

**A PRELIMINARY REPORT ON
MULTIPLAYER DOT CHASE GAME**

**SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE**

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

STUDENT NAME	Exam Seat No.
Shishir Kumar	71714762C
Swapnil Ahire	71714248F
Saurab Godse	71714730E
Prathamesh Kulkarni	71714508F
Parag Kaldate	71714444F



DEPARTMENT OF COMPUTER ENGINEERING

**BRACT'S
VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY**

**SURVEY NO. 3/4, KONDHWA (BUDRUK), PUNE – 411048, MAHARASHTRA
(INDIA).**

SAVITRIBAI PHULE PUNE UNIVERSITY
2020 -2021



CERTIFICATE

This is to certify that the project report entitles

MULTIPLAYER DOT CHASE GAME

Submitted by

STUDENT NAME	Exam Seat No. :
Shishir Kumar	71714762C
Swapnil Ahire	71714248F
Saurab Godse	71714730E
Prathamesh Kulkarni	71714508F
Parag Kaldate	71714444F

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. Vidya Gaikwad** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

(Prof. Vidya Gaikwad)
Guide
Department of Computer Engineering

(Dr. S. R. Sakhare)
Head,
Department of Computer Engineering

(Dr. V. S.Deshpande)
Director
BRAC's Vishwakarma Institute of Information Technology, Pune-48

Place : Pune
Date :

ACKNOWLEDGMENT

It is great pleasure for me to undertake this project, I feel highly doing the project entitled - **“Multiplayer DOT Chase Game based on AWS”**.

I am grateful to my project guide **Mrs. Vidya Gaikwad** Faculty of the Computer Engineering Department.

This project would not have completed without their enormous help and worthy experience. Whenever I was in need, they were behind me.

Although, this report has been prepared with the utmost care and deep routed interest. Even then I accept respondent and imperfection.

Shishir Kumar – 423069 (U1610142)
Swapnil Ahire – 423003 (U1610315)
Saurab Godse – 423065 (U1610327)
Prathamesh K. - 423035 (U1610111)
Parag Kaldate – 423027 (U1610415)

B.E COMP (SEM VIII)
VIIT, PUNE

DECLARATION

I hereby declare that the project work entitled “**Multiplayer DOT Chase Game based on AWS**” Submitted to Vishwakarma Institute Of Information Technology, Pune is a record of an original work done by me under the guidance of **Mrs. Vidya Gaikwad**, Assistant Professor of Computer Engineering Department, Vishwakarma Institute Of Information Technology and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor Of Engineering in Computer Science. The results embodied in this thesis have not been submitted to any university or institute for the award of any degree or diploma.

ABSTRACT

As a genre, multiplayer games are split between two key elements — competitive and cooperative. Multiplayer games bring together real life people into virtual contact. For better or for worse, the competition is heightened. That guy next to you isn't just some NPC, it's your real life best friend playing alongside you. Stakes are higher, losses are more bitter, victories harder won. Multiplayer games have helped gaming go mainstream. Any game that allows more than one player is considered multiplayer. A multiplayer game is most often played over the Internet, however, could be also a game played over a LAN (local area network) or dial-up connection. Due to above mentioned points we developed a multiplayer DOT Game over AWS.

Table of Contents

S. No.	Title of chapter
1	Introduction
1.1	Purpose
1.2	Motivation
1.3	Problem Definition
1.4	Scope
1.5	Definitions, acronyms, abbreviations
1.6	Rules
1.7	References
1.8	Overview
2.	The overall description
2.1	Product perspective
2.1.1	System interfaces
2.1.2	Software interfaces
2.1.3	Hardware Interfaces
2.1.4	Communication interfaces
2.2	Product functions
2.3	User characteristics
2.4	Assumptions and dependencies
3	Specific requirements
3.1	Performance requirements
4	Implementation
5	GUI of application
6	Conclusion
6.1	Conclusion
6.2	Future works
6.3	Application
7.	Code Snippet

1. INTRODUCTION

1.1 PURPOSE

A **multiplayer game** is a game in which more than one person can play in the same game environment at the same time, either locally or online over the internet. Multiplayer games usually require players to share the resources of a single game system or use networking technology to play together over a greater distance; players may compete against one or more human contestants, work cooperatively with a human partner to achieve a common goal, supervise other players' activity, co-op. Multiplayer games allow players interaction with other individuals in partnership, competition or rivalry, providing them with social communication absent from single-player games.

Games are sometimes played purely for entertainment, sometimes for achievement or reward as well. They can be played alone, in teams, or online; by amateurs or by professionals. The players may have an audience of non-players, such as when people are entertained by watching a chess championship. On the other hand, players in a game may constitute their own audience as they take their turn to play. Often, part of the entertainment for children playing a game is deciding who is part of their audience and who is a player.

1.2 MOTIVATION

Create a new game, with our own defined rules. Which is fun to play and can be played with multiple friends over LAN or Online. Also easily accessible and require no additional hardware and software. Ready to play with what one has.

1.3 PROBLEM DEFINITION

Make use of Cloud Services and make a web based application, accessible through Internet and through web browser. Create something new and fun and while doing so, learn to utilize the cloud platform. Make platform accessible by multiple users at same time.

1.4 SCOPE

The scope of our project is giving multiplayers a platform to compete, play and make best use of their recreational time.

1.5 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Dot: The colourful 2D circle players can control is termed DOT in our Multiplayer Dot Game.

