Alessio Provenzano, Maria Elena SIlletti, Luca Uccellatori

Blast and Furious

Technical Design Document

Link of Fire

Università degli Studi di Milano, Anno Accademico 2015 / 2016, Corso di Online Game Design

SUMMARY

1 Project Goal

1.1 Target Service

The service we want to provide for Blast and Furious have the following characteristics:

- **Always online** (7/24);
- Playable through **Steam**, with Windows, Mac OS X and Linux players being able to play together;
- Online storage of **user profiles** with their rank, cosmetic unlocks and personalised vehicles;
- Online storage of **statistics** of every game, without details on profiles or other potentially sensible data, to be used to improve the game itself – this includes a **log** for errors and client-generated **reports**;
- **Sharing options** towards Facebook and Twitter;
- **Low latency** in order to support real-time matches;
- Provide all of the above to the expected CCUs for a game that hits up to **1 million sales**.

1.2 Target platforms

Our only target platform is **Steam**, and we'll provide a **Windows**, **Mac OS X** and a **Linux** version of the game. This decision is justified in the Game Design Document, paragraph 3.2.

1.3 Development platforms

1.3.1 Development Hardware

As this game doesn't need anything special, hardware-wise, to run, we are going to need computers that satisfy the **requirements** of our development software (detailed in paragraph 1.3.2) – and we're going to need to have installed all of the **supported OSs** (detailed in paragraph 3.3 of the Game Design Document).

In addition to the machines our staff will need to work, we're going to need a **machine** for the git / continuous integration server and a solution for on-site **backup**.

1.3.1.1 Production Machines

To make our production staff (detailed in paragraph 7.7.1) up and running we are going to need:

- **6 high end desktop computers**, with the following indicative characteristics:
    - Skylake i7 CPU;
    - 16GB RAM;
    - 1TB storage;
    - Nvidia 970 graphic card;
    - Windows 10 and Linux Ubuntu as OSs.
- **1 medium\low end desktop**, to maintain the main git repository and continuous integration server;
- **8 monitors** (the two programmers get two monitors each) with the following characteristics:
    - 27";
    - 5ms refresh rate;
    - IPS technology;
    - Full HD.
- **2 high end graphics tablet**, like the Wacom Cintiq 27QHD touch + Ergo Stand;

1.3.1.2 Backup

We're going to use **onsite** and **offsite** solutions to back up everything we produce. The onsite solution is provided by **QNAP**, with its Turbo NAS ts-453 pro and all of its 4 HDD slots occupied by 6TB WD Red Pro using RAID 6. The offsite solution is provided by **SOS Backup Solutions**, which is able to synchronise with our private NAS and offers discounts for a yearly-based subscription.

1.3.2 Development Software

To develop this project, we need a few software and, just like for the hardware, we already possess some of them and we are going to list them here. All licenses costs are detailed in chapter 7.5.

- Trello (for project management);
- 3 Pro licenses Unity;
- Git;
- 2 Adobe Photoshop licenses;
- 2 Adobe InDesign licenses;
- Blender;
- Steamworks SDK.

2 Added Value Services

We want to provide widespread support for our game, using both a **site** with a contacts page and the most used **social media**. Social media have the huge advantage of providing a platform with already registered users that are able to interact with you at any time. We need to hire someone in charge of **administrating** the site and collecting feedback through it, and we also need to hire someone in charge of **managing** our social media accounts.

We believe in the "**come for the game, stay for the people**" motto, and we know that it's important to provide a platform to make the community grow and find mates for team play. Therefore, we want to provide our site of a community made with the open source bulletin board software **phpBB**. We'll have to hire someone in charge to manage and moderate the community, getting feedback from it and helping it to grow.

To sum up, we want to provide:

- An official website, which will feature:
    - Official statements;
    - News on maintenance;
    - Complete changelogs;
    - Press kits;
    - Trailers;
    - A contact form;
    - A community made with phpBB.
- Coverage on the most famous social networks:
    - An official Facebook page;
    - An official Twitter account;
    - An official YouTube channel.

3 Market

3.1 Competing technologies

As stated in paragraph 1.2 we are going to target Steam, with all of its supported OSs (Windows, Mac OS X, Linux). The direct competitors of these technologies are the **gaming consoles** (both home and handheld) as well as the **mobile market** (Android and iOS devices).

We've ruled out the gaming consoles because they have strict rules on the network infrastructure required in order to get the game on their platform, which would greatly increase the overall cost. In addition, in order to develop our project on console we would need to buy the relative development kits and hire additional, specialised staff.

The reason we've ruled out the mobile market is different: despite the low barrier to entry, we think the mobile platform is not suitable to our type of game and its complex controls, and that the delay that the mobile connection technologies introduce would not allow for an enjoyable gaming experience.

3.2 Competing platforms

For the PC market, Steam wasn't the only platform for releasing our game, but in our opinion it is by far the most viable: not only it has the largest user base available, but it provides part of the **infrastructure** that makes online gaming possible. There are a number of alternatives to selling the game through Steam:

- Itch.io;
- GOG Store;
- Other minor online stores.

