# **CS5225 Cloud Computing – Preliminary Report**

Semester 2 – 2018/2019



# **Chaos Monkey**

|  |  |
| --- | --- |
| Chirumamilla, Jagadeesh | A0191509B |
| Kemper, Philipp | A0191365B |
| Maddi Kamal, Divya | A0178511E |
| Thiyagarajan, Dhakshayani | A0100461B |

1. Description / Motivation

Cloud computing promises to provide high(er) availability through redundancy. However, applications need to be engineered in a certain way in order to make use of HA features. Moreover, the failover and recovery process have to be exercised and maintained continuously. To perform system stability tests in a realistic rather than lab-like environment, chaos monkey kills (redundant) instances randomly to see if it breaks the system.

The objective is to build a SaaS system providing chaos monkey features for cloud services.

2. Business Model

Main idea is to charge the user per demand. Several options are possible.

1. Charge by the number of target systems
2. Charge by the time the chaos monkey instance is running (for example 1 week of chaos)
3. Charge by the time the chaos monkey instance is running multiplied by the number of target systems, maybe with discounts
4. Provide monitoring service as independent service, charged by number of monitored target systems

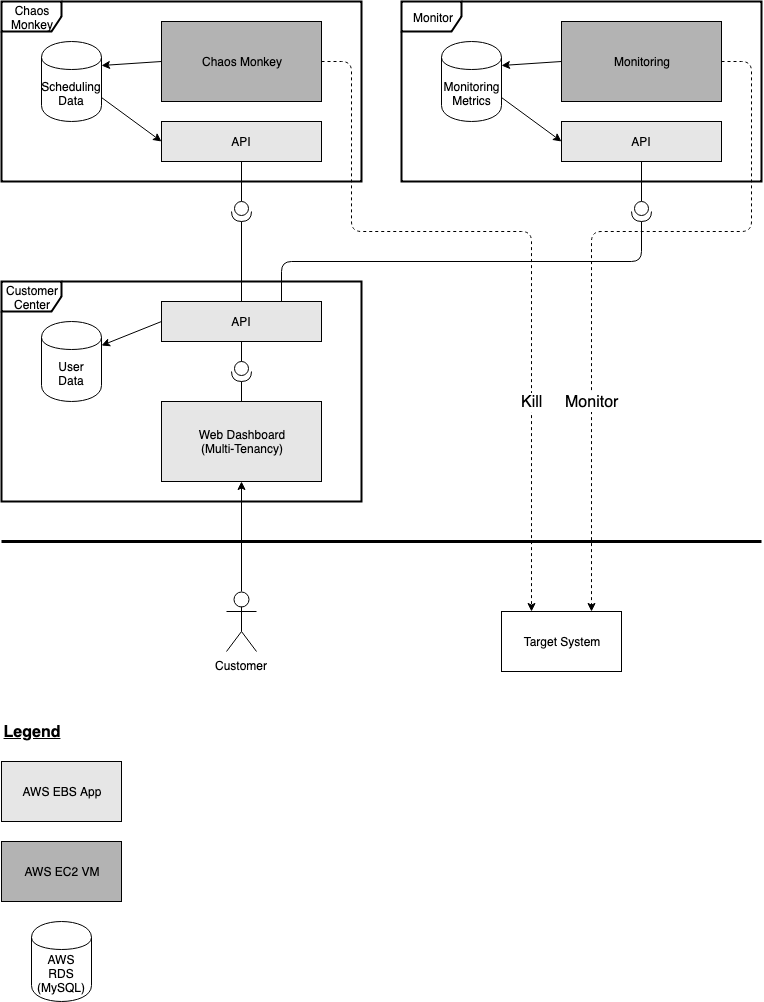
3. SaaS Services

From a customer's perspective the chaos monkey and its complimentary monitoring are used without any knowledge about the underlying platform. Customers can simply access a web-based application where they can manage their target systems, book chaos monkey instances and see a visualization of the results.

