Name: Robin Farrow-Yonge
Student Number: 160719011

Cloud Computing - Assessed Lab 1
Part 1: IaaS

Instance Types:

	r3.large	**m4.large**	**m3.medium**	**c3.large**
Service Scalability[1]:*	From: 2x 2.5Ghz vCPU* 15.25GiB memory 1x 32GB storage To: 32x 2.5Ghz vCPU 244GiB memory 2x 320GB storage	From: 2x 2.3Ghz vCPU* 8GiB memory (EBS storage) To: 64x 2.4Ghz vCPU 256GiB memory (EBS storage)	From: 1x 2.5Ghz vCPU* 3.75 GiB memory 1x 4GB storage To: 8x 2.5Ghz vCPU 30GB memory 2x 80GB storage	From: 2x 2.8Ghz vCPU* 3.75GiB memory 2x 16GB storage To: 32x 2.8Ghz vCPU 60GiB memory 2x 320GB storage
Service Availability:	99.95% Availability per region per month	99.95% Availability per region per month	99.95% Availability per region per month	99.95% Availability per region per month
Service Performance[2]:	vCPU*: 2 Clock: 2.5 Ghz RAM: 15.25 GiB Storage: 32 GB	vCPU*: 2 Clock: 2.3 Ghz RAM: 8 GiB Storage: EBS	vCPU*: 1 Clock: 2.5 Ghz RAM: 3.75 GiB Storage: 4 GB	vCPU*: 2 Clock: 2.8 Ghz RAM: 3.75 GiB Storage: 32 GB
On-Demand Cost[3]:*	Linux - $0.2 per hour	Linux - $0.129 per hour	Linux - $0.079 per hour	Linux - $0.129 per hour
Reserved Instance Cost[4]:*	No Upfront - $0.146 Partial Upfront - $0.126 All Upfront - $0.123	No Upfront - $0.098 Partial Upfront - $0.084 All Upfront - $0.082	No Upfront - $0.060 Partial Upfront - $0.055 All Upfront - $0.054	No Upfront - $0.096 Partial Upfront - $0.084 All Upfront - $0.083

[1] EC2 Instance Types – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/instance-types/
[2] EC2 Instance Types – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/instance-types/
[3] EC2 Instance Pricing – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/on-demand
[4] EC2 Instance Pricing – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

Dedicated Host Cost[5]:*	No Upfront - $2.591 Partial Upfront - $2.214 All Upfront - $2.170 On Demand - $3.521	No Upfront - $2.373 Partial Upfront - $2.028 All Upfront - $1.987 On Demand - $3.104	No Upfront - $2.259 Partial Upfront - $1.931 All Upfront - $1.892 On Demand - $2.772	No Upfront - $1.747 Partial Upfront - $1.493 All Upfront - $1.463 On Demand - $2.27
Spot Instance Cost[6]:*	Linux - $0.0222 per hour	Linux - $0.0325 per hour	Linux - $0.0121 per hour	Linux - $0.0205 per hour
Physical Server Cost:*	Server Cost: $773.24[7] Additional Storage: $62.77[8] Estimated Hourly Rate: $0.095	Server Cost: $167.06[9] Additional Memory: $67.47[10] Additional Storage: $62.77 Estimated Hourly Rate: $0.033	Server Cost: $167.06 Additional Storage: $62.77 Estimated Hourly Rate: $0.026	Server Cost: $224.54[11] Additional Storage: $62.77 Estimated Hourly Rate: $0.032

*Scalability: In the case of scalability I have used as the “From:” values the capacity of the chosen instance type, all of which are the lowest-capacity instance of their category (m3, r3, c3, etc), and for the “To:” value have selected the highest capacity instance in their category.

*vCPU: vCPU is the term used by Amazon (and others) to mean the number of virtual CPU cores a given instance has access to. The type of physical CPU used for each instance category, and the number of cores therein, varies. I have used vCPU in both the scalability and performance metrics, in an effort to make this clear.

---

[5] Amazon EC2 Dedicated Host Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/dedicated-hosts/pricing/

[6] Amazon EC2 Dedicated Host Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/dedicated-hosts/pricing/

[7] https://www.serversdirect.co.uk/p/1111246/hpe-proliant-dl0-gen9-intel-xeon-e3-130v5-quad-core-3.40ghz-8mb-16gb-1-x-16gb-133mhz-udimm-4-x-hot-plug-.5in-sc-sata-dynamic-smart-array-b140i-no-optical-90w-1yr-next-business-day-warranty

[8] Kingston Solid State Drives SMS200S3/120G | Servers Plus [Internet]. Serversplus.com. 2017 [cited 23 February 2017]. Available from: http://www.serversplus.com/servers/hard_drives/kingston_solid_state_drives/hdkin-sms200s312