Of course it is to be considered the possibility to have an e-commerce website to self-distribute the game. The fact is that all of these options don't provide any service for the online infrastructure, and we believe that going through Steam is worth the missed revenues for the percentage it'll demand.

As for the console market, all of the consoles could be in our plans but it's a riskier market: there are some costs we can't predict precisely, the dev-kit and eventual licenses for the APIs, and other restrictions on the online infrastructure. Moreover**, Steam has a higher user base** than all of the consoles and thinner barriers. Summing it all up, developing on consoles is riskier and pricier.

4 Delivery

4.1 Estimated delivery time

We estimate to deliver the game by the **December 1st, 2017**. To do that we set up the following Project Milestones.

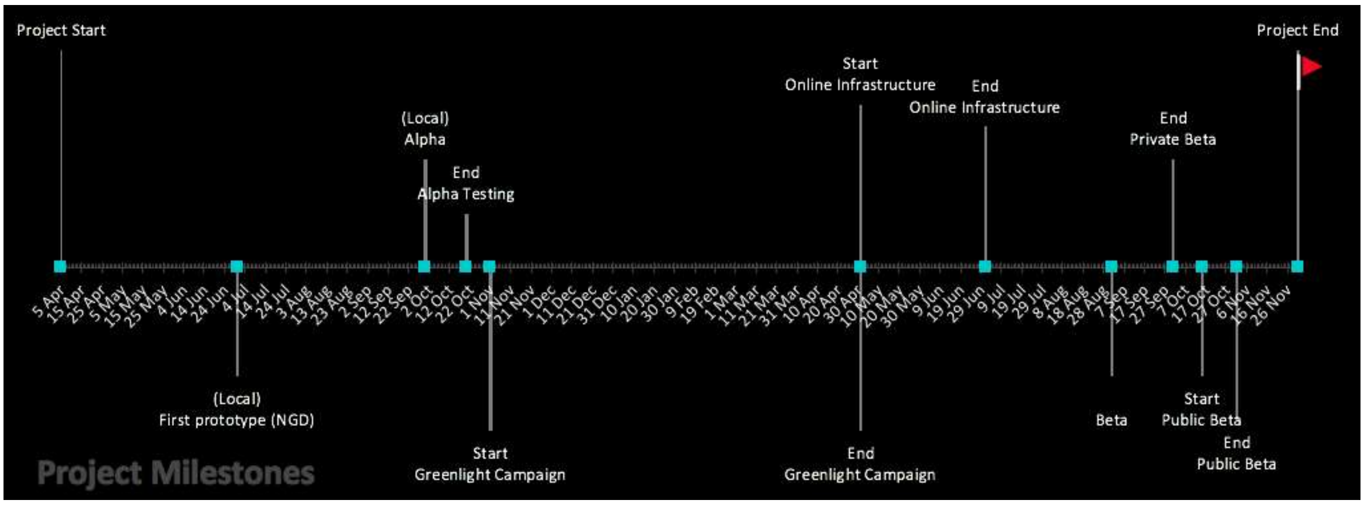


Figure 1 - Project Milestones

4.2 Delivery platform and strategy

4.2.1 Hardware and Software

As mentioned in paragraph 1.2, the delivery platform is Steam (on Windows, Mac OS X and Linux). We plan only to distribute **digital copies** of the game.

Steam offers a platform for emergent indie developers, which is **Steam Greenlight**. It is a platform where emergent developers, basically anyone who never used Steam before, can submit their game without constraints for the Community to play it and evaluate it. The game can be at any stage of development. When the **Community** approves it, with an unclear judgement meter by **Valve** which does not involve the number of "votes", Steam will contact the developer to set a release date and a price target. As of today, the projects that were approved and made it to the market through Steam Greenlight are more 4,000. To be part of this program, we'll need to pay a one-time fee of $100.00. We'll need to produce the following material:

- A square image as a **logo**;
- At least one **trailer** or **gameplay** video;
- At least 4 **screenshots** or images;
- A **description** of the game and the system requirements.