For the customers this is a ready-to-use product without any knowledge about the system itself. It therefore qualifies as a SaaS solution.

4. Preliminary Design

The system consists of 3 main components which are explained in greater detail.



Chaos Monkey

The Chaos Monkey an open source application developed by Netflix to test their own infrastructure. It’s written in Go and can be installed on any Linux system. The data for the scheduling and the results of the execution are stored in a MySQL database.

**Cloud Components**

* Chaos Monkey – AWS EC2 VM (Linux)
* API – AWS Elastic Beanstalk web app (Node.js)
* Scheduling database – AWS RDS (MySQL)

Monitor

The monitor constantly watches the customers target system to detect outages. It’s not connected to chaos monkey and could therefore be offered as an independent service.

**Cloud components**

* Zabbix – AWS EC2 VM (Linux)
* API – AWS Elastic Beanstalk web app (Node.js)
* Monitoring database – AWS RDS (MySQL)

Customer Center

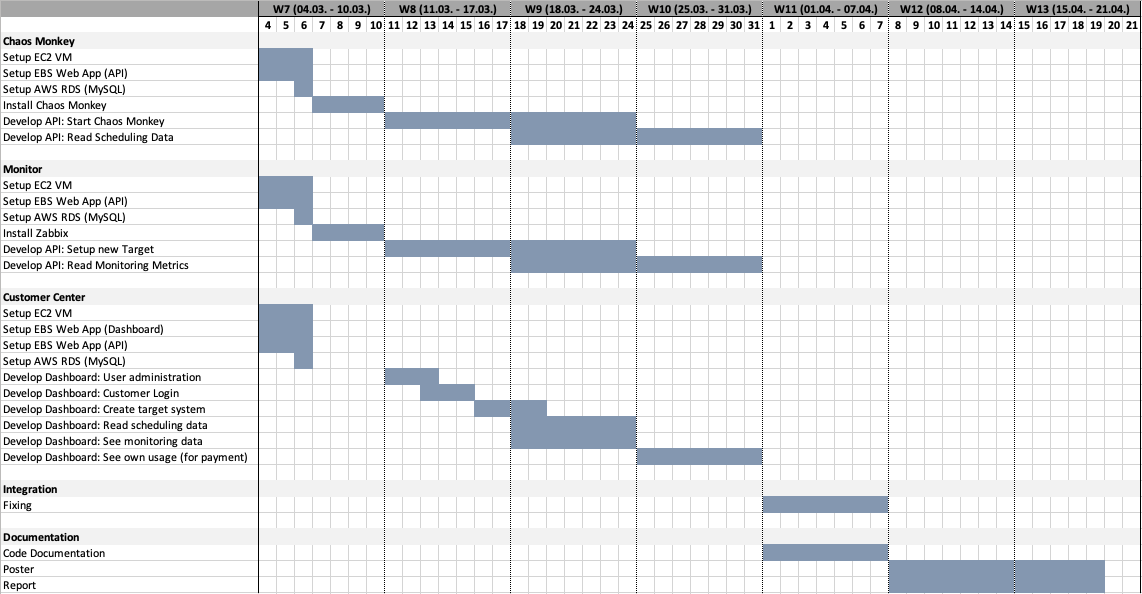
The customer center is a web application that lets the user add his target system, schedule a chaos monkey run on the target system and later collect the monitoring data. By combining chaos monkey results with the systems health data, the user can see which component was able to stop the system from functioning. In cases of automated disaster recoveries, it can also visualize how long it took until the system was back online.

Cloud components

* Dashboard – AWS Elastic Beanstalk web app (React.js, Node.js, Express.js)
* API – AWS Elastic Beanstalk web app (Node.js)
* User data database – AWS RDS (MySQL)

5. Implementation Plan

The three components can mainly be developed in parallel. If one components needs integration before the other component is developed, stubs can be used temporarily.



References:

1. chaos monkey - https://github.com/Netflix/chaosmonkey