	TCP	80	[redacted]
SSH	TCP	22	[redacted]

Figure 11. EC2 Launch Configuration example

3.3.2 Load Balancing

Normally you wouldn't think to use a load balancer with only one server instance, but this project uses an Application Load Balancer (ALB) for two reasons.

Firstly to to facilitate rolling upgrades of server instances. Performing a rolling upgrade means that server instances are re-deployed with no downtime experienced by end users.

Secondly to provide a static endpoint for the Cloudflare reverse proxy to access. Without using the ALB as a middleman it would be necessary to either implement a dynamic DNS or update the reverse proxy each time a new instance is launched.

The ALB points to a static target group (TG) that new instances are put in automatically by the ASG.

3.3.3 Network & security

Security groups allows the administrator to define a set of whitelisted incoming and outgoing network connections. Two security groups are used in this project: *cloudflare-security-group* and *instance-security-group*.

The group *cloudflare-security-group* is applied to the ALB and allow HTTP access from Cloudflare's reverse proxies. See figure 12 where inbound connections to the load balancer is defined.

The group *instance-security-group* is applied to the ASG instances and allows incoming HTTP connections from the ALB as well as incoming HTTP and SSH connections from the administrator's home IP address.

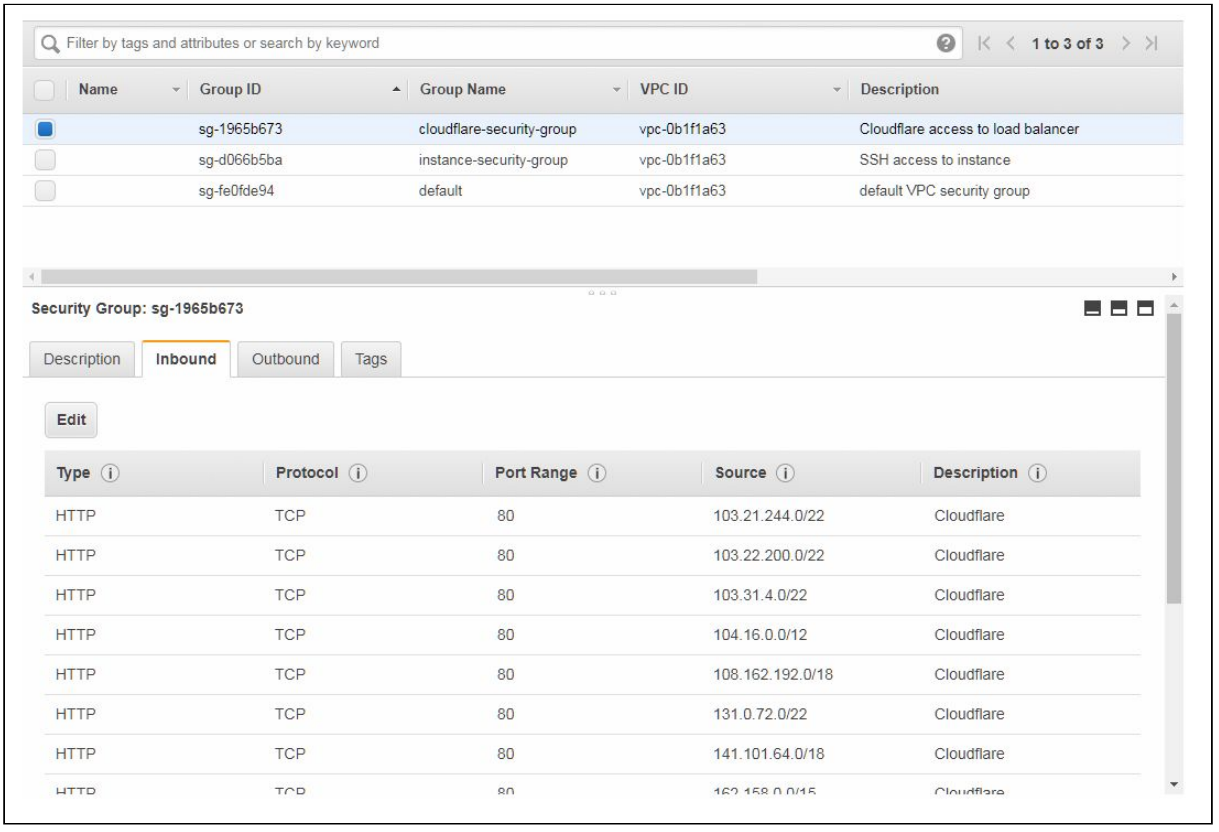


Figure 12. Allowed inbound connections to the load balancer's security group

3.4 Public endpoint security

Dagens lunchschnitzel på Åland uses Cloudflare's reverse proxy for increased security. Denial-of-service (DoS) attack protection is not a necessity in this project, and neither is protection against malicious users, but since these services are just a mouse click away when using Cloudflare's DNS infrastructure enabling them was not a hard choice to make.

Another useful feature of Cloudflare's reverse proxy is traffic encryption between the end user and Cloudflare's endpoint. This feature is turned on for schnitzel.ax. Traffic between Cloudflare and AWS is however sent unencrypted since there is no sensitive data and customer sessions involved.

3.5 Centralized logging

Since hosting applications in the public cloud often mean short-lived server instances it's important to push application logs to a central location. One option would be to use a shared storage volume, another would be to use a specialized log collection and analysis system. The second option was deemed the better option. When it comes to cloud-based log collection tools there are several options to choose from: e.g. Datadog, Splunk and Sumo Logic. For this project I chose Sumo Logic since it has a generous free-tier offer and I have previously used the software. Unfortunately centralized logging had to be cut from the scope of this project due to time constraints.

3.6 Monitoring

Monitoring suitable for this project can be split into two areas: system monitoring and website monitoring.

System monitoring allows the administrator to continuously check up on system resource usages and from that make decisions on necessary actions in regards to scaling up and down the environment. It would also allow the administrator to monitor separate components in the systems and it could trigger alerts if any component fails. Due to time limitations and the small size of this system such monitoring has not been implemented.

Website monitoring does on the other hand does not monitoring anything inside the system, instead it looks at the publicly available endpoints and simulates an end-user all the while collecting metrics. There are multiple popular services for this kind of monitoring. The services considered in this project were Uptime Robot, StatusCake and Pingdom. Since cost is a constant factor Pingdom seemed the best fit since they're the only ones offering 1 minute interval checks for free.

Due to emails easily being missed Pingdom has been set up with an integration towards a Slack chat channel where service interruption notifications are sent. The Slack mobile client can be configured to audibly notify the owner when a new incident occurs. See figure 13 where a service interruption on November 21st was reported.