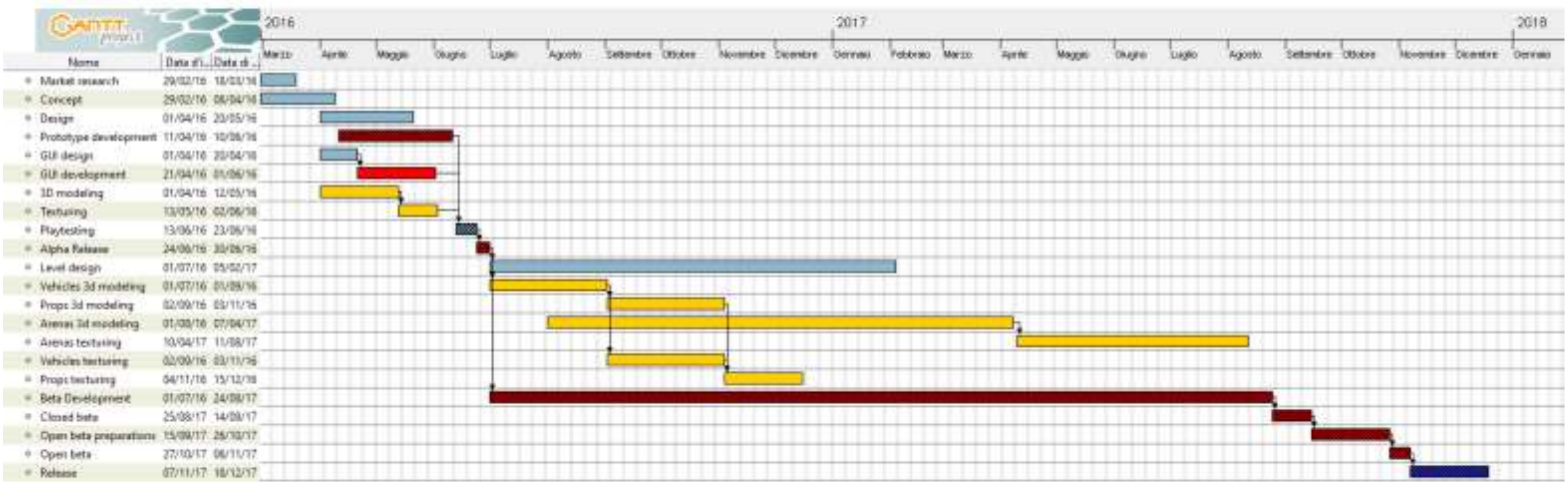
4.2.2 Impact on budget

All the four platforms demand a **30%** of every purchase, to calculate after deducting local taxes. In addition to this, developing for every platform does not come for free: Steam SDK is free of charge, once they accept you as a developer. However, we should consider that until our game is approved on Greenlight we won't earn money. It is very difficult to forecast how much time it takes for a game to be greenlit, the earliest were greenlit after 30-35 days while there is proof of games being greenlit after two years. While it is obvious we need to do our best promoting our game, the only thing we can do is absorbing part of this cost developing the game during the process of getting votes on Greenlight. Hopefully if the Community see improvements it'll be more prone to vote us.

5 Development

5.1 Major software development tasks

The following picture illustrates how the project will be developed for the intended ETA.



Nome	Data d'i...	Data di ...
Market research	29/02/16	18/03/16
Concept	29/02/16	08/04/16
Design	01/04/16	20/05/16
Prototype development	11/04/16	10/06/16
GUI design	01/04/16	20/04/16
GUI development	21/04/16	01/06/16
3D modeling	01/04/16	12/05/16
Texturing	13/05/16	02/06/16
Playtesting	13/06/16	23/06/16
Alpha Release	24/06/16	30/06/16
Level design	01/07/16	05/02/17
Vehicles 3d modeling	01/07/16	01/09/16
Props 3d modeling	02/09/16	03/11/16
Arenas 3d modeling	01/08/16	07/04/17
Arenas texturing	10/04/17	11/08/17
Vehicles texturing	02/09/16	03/11/16
Props texturing	04/11/16	15/12/16
Beta Development	01/07/16	24/08/17
Closed beta	25/08/17	14/09/17
Open beta preparations	15/09/17	26/10/17
Open beta	27/10/17	06/11/17
Release	07/11/17	18/12/17

Figure 2 - Gantt chart

Where the blackened bars are part of the critical path.

5.2 GUI specification and workflow

The User Interface has been described and wire-framed in full in the Game Design Document, paragraph 5.4.

6 Infrastructure Details

Since we have little experience with the setup and maintenance of an online game infrastructure, we are going to out-source it to a specialised third party. We chose **Amazon Web Services** as our vendor for a multitude of reasons:

- They have a long **experience** in the field, having worked for AAA companies such as Ubisoft, Naughty Dog and Mojang providing similar services;
- They offer a **360° service**: computing, file storage, block storage, backup, redundancy, security and maintenance of the infrastructure;
- **Scalability** and **Extendibility**;
- They offer plans with dynamic **bills** that depends on the usage.

We'll need to deploy our services on two regions, US and Europe, as we need very low latency. We don't currently plan to deploy our services on other regions because we don't expect much traffic coming from them, as stated on the Game Design Document in paragraphs 3.1 and 3.2.

6.1 Target sell point and Concurrent Players

As stated in paragraph 1.1, we aim to provide an infrastructure suitable for a game that sells **1 million copies**. To do that we must have at least an idea of how many concurrent **users** are going to use our game on a daily basis. We can estimate this number looking at how other multiplayer games behaved – we are going to look at the **PDCCU** (Peak Daily Concurrent Users) at day one, after a week and after three months. After that we are going to draw a few conclusions and set a few numbers to set our service dimensions on.