Player Number: The randomly generated number by server above the DOT is Player Number. It is uniquely assigned to each player to identify their targets.

Targets: DOTs having Player Number greater than your DOT is your Target.

Score: Number indicated below dot is score obtained by certain player. Whoever have greatest Score Wins.

Player Name: Your name is visible at right side of DOT, which is often referred as Player Name.

Boundary: The Rectangle visible on screen in which all DOTs are visible is termed to be boundary.

Point: Player gets a point every time they touch their targets. More time they hover over it, more will be the point awarded.

1.6 RULES

1. Players are advised to stay inside the boundary.
2. Players get point only when they touch a valid target.
3. Player with highest score win.
4. Winners are updated at real time.
5. Game is restarted every quarter of an hour (i.e. every 15 mins).
6. Movement keys are keyboard Arrow keys.

1.7 REFERENCES

1. How To Build A Multiplayer Browser Game (Part 1)- Elvin Lin

<https://hackernoon.com/how-to-build-a-multiplayer-browser-game-4a793818c29b>

2. NPM - <https://www.npmjs.com/>

3. HTML - <https://www.w3schools.com/html/>

1.8 OVERVIEW

As the World Wide Web developed and browsers became more

sophisticated, people started creating browser games that used a web browser as a client. Simple single player games were made that could be played using a web browser via HTML and HTML scripting technologies (most commonly JavaScript, ASP, PHP and MySQL). The development of web-based graphics technologies such as Flash and Java allowed browser games to become more complex. These games, also known by their related technology as "Flash games" or "Java games", became increasingly popular. Browser-based pet games are popular among the younger generation of online gamers. These games range from gigantic games with millions of users, such as *Neopets*, to smaller and more community-based pet games. More recent browser-based games use web technologies like Ajax to make more complicated multiplayer interactions possible and WebGL to generate hardware-accelerated 3D graphics without the need for plugins.

We've created DOT game with help of such browser technologies and other backend technologies to create a real-time game. That is good enough to have fun and play on multi-player environment with friends using LAN.

2. THE OVERALL DESCRIPTION

The game is such developed that anyone can join game using just a browser and is connected to same LAN or internet. On joining every player is added into a multiplayer environment. Where players are added in real-time. Rules are such devised to change the gaming situation time to time. Each player has unique Id and various roles to play when in play. Rules are mentioned in the sections above. Game is easy to understand and can be played using minimal arrow keys. And scoring system is also kept simple to understand. This project aims to develop a prototype for Cloud base.

2.1 PRODUCT PERSPECTIVE

2.1.1 SYSTEM INTERFACES

Node JS is an open-source, cross-platform, JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,^[6] unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.

Though `.js` is the standard filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.^[4]

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

npm (originally short for **Node Package Manager**)^[3] is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript

runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

Amazon Web Services

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard-disk/SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

The AWS technology is implemented at server farms throughout the world, and maintained by the Amazon subsidiary. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), the hardware/OS/software/networking features chosen by the subscriber, required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either. As part of the subscription agreement,^[5] Amazon provides security for subscribers' system. AWS operates from many global geographical regions including 6 in North America.^[6]

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (EC2) forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.^[1]

In November 2010, Amazon switched its own retail website to use EC2 and AWS.^[2]

2.1.2 SOFTWARE INTERFACES

Any browser. Chrome is Recommended though.

2.1.3 HARDWARE INTERFACES

Any device (mobile -if it has external keyboard, laptop, PC) is required.

2.1.4 COMMUNICATION INTERFACES

A **website** is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server. Notable examples are wikipedia.org, google.com, and amazon.com. Today roughly 380 new websites are created every minute across the World.

A website may be accessible via a public Internet Protocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing a uniform resource locator (URL) that identifies the site.

Websites can have many functions and can be used in various fashions; a website can be a personal website, a corporate website for a company, a government website, an organization website, etc. Websites are typically dedicated to a particular topic or purpose, ranging from entertainment and social networking to providing news and education. All publicly accessible websites collectively constitute the World Wide Web, while private websites, such as a company's website for its employees, are typically a part of an intranet.

Web pages, which are the building blocks of websites, are documents, typically composed in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML, XHTML). They may incorporate elements from other websites with suitable markup anchors. Web pages are accessed and transported with the Hypertext Transfer Protocol (HTTP), which may optionally employ encryption (HTTP Secure, HTTPS) to provide security and privacy for the user. The user's application, often a web browser, renders the page content according to its HTML markup instructions onto a display terminal.

2.2 PRODUCT FUNCTIONS

NodeJS is used as a back end and a server for the multiplayer DOT game. It is used to render canvas again and again for each change. Score of all players is stored in Node Server. Same is sent to all player clients. Use of Server sockets and Client Sockets is used to achieve the same. Various emitter and listeners are defined in Node server code to handle the moves made by players in the game.

2.3 USER CHARACTERISTICS

User should adhere to the given boundary.

2.4 ASSUMPTIONS AND DEPENDENCIES

Node Dependencies:

1. Express ^4
2. Node-cron ^2
3. Socket.io ^1.7

Once server is started it listens on port 5000 of the server pc. So for LAN player needs the IP of server PC and needs to be on same network as the server PC. In case of AWS Public dns is required and game runs on port 5000.