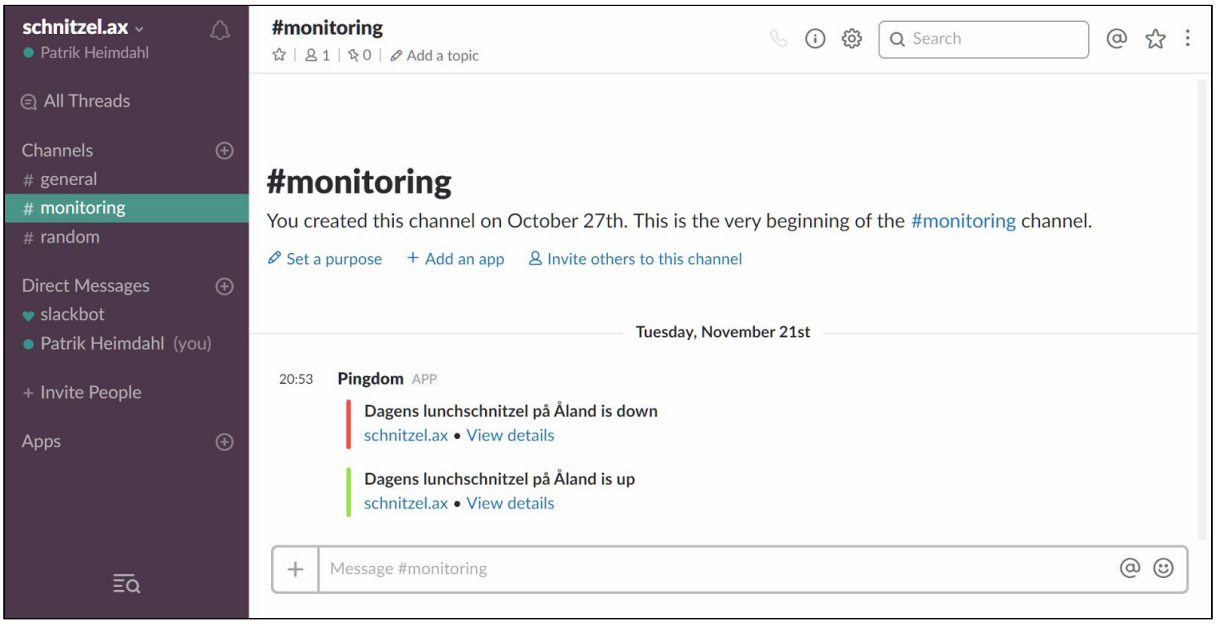


Figure 13. Website incident on 2017-11-21 at 20:53

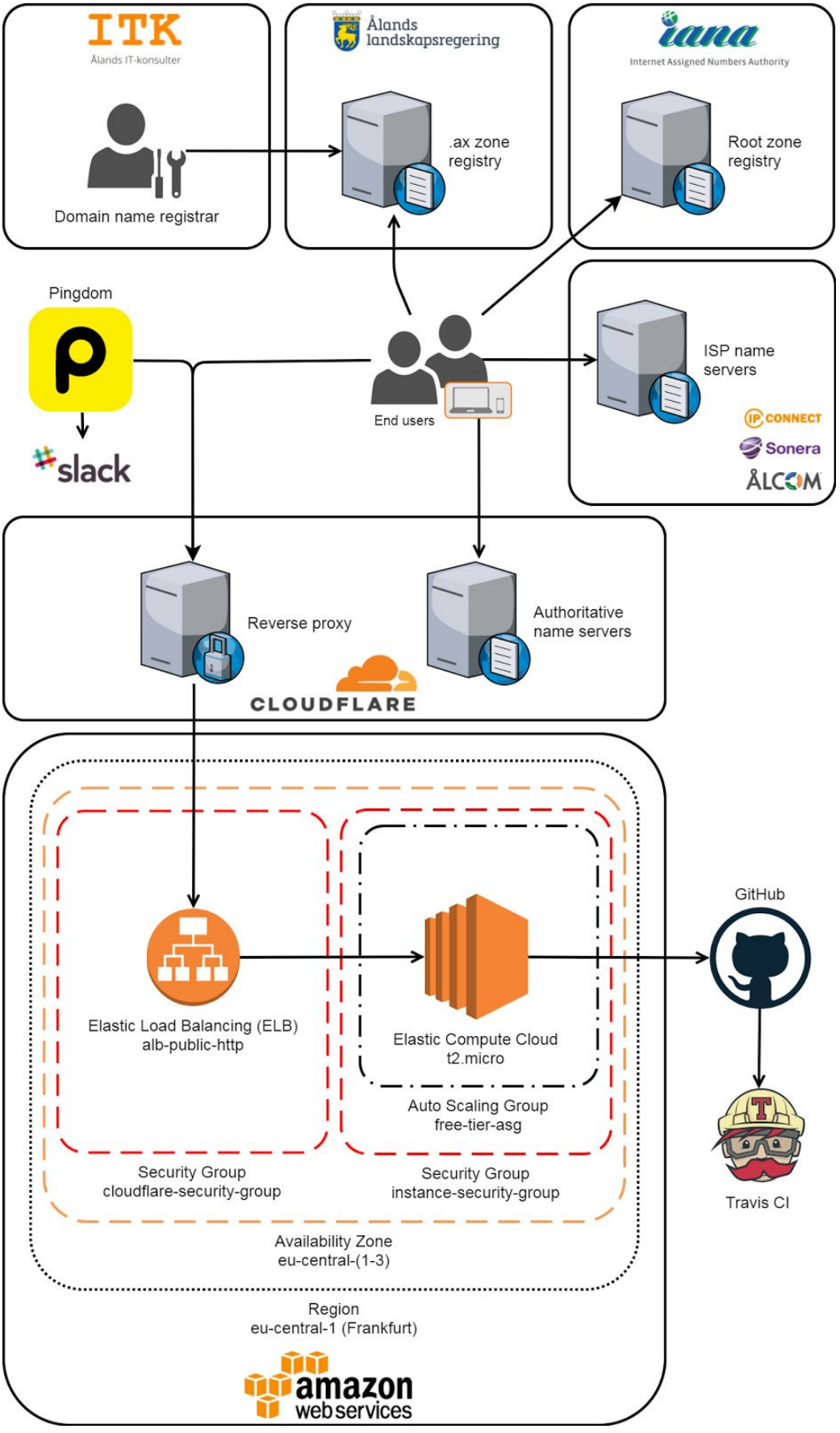


Figure 14. Visualization of the full schnitzel.ax infrastructure and related third-party services

4. CONCLUSIONS

4.1 Results

The main goal of providing a low-cost, low-maintenance schnitzel-of-the-day website to the public was successful.

Just after the official launch of schnitzel.ax there was a big rush of traffic lasting for a couple of days. During the first week schnitzel.ax had 1083 visits from 936 unique users. See figure 15 and 16 for charts and details of schnitzel.ax's usage.