Game	Owners	PDCCU (Day 1)	PDCCU (Day 7)	PDCCU (Day 90)
Metal Gear Solid V	955,000	91,000 (9.52%)	71,000 (7.43%)	9,000 (0.90%)
Tom Clancy's The Division	765,000	114,000 (14.9%)	92,000 (12.02%)	11,000 (1.43%)
ARK: Survival Evolved	3,000,000	65,000 (2.16%)	80,000 (2.66%)	60,000 (2%)
Just Cause 3	548,000	23,400 (4.27%)	14,800 (2.70%)	2,900 (0.5%)
TC's Rainbow Six Siege	512,000	9,300 (1.81%)	9,600 (1.87%)	14,000 (2.73%)
Rocket League	3,276,000	27,500 (0.80%)	29,000 (0.88%)	31,000 (0.94%)
Call of Duty: Black Ops III	1,146,000	63,000 (5.49%)	47,000 (4.10%)	26,500 (2.31%)

Table 1 - Popular games and their owners - PDCUU ratio

Means	1,457,500	56,170 (3.85%)	49,000 (3.36%)	22,000 (1.50%)

Table 2 - Means of Table 1

We can note that not even on Day 1 we can expect more than 10% of users to be connected concurrently. In the table above we can see four games that peak at Day One and then slowly go down and three games that are really stable in the percentage of CCU. We can note that all of the three "stable" games have around 2% of CCU. All of the others, settle around the same percentage after three months. Considering these data, we are going to buy servers to ensure that the 2% (20,000 CCU) of our intended audience can play smoothly, managing bursts and peaks paying AWS per-hour.

6.2 How services are deployed on servers

AWS offer us the service **CodeDeploy**, which comes free with Amazon EC2 instances. CodeDeploy allows us to deploy automatically code to our infrastructure. The service scale with the infrastructure so we won't have any problem with deploying on different instances of the services we'll buy. Deployment can be stopped and rolled back at any time, and CodeDeploy offer us tools to analyse the **application's health**. It is designed to support continuous integration and has plenty documentation available to setting it up with the most used **developer tools**. In particular, we are going to need to integrate it with our **GitHub Repository**. Here's how it works:

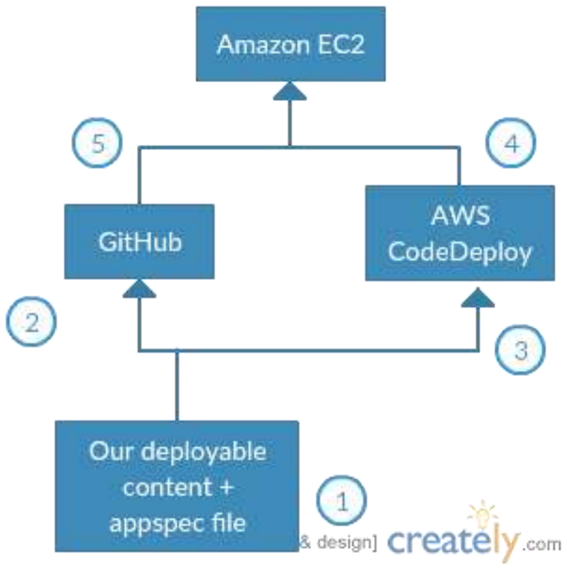


Figure 3 - CodeDeploy workflow

1. First we have to make an archive that has the deployable content with an Application Specification File, which define what actions we want to make for the deployment;
2. Then we will push our revision to our GitHub Repository.
3. Next we provide AWS CodeDeploy the info about where we want to deploy (our instances of Amazon EC2) and from where he should pull (our GitHub repository);
4. Now it is AWS CodeDeploy that will take care of the preparation, preparing the instances of AWS EC2 to the deploy;
5. Finally, Amazon EC2 will pull the revision from our GitHub Repository.

6.3 What is on each server

6.3.1 AWS Elastic Load Balancing

This service coming from Amazon takes care of the **workload**. We are going to need one for every region to better distribute the **traffic**. This choice won't affect the budget much as Amazon charges for traffic and not for the number of instances.

Amazon did not share any of the technical details of ELB. But they promise that it is able to **scale** with the needs of the infrastructure and whatever incoming traffic we'll face, even on Day One. ELB is able to determine if some of our EC2 instances are unhealthy and route the traffic to other instances.

6.3.2 AWS EC2 Dedicated Host

This kind of service allow us to create **clusters** of game servers without relying on pure cloud computing. We'll have the possibility configure our own dedicated host, still maintaining the benefit of automatic scaling of the resources. Our choice of EC2 is the compute-optimised c4.2xlarge **dedicated host**. It has the following characteristics:

Name	vCPU	RAM (GiB)	Storage	EBS Bandwidth (Mbps)
C4.2xlarge	8x Xeon E5-2666 v3 (Haswell)	15	EBS-only	1,000

Table 3 - Our AWS EC2 configuration choice

Giving our target System Requirements, detailed in paragraph 3.3 of the Game Design Document, we expect these CPUs to be able to manage **10 concurrent matches** each, at a solid **30fps**. However, this requires a stress test of confirmation.