3. SPECIFIC REQUIREMENTS

3.1 PERFORMANCE REQUIREMENTS

Transparency :

1. Access Transparent: All other players' data is shared to all players and how this is achieved is hidden from users. Other user's real time data is shared with every player of game using only joining link and player doesn't have any knowledge about how that data is stored over various nodes of network. So, our system is access transparent.
2. Location Transparent: Player doesn't have knowledge about which server is serving him at a given time and at which server or network node his files/data is stored. So, our system is location transparent.
3. Replication Transparency: Data of each player is replicated on every network node. User doesn't know how many copies of his data there is.
4. Concurrency Transparency: The shared data i.e. location coordinates of all player are accessed by all players at same time. They can do so and this doesn't lead to any denial of service to other user. Thus concurrent system.
5. Failure Transparency: Failure of one or more server node doesn't lead to gaming network failure. Other servers serve the players without giving any hint of change to the players.

Openness

1. Portability: Any hardware having nodeJS & npm installed can be used as server. Server doesn't perform differently on different OS given they have same version of nodeJS. Hence portable.
2. Extensibility: NodeJS supports lots of packages hence various new services can be easily implemented without changing any above mentioned properties.

Scalability:

The user load can be distributed among various servers on server machine. Servers can be easily added to our system. When there is increase in number of users on a particular server or crashes other server comes into play.

Reliability:

Compared to a single system, a distributed system is highly capable of being secure, consistent and have a high capability of masking errors.

Performance: Compared to other models, distributed models give a much-wanted boost to performance.

Challenges :

Security is a big challenge in a distributed environment, especially when using public networks.

- Fault tolerance could be tough when the distributed model is built based on unreliable components.


4. IMPLEMENTATION:

1. Create an AWS account and log in to AWS Console.
2. Go into EC2 console and create a free-tier instance of ubuntu-18.
3. Using preferred option connect to this instance using pem keys.
4. Clone git repository of your project to root folder and go into it.
5. Install NPM and nodeJS.
6. Now enter commands- npm install && npm start
7. Your server is now running on localhost:5000
8. In security groups of your EC2 instance, add TCP rules to all , and add HTTP and HTTPS inbound rules.
9. copy your public DNS/IP and share with your friends.
10. open link to port 5000 in respective browsers and enter details and join game.
11. Read all rules and start competing.

awseducate.com/student/s/awssite

awseducate

Portfolio Career Pathways Badges Jobs AWS Account Logout



AWS Educate Starter Account

Your cloud journey has only just begun. Use your AWS Educate Starter Account to access the AWS Console and resources, and start building in the cloud!

[AWS Educate Starter Account](#)

Your account has an estimated **74 credits** remaining and access will end on **Apr 8, 2021**.

Note: Clicking this button will take you to a third party site managed by Vocareum, Inc. ("Third Party Servicer"). In addition to the AWS Educate terms of service, your use of the AWS Educate Starter Account is governed by the Third Party Servicer's terms, including its Privacy Policy. AWS assumes no responsibility or liability and makes no representations or warranties regarding services provided by a Third Party Servicer.

labs.vocareum.com/main/main.php?m=editor&nav=1&asrid=14334&stepid=14335

vocareum

My Classes Help prathamesh1.kulkar...

Welcome to your AWS Educate Account

AWS Educate provides you with access to a wide variety of AWS Services for you to get your hands on and build on AWS! To get started, click on the AWS Console button to log in to your AWS console.

Please read the FAQ below to help you get started on your Starter Account.