[9] HPE ProLiant Microserver Gen8 Intel Celeron G1610T Dual-Core 2.30GHz 4 x Non Hotplug on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/1039108/hpe-proliant-microserver-gen8-intel-celeron-g1610t-dual-core-.30ghz-mb-4-x-non-hotplug-

[10] x4 K. Kingston ValueRAM 8GB 1x8GB DDR3 1600MHz ECC 240-pin DIMM Memory Module DR x4 on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/771527/kingston-valueram-8gb-1x8gb-ddr3-1600mhz-ecc-40-pin-dimm-memory-module-dr-x4

[11] HPE ProLiant ML10 Gen9 G4400 Dual Core 4GB Non hot plug Entry level tower server on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/1107184/hpe-proliant-ml10-gen9-g4400-dual-core4gb-noon-hot-plug-entry-level-tower-server-

*Cost: I have used costs for services in the EU (Frankfurt) region because of limited availability of certain instances in EU(London). Cost is also based on the hourly cost of Linux instances in the cases of Spot, On-Demand, and Reserved instances for consistency. In the case of reserved instances I have split the rate into three parts to represent the effective hourly rate with no upfront cost, partial upfront cost, or with the full payment upfront. For dedicated instances I have similarly split the cost, but have added an extra to represent the on-demand pricing as well.

*Physical Server Cost: The whole cost is largely an estimation, there was considerable difficulty when it came to finding the exact combination of cores/clock speed/memory/storage (in the case of storage I used the smallest SSD I could find which is much larger than any of the AWS options). The hourly cost estimation assumes that the server is being run for a year (total price/hours in a year) and does not include: Software purchases/subscriptions, administration and work on the server itself, part failure and replacement, power or networking costs.

Pricing[12]:

On-Demand Pricing: With on-demand pricing you pay by the hour without commitments or initial costs. This allows for maximum flexibility and scalability and is especially useful when demand for the application is unpredictable or project duration is short.

Reserved Instance Pricing: Reserved instances have a lower cost than on-demand equivalent, but require the user to commit to paying for the instance for a one to three year term in order to get that reduction. There is flexibility in the way in which you may pay, depending on whether you pay for the space up front, partially upfront with monthly payments thereafter, or with monthly payments alone.

Spot Instance Pricing: Spot instances allow you to bid on spare EC2 computing space at a reduced price (compared to on-demand). This bidding process involves requesting an instance type, and stating your maximum bid, and when the spot price is lower than your maximum bid price your request will be fulfilled. If the spot price goes higher than your bid price, your spot will be terminated.

Dedicated Host Pricing: A physical EC2 server dedicated to the user. These can either be paid for using an on-demand pricing model on an hourly basis, or for considerably less money they can be purchased under the same terms as a reserved instance - based on varying levels of upfront payments.

Part 2: SaaS

Code that calculates the response time of an API request:

[12] Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/

```
//the actual API testing function
function testAPI (url, requests, rateLimit) {
        if (requests > 1) {
                var count = new Array(requests).fill(0);
                var results = count.map(function() {
                        var startTime = Date.now();
                        var result = request.sync(url);
                        var responsetime = Date.now() - startTime;
                        //the below if statement checks if the responsetime is less
                        //than 15 minutes/rateLimit, if it is, we wait
                        if (responsetime < 900000/rateLimit) {
                                var waitTime = (900000/rateLimit)-responsetime;
                                var start = Date.now();
                                while (Date.now() != start+waitTime) {
                                }//pretty hacky, but there isn't a nice wait() function
                        }
                        //for debugging console.log(JSON.parse(result.body).forms[0]);
                        return responsetime;
                });
                return results;
        } else {
                var startTime = Date.now();
                var result = request.sync(url);
                var responsetime = Date.now() - startTime;
                return responsetime;
        }
};
```

The above code will make a GET request for data about the Pokemon Zapdos[13] to pokeapi.co[14] (or any URL specified by the url variable), a specified number of times. If you specify just one request it returns a single value representing the response time, and for multiple requests an array of response times from which we can derive the average/median/max response times. The function checks the time before the request is sent and the time after, and calculates the response time based on that (the functions used to calculate these are in the submitted file, but it didn't seem necessary to add them here. The computation time taken up by the Date.now() method may make the response time appear marginally greater than it actually is, but any difference should be mostly negligible.