Since AWS allows us to have clusters of 4 C4.2xlarge instances for subscription, we can calculate how many **subscriptions** of EC2 we'll need to manage our expected CCU: every vCPU manages 10 concurrent matches and every instance have 8 vCPU, that means 80 matches of 4vs4 players – 640 players. So every cluster of 4 instances is able to manage 2,560 players. That means nearly 8 subscriptions of EC2 are needed to cover 20,000 players under the assumption of 10 matches per vCPU. Since we can do this without further costs, we are going to distribute these subscriptions like this:

- 3 in the US;
- 3 in Europe;
- 1 in South America;
- 1 in Asia.

Whenever the computational power turns out to not be enough for that particular region we are going to pay on a **per hour** basis.

With this service Amazon provide a **firewall**, too, manageable through a complete API for every instance. We'll have to setup a "**Security Group**", managing rules for access control and threats individuation.

6.3.3 Storage

All of the subsequent Storage options are charged per GB provisioned per month.

6.3.3.1 Throughput Optimised HDD (st1)

These are low cost **HDD volumes** designed for frequently accessed, throughput intensive workloads. This is where we are going to store the database of our users, statistics about the matches. We didn't chose the pricier SSDs for this purpose because during the matches we won't need to access this DB very often – it is updated only once matches are over. This is difficult to estimate as it depends on the number of matches played.

6.3.3.2 EBS General Purpose SSD (gp2)

These are pricier but more performant storage options. This is where we are going to **store** the AMI for the boot of every EC2 instance as well as the game's current build. This is easier to calculate because it isn't going to change often.

6.3.3.3 Backup and logs

AWS EBS offers a "Snapshots" service which is the ability to save point-in-time snapshots to Amazon S3. These snapshots are **incremental** – we are billed only for the differences between a snapshot and its precedent. We plan to make a snapshot every day, automatically.

The same Amazon S3 service will be used for **logs**, which are generated and stored automatically by **Amazon CloudWatch**.

6.3.4 Analytics

AWS offers Cloudwatch to monitor **EC2 instances**, grouped by AMI ID with the Detailed Monitoring option, at a one-minute frequency. With Cloudwatch, monitoring data is retained for two weeks. Furthermore, Cloudwatch grant the access for the **Auto Scaling** service at no additional fee, which will make sure that resources scale up with the demands.

6.4 How servers are connected

At a very high level, we have three big entities: the **players** (clients), the **Steam infrastructure** and **our infrastructure**. It is to be acknowledged that Steam puts a firewall between their infrastructure and the players as well as their infrastructure and us.

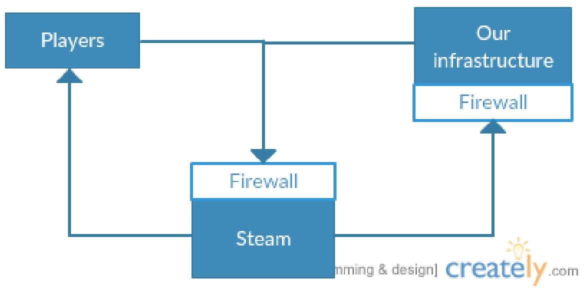


Figure 4 - Player, Steam, Infrastructure data streams

We can't really do much for the last mile – that piece of infrastructure that connects the players to the internet. We can only acknowledge that players have very **different connection speeds** and try to optimise our game to make the impact on the bandwidth as low as possible.

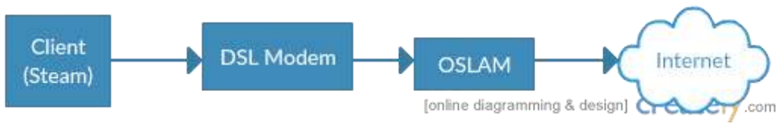


Figure 5 - How clients are tipically connected to the internet

We know little about the infrastructure of Steam, because details are shared under NDA after you are chosen as a developer, which means after having been greenlit. However, they disclosed some very high level information about their infrastructure and the services they provide to online games. Particularly, Steam has 66 data centres all over the world, managed 7*24*365, PCI compliant with an availability "much greater than 99%". From the API overview, which is visible to everyone, we can see what services they provide without much details:

- **Authentication** – users must be authenticated to access the infrastructure; Steam also makes sure that users actually own a game before accessing its infrastructure;
- **Stats & Achievements** – Steam will keep track of statistics such as how many time a user played a game and what achievements he has unlocked so far;
- **Anti-cheat system**;
- **Voice**;
- **Steam Cloud** – which is a service that allow players to keep saved data through all of their machines or OSs;
- **Community** – which is a service that allow to access in a secure way other player's information such as their displayed name;

- **Matchmaking** – Steam will take care of the matchmaking through some kind of leader board system, taking into account also information that concern their distance to the various servers, ping, etc. To make this happen, we have to make sure that our servers publish their information to the Steam Master Servers.