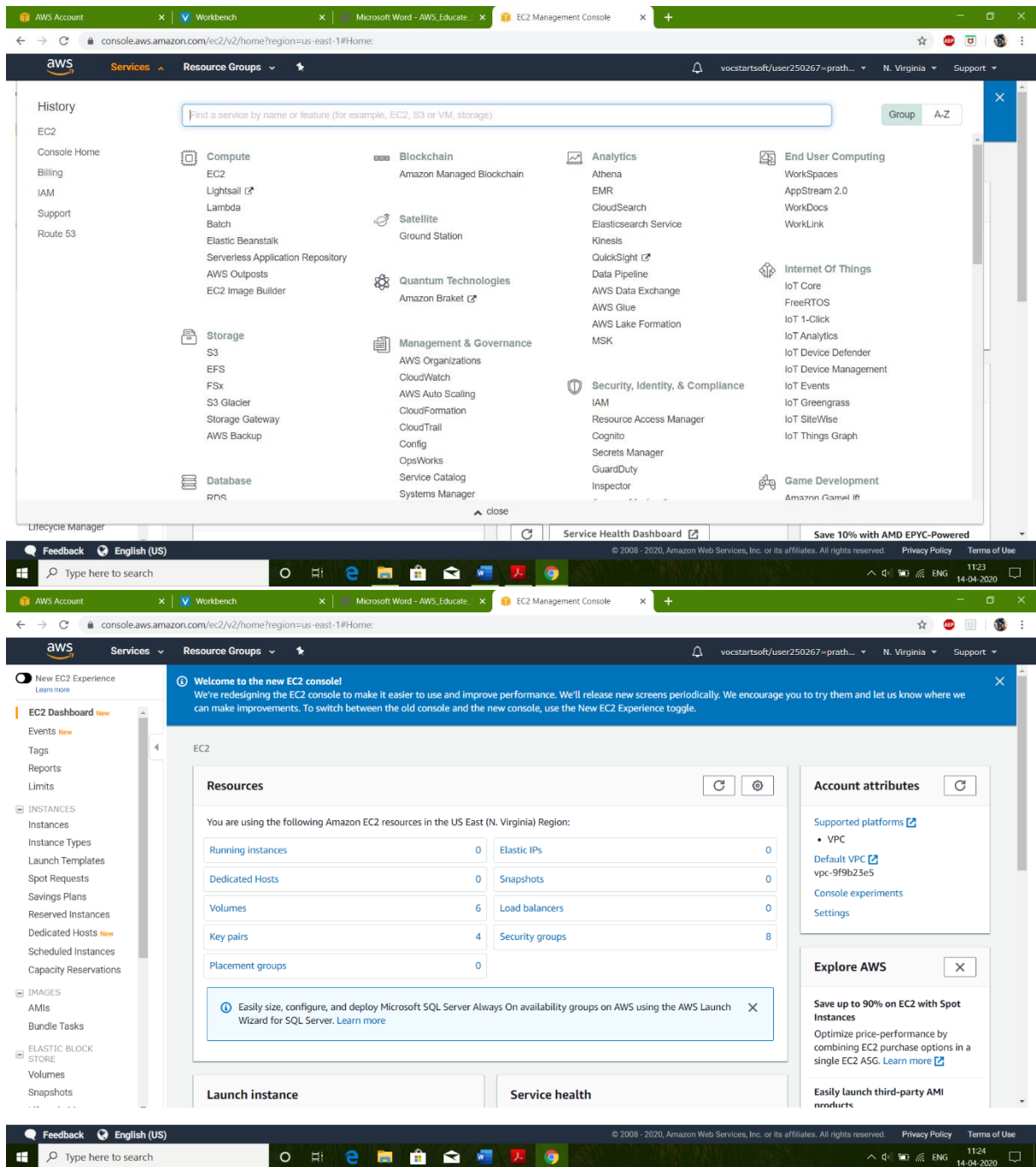
- [What are the list of services supported?](#)
Please check the current list of supported services by clicking [here](#). AWS Educate Classroom Accounts may have slightly different permissions.
- [What regions are supported with Starter Accounts or Classroom Accounts?](#)
Only us-east-1 is currently supported.
- [I can't start any resources. What happened?](#)
Please check in top right of you AWS Console view to confirm the region is N. Virginia. Only us-east-1 (Northern Virginia) is supported by AWS Educate Accounts Change your region to us-east-1 to start your resources. Alternatively please check allowed services above. Only services listed above can be used in AWS Educate Accounts.
- [Can I create users within my Starter or Classroom Account for others to access?](#)
You may create users and groups within IAM consoles. However, you cannot attach a login profile or associate keys with the users you create. This means that additional users you create cannot log into your account. If you need to use your credentials, please click the "Account Details" on the right.
- [Can I create my own IAM policy within Starter Account or Classroom?](#)

Your AWS Account Status

Active	full access (prathamesh1.kulkarni@vilit.ac.in)
\$74.38	remaining credits (estimated)
2:58	session time

[Account Details](#) [AWS Console](#)

Please use AWS Educate Account responsibly. Remember to shut down your instances when not in use to make the best use of your credits. And, don't forget to logout once you are done with your work!



AWS Account | Workbench | Microsoft Word - AWS Educate... | Instances | EC2 Management Co... | (1) WhatsApp

console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=securityGroupNames

Services | Resource Groups

New EC2 Experience

Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
	i-0101857dd8d8dc0f	t2.micro	us-east-1b	stopped		None		-	-
first_pratham...	i-023bea3401017e3ae	t2.micro	us-east-1b	stopped		None		-	-
	i-01b67f9e80c556735	t2.micro	us-east-1a	stopped		None		-	-
	i-0f6ebdea7af81d5b8	t2.micro	us-east-1d	stopped		None		-	-
	i-07eb22a69c6a2b8f	t2.micro	us-east-1b	stopped		None		-	-
	i-0481c1506a1d85949	t2.micro	us-east-1b	stopped		None		-	-

Instance: i-023bea3401017e3ae (first_pratham_cc) Private IP: 172.31.34.132

Description | Status Checks | Monitoring | Tags

Instance ID: i-023bea3401017e3ae
Instance state: stopped
Instance type: t2.micro
Finding: You may not have permission to access AWS Compute Optimizer.
Private DNS: ip-172-31-34-132.ec2.internal
Private IP: 172.31.34.132
Secondary private IPs: -
VPC ID: vpc-9f9b23e5
Subnet ID: subnet-47a78a2a
Public DNS (IPv4): -
IPv4 Public IP: -
IPv6 IPs: -
Elastic IPs: -
Availability zone: us-east-1b
Security groups: launch-wizard-2, view inbound rules, view outbound rules
Scheduled events: -
AMI ID: ubuntu/images/hvm-sd/ubuntu-bionic-18.04-amd64-server-20200112 (ami-07ebfd5b3428b6f4d)
Platform details: -

Connect
Get Windows Password
Create Template From Instance
Launch More Like This
Instance State: Start, Stop, Stop - Hibernate, Reboot, Terminate
Instance Settings
Image
Networking
CloudWatch Monitoring

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

11:27 14-04-2020

AWS Account | Workbench | Microsoft Word - AWS Educate... | Instances | EC2 Management Console

console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=securityGroupNames

Services | Resource Groups | Launch Instance

EC2 Dashboard | Events | Tags | Reports | Limits

INSTANCES | Instance Types | Launch Templates | Spot Requests | Savings Plans | Reserved Instances | Dedicated Hosts | Scheduled Instances | Capacity Reservations