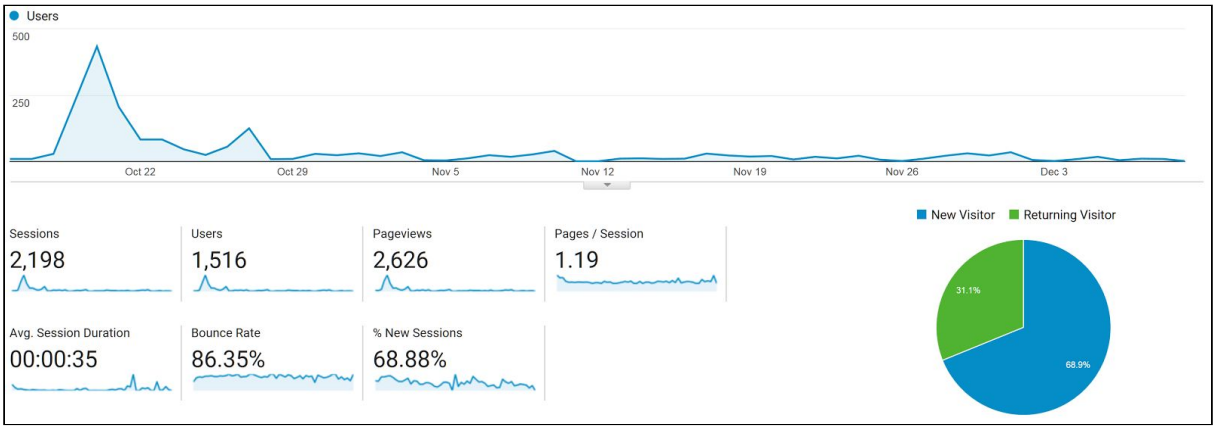


Figure 15. Daily number of users from 2017-10-16 to 2017-12-09

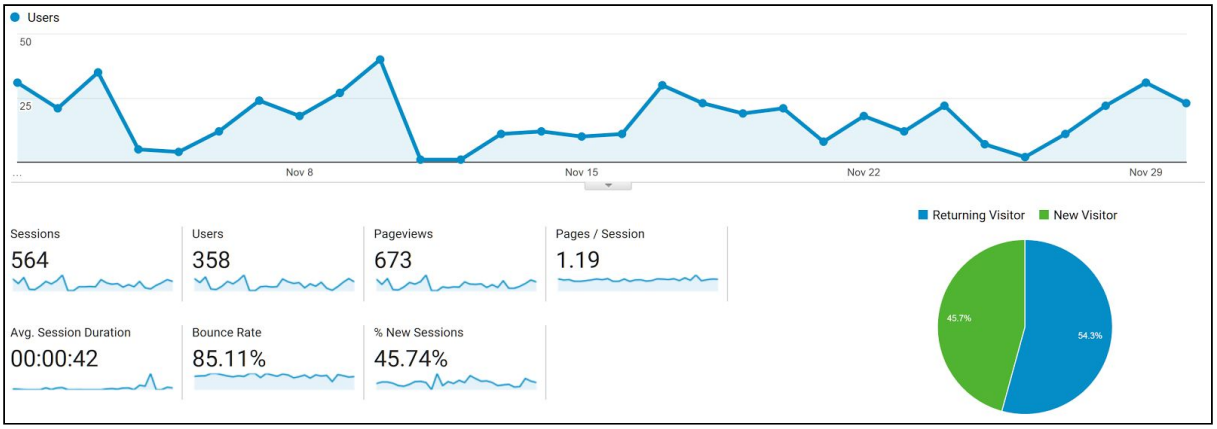


Figure 16. Daily number of users in November

The website has proven very stable with only three service interruptions noticed by Pingdom, resulting in an uptime measure of over 99.99%. See figure 17 for details regarding website performance and uptime.