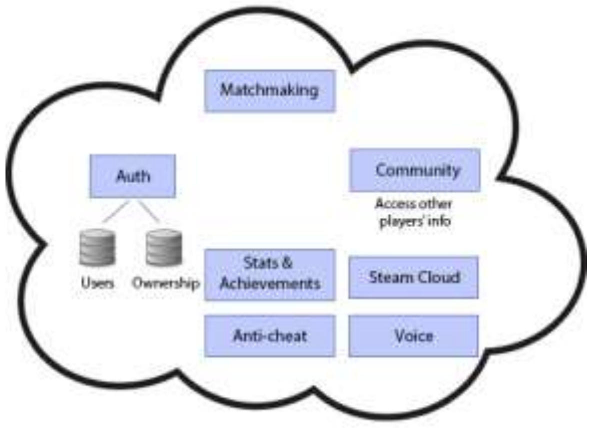


Figure 6 - Steam infrastructure's services

Now, for our infrastructure:

- The entry point is AWS ELB who take care of load balancing;
- The game servers are the AWS EC2, which run separate instances of the game – this is one cluster of four EC2s with more on-demand if needed;
- The EC2s are monitored through CloudWatch, which englobes Auto Scaling for automatic resource scaling of the EC2 instances;
- The EC2 instances are attached to two types of storage: the st1 storage takes care of the users' Database (rank, cosmetics, etc.), the server logs and the game statistics; the gp2 is for the AMI and the game's server code;
- All of the data stored in the EBS instances are snapshotted incrementally and saved in Amazon S3.

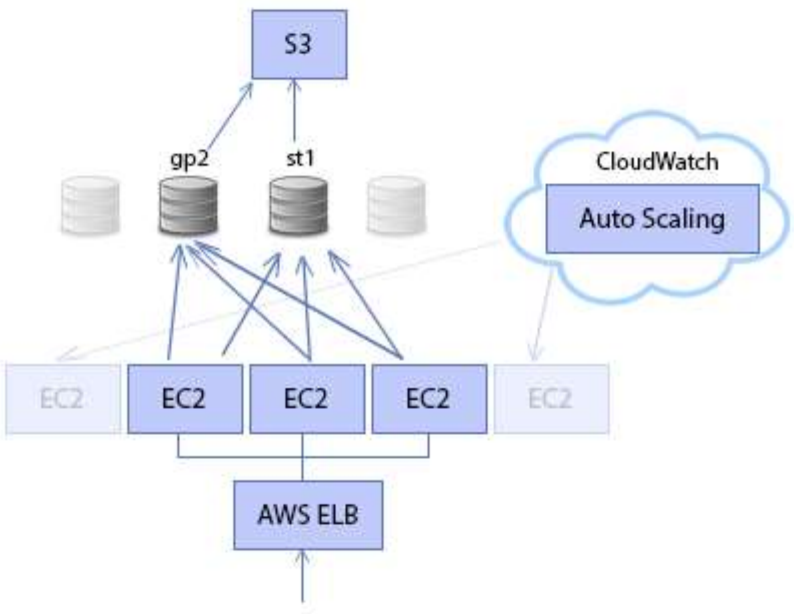


Figure 7 - Our AWS infrastructure

This architecture is replicated 8 times:

- 1x US East (N. Virginia);
- 1x US West (Oregon);
- 1x US West (Northern California);
- 2x EU (Frankfurt);
- 1x EU (Ireland);
- 1x South America (Sao Paulo);
- 1x Asia Pacific (Tokyo).

We are going to assume that each of these infrastructures will go under equal data traffic.

6.5 Capacity planning

6.5.1 Storage

As we are charged per GB and are not able to buy disks it is important to estimate how much storage we'll need. Since we use three different storage types we will also divide these raw estimates by storage type since they have a different Cost per GB.

6.5.1.1 st1

As stated in paragraph 6.3.3.1 the throughput optimised HDDs are used for the database of users, the match statistics and the logs. Since the match statistics are aggregated they won't make a great difference. However, with a million users as target goal the user database can get big quickly. We are going to dimension this as **1TB**.

6.5.1.2 gp2
This is the storage type that's most suitable to the boot and loading of interactive apps – here we are going to store the code for our server, which is basically a server version of the game itself. We are going to assume **100GB** of space will be enough for this.

6.5.1.3 Amazon S3
We are going to use this service to store incremental **snapshots** of the above storage for backup, on a daily basis, plus the server logs. Both of these things can grow very quickly and very large. Therefore, we are going to dimension this as **10TB**. Due to the purpose of this volume, we are going to use the Amazon S3 – IA (Infrequent Access) service.

6.5.2 Bandwidth
We calculated how much data our players send and inferred how much data our server should send basing on the prototype we made.

- Each player sends in 7Kb/s;
- For each match our serve sends out 230Kb/s, which are 28.75Kb/s per user connected.

For a total bandwidth of 237kb/s. These are only the current estimates and will be refined when we'll be closer to the final version of the game. We can use this information and combine it with the amount of CCU we expect to calculate how much **bandwidth** we need on a daily, monthly and yearly basis and the results can be seen in the table below. In the second row of that table we made a 50% increment to the values calculated to take into account bursts and peaks.