IMAGES | AMIs | Bundle Tasks

ELASTIC BLOCK STORE | Volumes | Snapshots | Lifecycle Manager

Feedback | English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy | Terms of Use

11:28 14-04-2020

Connect to your instance

Connection method

- ☒ A standalone SSH client
- ☐ Session Manager
- ☐ EC2 Instance Connect (browser-based SSH connection)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (first_pratham.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:
`chmod 400 first_pratham.pem`
4. Connect to your instance using its Public DNS:
`ec2-34-227-83-197.compute-1.amazonaws.com`

Example:

```
ssh -i "first_pratham.pem" ubuntu@ec2-34-227-83-197.compute-1.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

Select ubuntu@ip-172-31-34-132 ~ /first

Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\godse\Desktop\BE PROJECT PERSONALITY>aws-keypair ssh -i "first_pratham.pem" ubuntu@ec2-34-227-83-197.compute-1.amazonaws.com

The authenticity of host 'ec2-34-227-83-197.compute-1.amazonaws.com (34.227.83.197)' can't be established.
ECDSA key fingerprint is SHA256:JASvH60f8svz11KG119v9Ase2zn+8fg0kaT1K7ko.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-227-83-197.compute-1.amazonaws.com,34.227.83.197' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)

* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/advantage>

System information as of Tue Apr 14 06:00:11 UTC 2020

System load: 0.09 Processes: 92
Usage of /: 23.3% of 7.69GB Users logged in: 0
Memory usage: 15% IP address for eth0: 172.31.34.132
Swap usage: 0%

* Kubernetes 1.18 GA is now available! See <https://microk8s.io> for docs or
install it with:
`sudo snap install microk8s --channel=1.18 --classic`

* Multipass 1.1 adds proxy support for developers behind enterprise
firewalls. Rapid prototyping for cloud operations just got easier.
<https://multipass.run/>

34 packages can be updated.