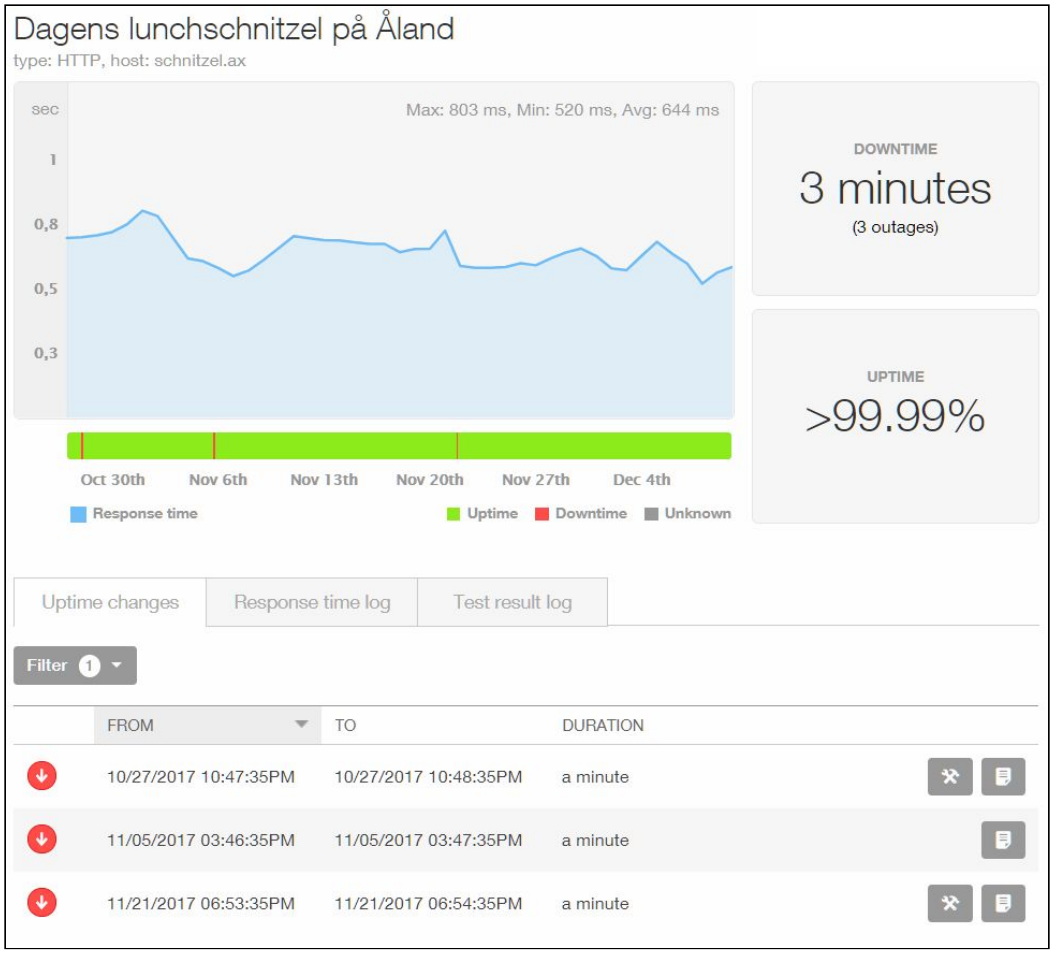


Figure 17. Pingdom uptime report for the period 2017-10-27 to 2017-12-10

Some future development ideas include:

- AMP (Accelerated Mobile Pages), a way of decreasing the load time of webpages on mobile devices.
- Deployment pipeline where the production servers are redeployed after a successfully tested code change pushed to GitHub.
- Restaurant ratings, either by schnitzel.ax's users or through some third party like Google Maps or Facebook.
- Facebook integration, posting updates of which restaurants serve schnitzels on e.g. every Friday.

4.2. Reflections

I have gotten a good insight into the basics of Amazon Web Services, which was my main personal goal for this project. The MVP was reasonably sized and both the project and thesis scope were flexible enough that I did not have to worry about finishing on time or running out of things to do and write. I wish there would have been time to implement centralized logging and a proper deployment pipeline, two features very compelling to me.

In hindsight it would have been interesting to properly plan the project, do time estimates for tasks and keep a log of what I worked on when and for how long. It would also have been good to write down the problems I encountered and how I solved them. No major problems come to mind but of course there is some trial and error when using new technologies and services.

If you ever find yourself on Åland at lunchtime and you fancy a schnitzel, you now know where to look – at least until the 12 month AWS free tier period runs out…

REFERENCES

Oracle. (2017). Java Software. Retrieved 2017-12-04 from https://www.oracle.com/java/index.html