	DCCU	**Daily bandwidth**	**Monthly bandwidth**	**Yearly bandwidth**
Estimated bandwidth	20,000	61.776 Gb	1.853 Tb	22.239 Tb
With 50% Adjustment	30,000	92.664 Gb	2.779 Tb	33.358 Tb

Table 4 - Bandwidth estimates

6.6 Potential security issues
- **Asymmetric resource consumption**: if we need to write our own code for ranks of the game and other information about the game, we could have this kind of vulnerability: if information is saved on the server using streams, but these streams will fail to release, then attackers could force the application to use up all resources so the application may become very unstable, slow, or may stop working.
- **Repudiation Attack**: not tracking properly users' actions could expose to this kind of attack, where malicious users can change the authoring information of actions executed in order to log wrong data to log files.
- **Custom Special Character Injection**: using phpBB for the forum can lead to this kind of attack where if the input is not properly filtered, it can leak information using reserved words that are used in this proprietary application.
- **SQL Injection**: using SQL database we could lead in a SQL Injection attack if parameters of GET or POST methods are not filtered properly, causing a high risk vulnerability. The Blind SQL Injection attack is a variant of this.
- **Denial of Service**: with a large number of requests the service may cease to be available to the users. Sometimes attackers could inject and execute arbitrary code during this attack to access critical information or execute commands.

- **Session Hijacking:** using session tokens could lead in a vulnerability. In fact, with this attack the session token is compromised, by stealing or predicting a valid session token to gain unauthorized access to the Web Server. Possible scenario is one in which the session token is predictable or the session token can be sniffed.
- **Mass defacement** If the game is hosted on a server with other games, a vulnerability in one of these could lead in a mass defacement attack in which other games could be damaged.

7 Estimated Resources Needed

In this chapter we are going to estimate the resource needed to deliver the project, in terms of time, people and money. We are going to assume approximately **3 years** of support as **initial commitment**, as the plans of Amazon AWS are actually more convenient if bought for 3 years than they are if bought for only 2.

7.1 Hardware for development

With regard to the hardware equipment we listed in paragraph 1.3.1, the following table details the costs:

Machine	Number	Estimated Cost	Total Cost
High end desktop	6	\$1,500.00	\$9,000.00
Medium\low end desktop	1	\$700.00	\$700.00
Medium\high end monitor	8	\$800.00	\$6,400.00
High end graphics tablet	2	\$2,600.00	\$5,200.00

Table 5 - Hardware equipment costs

7.1.1 Backup solutions

The **onsite** solution is formed by the QNAP Turbo NAS ts-453, which costs approximatively \$500.00, with four 6TB WD Red Pro, which come at a price of \$270.00 each circa. This makes a total of \$1,580.00.

The **offsite** solution is provided by SOS Online Backup, which charge \$2,249.99 per year, which per three years makes \$6,749.97.

7.2 Hardware for servers

7.2.1 AWS EC2

In our case this is equal to calculate the cost of the EC2 instances, plus CloudWatch. Given what we said in paragraph 6.3.2, the associated cost (for three years) is the cost of a three-year reservation of a C4 type EC2 times 8:

$$\$18{,}892.00 * 8 = \$151{,}136.00$$

We have to sum at this price what it would cost to increment (hourly) our resources. We will assume the worst case we've seen in 6.1 – players will take three months to settle at a steady 2% of the owners. This means we'll need on average three times this computational power for these three months, billed hourly. The hour fee is \$1.852, so:

$$\$1.852 * 24 * 90 * 8 = \$32{,}002.56$$

And the total expense is:

$$\$151{,}136.00 + \$32{,}002.56 = \$183{,}138.56$$

7.2.2 AWS CloudWatch

CloudWatch costs \$3.50 per month per instance. We have 8 clusters of 4 instances, so that makes 32 instances. The total cost for three years is:

$$3.50 \frac{\$}{instances * month} * 32\ instances * 36\ months = \$4{,}032.00$$

7.3 Network hardware

For the network hardware we only have to consider AWS ELB which was described in paragraph 6.3.3. Considering the bandwidth estimated we made in paragraph 6.5.2 and the pricing of ELBs Amazon does (\$0.025 per ELB-hour plus \$0.008 per GB of data processed by an ELB) the cost of it in three years would be:

$$(22.239 * 1{,}000)\text{Gb} * 0.008\,\frac{\$}{Gb} + 0.025\frac{\$}{hour} * (24 * 365 * 3)hours = \$834.92$$

As we did in bandwidth planning we are going to add a 50% as a margin to manage peaks and bursts.

$$834.92\$ * 1.5 = \ 1{,}252.37\$$$

7.4 Storage

7.4.1 st1

We assumed **1TB** for this type of storage in paragraph 6.5.1.1 and they are charged on a GB provisioned per month, with a price of **$0.045**.

$$1{,}000GB * 0.045\frac{\$ * GB}{month} * 36\,months = \$360.00$$

7.4.2 gp2

We assumed **100GB** for this type of storage in paragraph 6.5.1.2 and they are charged on a GB provisioned per month, with a price of **$0.010**.

$$100GB * 0.010\frac{\$ * GB}{month} * 36\,months = \$36.00$$

7.4.3 Amazon S3

These are the volumes we are going to use to store all our backups, plus the logs. We dimensioned this as a **10TB** volume on the IAS, and it is charged per GB ($0.0125 per GB). That makes:

$$10{,}000GB * 0.0125\frac{\$ * GB}{month} * 36\,months = \$4{,}500.00$$

7.5 Software licenses

All of the software licenses here described are intended for a **three-year period**.