0 updates are security updates.

Last login: Fri Apr 10 06:43:09 2020 from 157.33.27.230
ubuntu@ip-172-31-34-132:~\$ ls
first
ubuntu@ip-172-31-34-132:~\$ cd first
ubuntu@ip-172-31-34-132:~/first\$ ls
README.md index.html intro.html node_modules npm-debug.log package-lock.json package.json server.js static
ubuntu@ip-172-31-34-132:~/first\$ npm start
> dot_mult_game@1.0.0 start /home/ubuntu/first
> node server.js
Starting server on port 5000

Type here to search

11:30 14-04-2020

AWS Account | Workbench | Microsoft Word - AWS Educate... | Instances | EC2 Management Co... | +

console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=securityGroupNames

aws Services Resource Groups

New EC2 Experience

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
	i-0101857dd88dfdc0f	t2.micro	us-east-1b	stopped		None			
first_pratham...	i-023bea3401017e3ae	t2.micro	us-east-1b	running	Initializing	None	ec2-34-227-83-197.co...	34.227.83.197	
	i-01b67f9e0fc556735	t2.micro	us-east-1a	stopped		None			
	i-0f6ebdea7af81d5b8	t2.micro	us-east-1d	stopped		None			
	i-07eb22a69c6a2b8f	t2.micro	us-east-1b	stopped		None			
	i-0481c1506a1d85949	t2.micro	us-east-1b	stopped		None			

Instance: i-023bea3401017e3ae (first_pratham_cc) Public DNS: ec2-34-227-83-197.compute-1.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID: i-023bea3401017e3ae
Instance state: running
Instance type: t2.micro
Finding: You may not have permission to access AWS Compute Optimizer.
Private DNS: ip-172-31-34-132.ec2.internal
Private IPs: 172.31.34.132
Secondary private IPs:
VPC ID: vpc-9f9b23e5
Subnet ID: subnet-47a76a3a

Public DNS (IPv4): ec2-34-227-83-197.compute-1.amazonaws.com
IPv4 Public IP: 34.227.83.197
IPv6 IPs:
Elastic IPs:
Availability zone: us-east-1b
Security groups: launch-wizard-2, view inbound rules, view outbound rules
Scheduled events: No scheduled events