Pivotal Software, Inc. (2017). Spring Boot. Retrieved 2017-12-04 from https://projects.spring.io/spring-boot/

Jonathan Hedley. (2017). Jsoup Java HTML Parser, with best of DOM, CSS, and jquery. Retrieved 2017-12-06 from https://jsoup.org/

Gradle Inc. (2017). Gradle Build Tool. Retrieved 2017-12-04 from https://gradle.org/

Bootstrap. (2017). The most popular HTML, CSS, and JS library in the world. Retrieved 2017-12-10 from https://getbootstrap.com/

SevOne. (n.d.). Monitoring Cloud Infrastructure Performance to Eliminate Visibility Gaps. Retrieved 2017-12-10 from https://sevone.com/white-paper/monitoring-cloud-infrastructure-performance-eliminate-visibility-gaps

GitHub, Inc. (2017a). The world's leading software development platform. Retrieved 2017-12-04 from https://github.com/

GitHub, Inc. (2017b). Celebrating nine years of GitHub with an anniversary sale. Retrieved 2017-12-06 from https://github.com/blog/2345-celebrating-nine-years-of-github-with-an-anniversary-sale

Travis CI, GmbH. (2017). Test and Deploy Your Code with Confidence. Retrieved 2017-12-04 from https://travis-ci.org/

Cloudflare, Inc. (2017). Fast, Powerful, and Secure DNS. Retrieved 2017-12-10 from https://www.cloudflare.com/dns/

Synergy Research Group. (2017). Cloud Market Keeps Growing at Over 40%; Amazon Still Increases its Share. Retrieved 2017-12-06 from https://www.srgresearch.com/articles/cloud-market-keeps-growing-over-40-amazon-still-increases-share

Amazon Web Services, Inc. (2017). Cloud Products. Retrieved 2017-12-06 from https://aws.amazon.com/products/

Pingdom AB. (2017). Website performance and availability monitoring. Retrieved 2017-12-06 from https://www.pingdom.com/

Google LLC. (2017). Google Analytics. Retrieved 2017-12-06 from https://www.google.com/analytics/