This testAPI() function is called by the following function expression:

[13] Zapdos | Pokédex [Internet]. Pokemon.com. 2017 [cited 23 February 2017]. Available from: http://www.pokemon.com/uk/pokedex/zapdos

[14] Hallett P. pokéapi - The Pokemon RESTful API [Internet]. Pokeapi.co. 2017 [cited 21 February 2017]. Available from: http://pokeapi.co

```
//handles which functions get called/what the final print looks like
(function() {
        if (requests == 1) {
                console.log(testAPI(url, requests, rateLimit));
        } else {
                var results = testAPI(url, requests, rateLimit);
                console.log('Response Times: '+results.toString());
                console.log('Calculations Complete!\nThe average response time was: '+calcAverage(results)+'\nThe median response time was:
        }
}());
```

This function will either print a single value representing the response time if 'requests' is 1, or all of the requested response times, and the results from the analysis functions if 'requests' is greater than 1, producing output similar to this:

```
Response Times: 3869,1913,2057,2730,2031
Calculations Complete!
The average response time was: 2520
The median response time was: 2057
The max response times was: 3869
The standard deviation was: 732.9174578354646
```

Response times this high seem common from pokeapi.co, and other users have reported similarly lengthy response times recently thanks to an uptick of interest in Pokemon. In tests the average response has been anywhere from <1000ms, to > 5000ms.

Rate Limiting:

The pokeapi.co API just has a rate limit of 300 requests per resource per day[15].More often, however, API rate limits are based on shorter time frames. I decided to base the code on a seemingly more common requests-per-15-minutes, so for general use the code now requires a user to make a short calculation to adjust for the rate limit of their given API if it happens to not be done in fifteen-minute increments. This is not ideal for the case of pokeapi.co, but it's a better example of how to account for a more common rate limit.

```
//the below if statement checks if the responsetime is less
//than 15 minutes/rateLimit, if it is, we wait
if (responsetime < 900000/rateLimit) {
        var waitTime = (900000/rateLimit)-responsetime;
        var start = Date.now();
        while (Date.now() != start+waitTime) {
        }//pretty hacky, but there isn't a nice wait() function
```

Part 3: PaaS

[15] Hallett P. pokéapi - The Pokemon RESTful API [Internet]. Pokeapi.co. 2017 [cited 21 February 2017]. Available from: http://www.pokeapi.co/docsv2/

The code for this section is super simple, and just creates and deletes an app on OpenShift 5 times. Each time, the instantiation time is recorded, and at the end the maximum, average, and total instantiation times. Total isn't necessary, but exists as a proof that the average is correct.

```
response=0
max=0
total=0

for i in {1, 2, 3, 4, 5} ; do
        start=`date +%s`
        echo 'yes' | rhc app create anodeapp nodejs-0.6
        end=`date +%s`
        echo 'yes' | rhc app delete anodeapp
        response=$((end-start))
        total=$((total+response))
        if ((response>max)) ; then
                max=$response
        fi
done

echo 'total:'$total
average=$((total/5))
echo 'average: '$average
echo 'maximum: '$max
```

References

1. EC2 Instance Types – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/instance-types/

2. EC2 Instance Pricing – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/on-demand

3. EC2 Instance Pricing – Amazon Web Services (AWS) [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

4. Amazon EC2 Dedicated Host Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/dedicated-hosts/pricing/

5. Amazon EC2 Spot Instances Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/spot/pricing/

6. Pricing [Internet]. Amazon Web Services, Inc. 2017 [cited 21 February 2017]. Available from: https://aws.amazon.com/ec2/pricing/

7. Hallett P. pokéapi - The Pokemon RESTful API [Internet]. Pokeapi.co. 2017 [cited 21 February 2017]. Available from: http://pokeapi.co

9. Zapdos | Pokédex [Internet]. Pokemon.com. 2017 [cited 23 February 2017]. Available from: http://www.pokemon.com/uk/pokedex/zapdos

8. Hallett P. pokéapi - The Pokemon RESTful API [Internet]. Pokeapi.co. 2017 [cited 21 February 2017]. Available from: http://www.pokeapi.co/docsv2/

9. Unity - Unity - Overview [Internet]. Unity. 2017 [cited 22 February 2017]. Available from: https://unity3d.com/unity

10. Kingston Solid State Drives SMS200S3/120G | Servers Plus [Internet]. Serversplus.com. 2017 [cited 23 February 2017]. Available from: http://www.serversplus.com/servers/hard_drives/kingston_solid_state_drives/hdkin-sms200s312

11. HPE ProLiant Microserver Gen8 Intel Celeron G1610T Dual-Core 2.30GHz 4 x Non Hotplug on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/1039108/hpe-proliant-microserver-gen8-intel-celeron-g1610t-dual-core-.30ghz-mb-4-x-non-hotplug-

12. x4 K. Kingston ValueRAM 8GB 1x8GB DDR3 1600MHz ECC 240-pin DIMM Memory Module DR x4 on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/771527/kingston-valueram-8gb-1x8gb-ddr3-1600mhz-ecc-40-pin-dimm-memory-module-dr-x4

13. HPE ProLiant ML10 Gen9 G4400 Dual Core 4GB Non hot plug Entry level tower server on Servers Direct [Internet]. Serversdirect.co.uk. 2017 [cited 23 February 2017]. Available from: https://www.serversdirect.co.uk/p/1107184/hpe-proliant-ml10-gen9-g4400-dual-core4gb-noon-hot-plug-entry-level-tower-server-