Software	Number of licenses	Price per license per month	Total (3-year period)
Unity	3	$75.00	$8,100.00
Github Pro	3	$9.00	$972.00
Adobe Photoshop	2	$29.99	$2,159.28
Adobe InDesign	2	$29.99	$2,159.28
Logic Pro X	2	N.A.	$399.98

Table 6 - Software licenses

Windows Server and SQL licenses are included in the **Amazon EC2** subscription – all of the software needed to run the Amazon Web Services are already included in the subscriptions we chose.

7.6 External services

Other than Amazon, we are going to use a lot of external services from Steam, which are included in the 30% cut they'll take out of every purchase. These are:

- **Payment service** – we are not going to need to worry about the payment infrastructure, the credit card's interface, PayPal's interface and so on and so forth;
- **Authentication service** – Steam is going to manage authentication and ownership;
- **Matchmaking** – Steam is going to take care about the matchmaking, grouping together players with similar skills and latency from the servers;
- **Greenlight access** – to make our game visible on Greenlight we need to pay a one-time fee of $100.

We are also going to integrate a few social integrations, allowing players to share their personalised vehicles and their match results on Facebook and on Twitter. This feature won't cost us anything, as these social platforms don't charge for this kind of integration and as the bandwidth to share content won't be on us.

7.6.1 Website and Community

For our website and the optimisation, we are going to rely on **CloudFlare web hosting**. The price for the Pro Plan is $20.00 / month. For three years, that makes $720.00. We are going to set a budget of $5,000.00 for the web agency that will design and develop both the site and the community for us.

7.6.2 QA Service

We are going to hire a company to do the QA service for us, and we are going to give it an estimated budget of $200,000.00.

7.7 Staff

7.7.1 Staff for development

The staff we are going to hire to develop and support the project through its life-cycle is detailed, along with their gross income in the table 7.

Role	Gross yearly salary	Total (3-year period)
Game Director	$75,000.00	$225,000.00
Junior Game and Level Designer	$55,000.00	$165,000.00
Senior Programmer	$100,000.00	$300,000.00
Junior Programmer	$50,000.00	$150,000.00
3D Artist	$50,000.00	$150,000.00
2D Artist	$50,000.00	$150,000.00

Table 7 - Staff salaries

We are also going to hire additional staff on a per-project basis.

Role	Per-project fee (gross)
Security Consultant	$30,000.00
Concept Artist	$20,000.00
OST e SFX composers	$35,000.00

Table 8 - Per project staff

7.7.2 Staff for production stage (people and time)

For the production stage we are going to need:

- 3x FTE **Maintenance Programmer** that ensure availability 24/7 for bug fixing and disaster recovery, along with the already hired programmers from table 7;
- 5x FTE **Social Media Managers** to cover all of our social accounts 24/7;
- 1x FTE **Webmaster** that manages our website;
- 1x FTE **Community Manager** that will have to keep our community engaged, not only on our forum but also looking at the Steam Community forums.

Role	Number	Gross yearly salary per person	Total (3-year period)
Maintenance Programmer	3	$50,000.00	$450,000.00
Social Media Manager	5	$40,000.00	$600,000.00
Webmaster	1	$30,000.00	$90,000.00
Community Manager	1	$30,000.00	$90,000.00

Table 9 - Additional staff budget

7.7.3 Office

We are going to need an **office** to maximise the productivity in **Milan**, where all the team can meet and work **together**. This is ideally for the whole team, and that makes **6 people** but we have to consider room for people hired per-project and in the production phase. Looking at the prices online, we can be fairly sure to find a suitable office with a budget of $2000.00 / month, everything included. For three years, that makes $72,000.00.

7.8 Estimated total cost

In this paragraph we are going to sum up all of the costs estimated above. We are going to assume the worst case scenario for every resource needed.

Resource	Ref. paragraph	Estimated cost
Hardware for development	7.1	$28,049,00
Hardware for server	7.2	$187,170.56
Network hardware	7.3	$1,252.37
Storage	7.4	$4,896.00
Software licenses	7.5	$13,790.54
External services	7.6	$205,820.00
Staff	7.7	$2,527,000.00

Table 10 - Sum of every cost

Total	$2,967,979.44

Table 11 - Grand total

7.8.1 Break-even point

That price tag is the one that makes our revenues equal to the total calculated on paragraph 7.8 after a million sales, minus VAT and the 30% cut Steam retains of itself.

$$1{,}000{,}000 * \left(\frac{2 * x * (1 - 0.22)}{3}\right) = 2{,}967{,}979.44$$

$$\boldsymbol{x = \$5.71}$$

We can now calculate how many copies we need to sell to break even for a number of price tags.

Price	Copies
$10,00	570,766
$15,00	380,511
$20,00	285,383
$30,00	190,256

Table 12 – Break-even point, in terms of sales, for every price tag