AMI ID: ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200112 (ami-07ebf5b3428b6f4d)
Platform details:

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Type here to search

Not secure | ec2-34-227-83-197.compute-1.amazonaws.com:5000

DOT Intro

This is your DOT

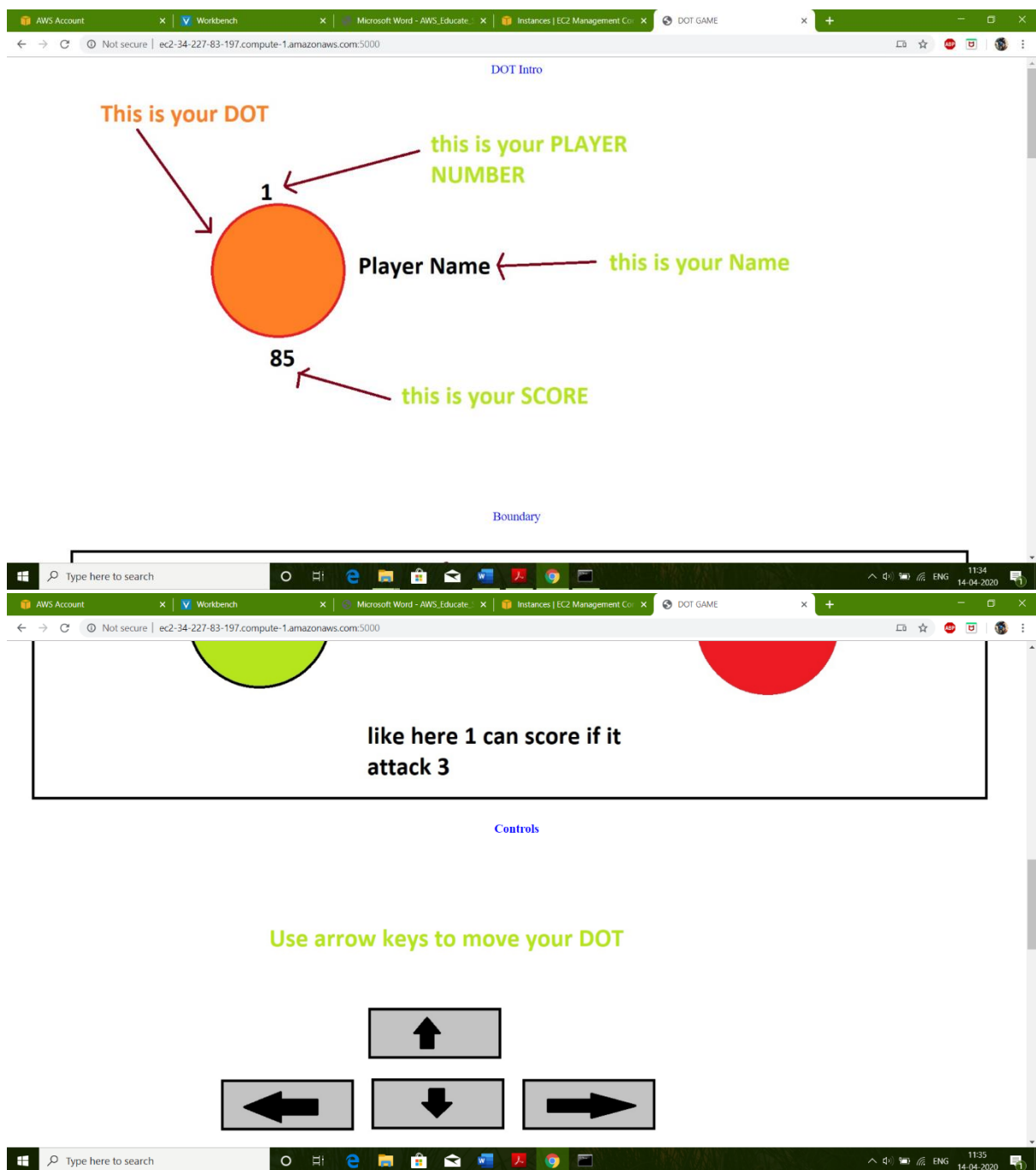
1 this is your PLAYER NUMBER

Player Name this is your Name

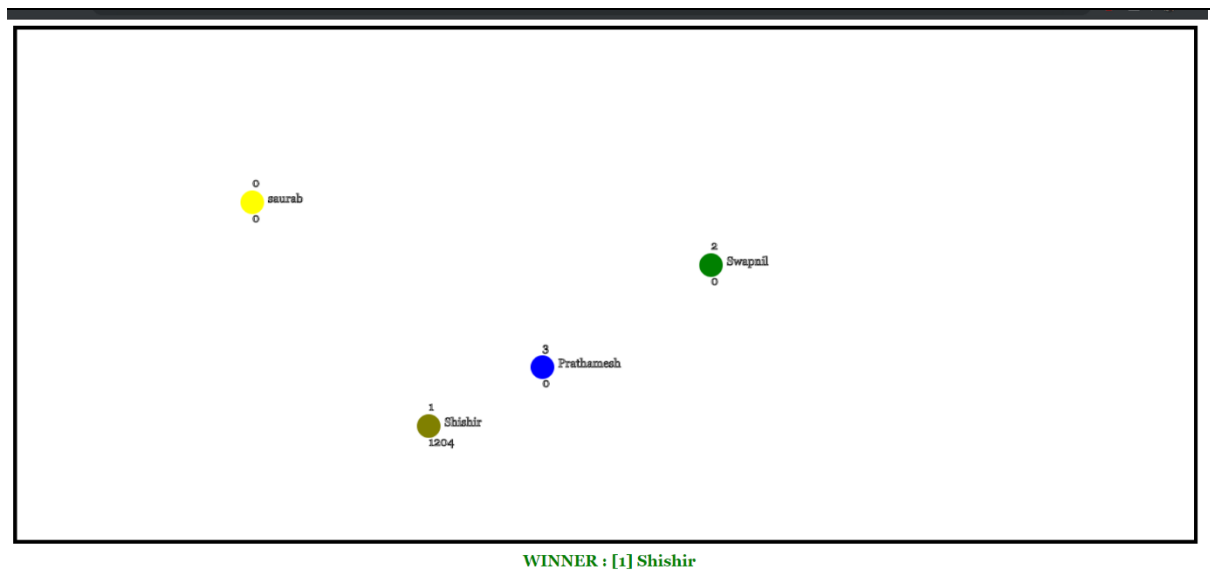
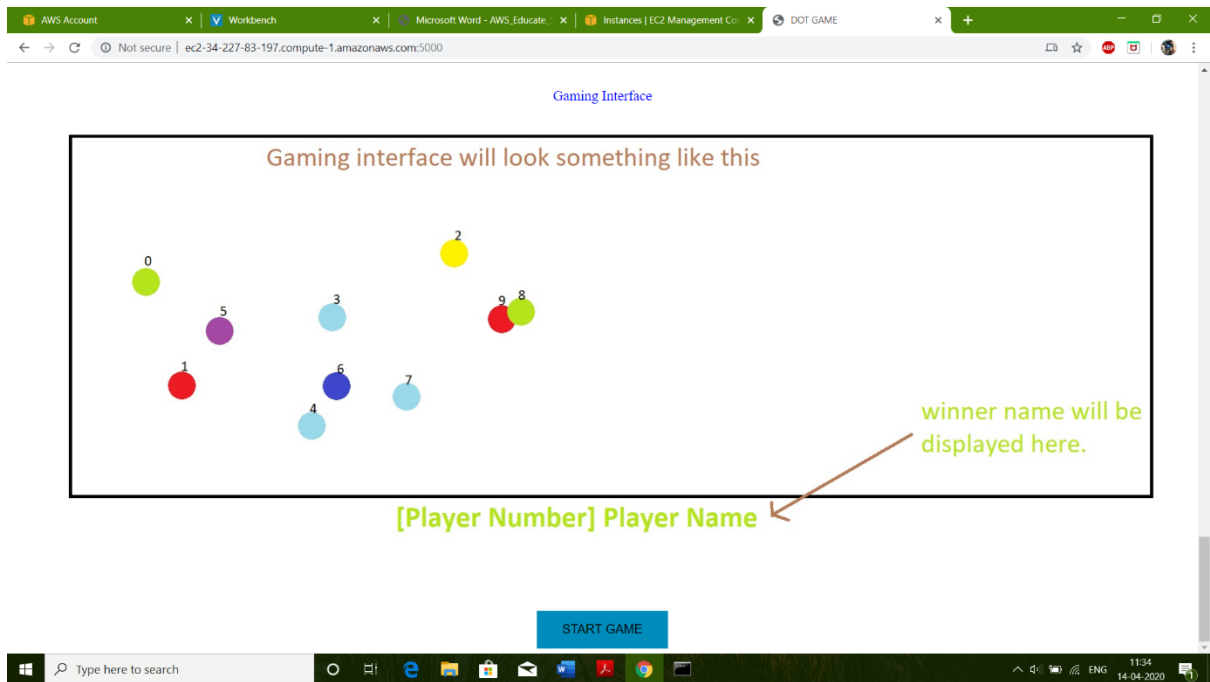
85 this is your SCORE

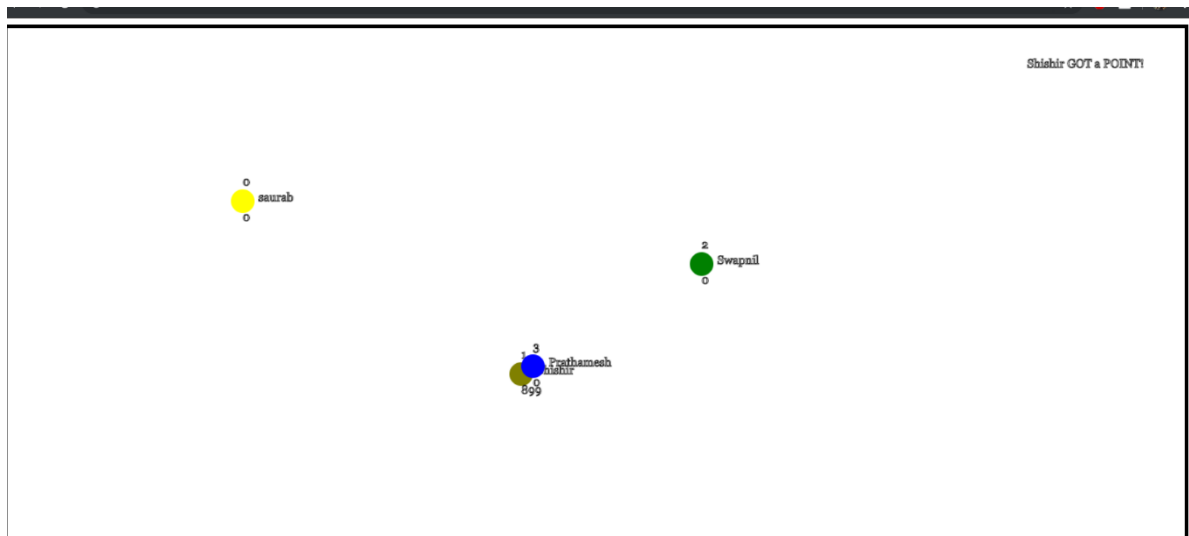
Boundary

5. GUI of Application (DOT Game):



Players' Console





WINNER : [1] Shishir

SERVER

EC2 Dashboard

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
first_pratham...	i-023bea3401017e3ae	t2.micro	us-east-1b	running	Initializing	None	ec2-34-227-83-197.co...	34.227.83.197	-
	i-0101857dd8d8dc0f	t2.micro	us-east-1b	stopped		None		-	-
	i-01b67f9e0fc556735	t2.micro	us-east-1a	stopped		None		-	-
	i-0f6bdea7a81d5b8	t2.micro	us-east-1d	stopped		None		-	-
	i-07eb22a69c6af2b8f	t2.micro	us-east-1b	stopped		None		-	-
	i-0481c1506a1d85949	t2.micro	us-east-1b	stopped		None		-	-

Instance: i-023bea3401017e3ae (first_pratham_cc) Public DNS: ec2-34-227-83-197.compute-1.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID	i-023bea3401017e3ae	Public DNS (IPv4)	ec2-34-227-83-197.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	34.227.83.197
Instance type	t2.micro	IPv6 IPs	-
Finding	You may not have permission to access AWS Compute Optimizer.	Elastic IPs	-
Private DNS	ip-172-31-34-132.ec2.internal	Availability zone	us-east-1b
Private IPs	172.31.34.132	Security groups	launch-wizard-2. view inbound rules. view outbound rules
Secondary private IPs	-	Scheduled events	No scheduled events
VPC ID	vpc-9f9b23e5	AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200112 (ami-07ebfd5b3428b6f4d)
Subnet ID	subnet-47a78a2a	Platform details	-

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

```
Select ubuntu@ip-172-31-34-132: ~/first
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\gnase\Desktop>BE PROJECT PERSONALITY\aws_keypair ssh -i "first.pem" ubuntu@ec2-34-227-83-197.compute-1.amazonaws.com
The authenticity of host 'ec2-34-227-83-197.compute-1.amazonaws.com (34.227.83.197)' can't be established.
ECDSA key fingerprint is SHA256:JAsvJ60FBSvz11EKGi19v9Aow5Zn+8gfG8kqf1K7Ko.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-227-83-197.compute-1.amazonaws.com,34.227.83.197' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Apr 14 06:00:11 UTC 2020

System load: 0.00          Processes: 92
Usage of /: 23.3% of 7.69GB Users logged in: 0
Memory usage: 15%          IP address for eth0: 172.31.34.132
Swap usage: 0%

 * Kubernetes 1.18 GA is now available! See https://microk8s.io for docs or
   install it with:
     sudo snap install microk8s --channel=1.18 --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.
     https://multipass.run/

34 packages can be updated.
0 updates are security updates.

Last login: Fri Apr 10 06:43:09 2020 from 157.33.27.230
ubuntu@ip-172-31-34-132:~$ ls
first
ubuntu@ip-172-31-34-132:~$ cd first
ubuntu@ip-172-31-34-132:~/first$ ls
README.md index.html intro.html node_modules npm-debug.log package-lock.json package.json server.js static
ubuntu@ip-172-31-34-132:~/first$ npm start

> dot_mult_game@1.0.0 start /home/ubuntu/first
> node server.js

Starting server on port 5000
```

6. CONCLUSION

6.1 Conclusion:

By creating a small game based on distributed architecture, we learned and understood the key features and goal of distributed systems like Transparency, Openness, Reliability, Performance, Scalability etc. Also tackled some common problems faced by distributed systems like Security, fault tolerance, coordination and resource sharing between nodes of distributed network. Also understood the advantages of distributed system over centralised system. Thus we conclude that for a given setup distributed system will be far better than centralised system for various important reasons.

6.2 Future Works :

Better handling of traffic at cloud server requires some load balancers like NGINX, etc. Which will improve user experience and performance.

6.3 Application:

Recreational purposes

7 Code Snippet:

<https://github.com/godsaurab/dotGameDS/tree/newVersion>

SERVER

```
// Dependencies.
var express = require('express');
var http = require('http');
var path = require('path');
var socketIO = require('socket.io');
var app = express();
var server = http.Server(app);
var io = socketIO(server);
var cron = require('node-cron');

cron.schedule('*/*15 * * * *', () => {
  players = {};
  count = 0;
  //console.log('restarting server every fifteen minutes');
});
app.set('port', 5000);
app.use('/static', express.static(__dirname + '/static'));
// Routing
app.get('/', function(request, response) {
  response.sendFile(path.join(__dirname, 'index.html'));
});
server.listen(5000, function() {
  console.log('Starting server on port 5000');
});
var count=0;
var players = {};
io.on('connection', function(socket) {
  // console.log("socketId: "+socket.id);

  socket.on('new player', function(playerName) {
    //var newname=count+' '+playerName;
    // console.log(newname);
    //console.log(count);

    players[socket.id] = {
      x: 700,
      y: 300,
      num:count,
      name:playerName,
      score:0
    };
    count++;
  });
  socket.on('movement', function(data) {
    var player = players[socket.id] || {};
  });
});
```



```

    if (data.left) {
        player.x -=5;

    }
    if (data.up) {
        player.y -=5;

    }
    if (data.right) {
        player.x +=5;

    }
    if (data.down) {
        player.y +=5;

    }
});
socket.on('score',function(data){
    players[data].score+=1;
});
});
setInterval(function() {
    io.sockets.emit('state', players);
}, 1000 / 60);
setInterval(function() {
    io.sockets.emit('winner', players);
}, 1000 /60);

```

CLIENT

```

var socket = io();
var movement = {
    up: false,
    down: false,
    left: false,
    right: false
}
document.addEventListener('keydown', function(event) {
    switch (event.keyCode) {
        case 37: // arrowleft
            movement.left = true;
            break;
        case 38: // arrowup
            movement.up = true;
            break;
        case 39: // arrowright
            movement.right = true;
            break;
        case 40: // arrowDown
            movement.down = true;
            break;
    }
});

```

```

    }
  });
  document.addEventListener('keyup', function(event) {
    switch (event.keyCode) {
      case 37: // arrowleft
        movement.left = false;
        break;
      case 38: // arrowup
        movement.up = false;
        break;
      case 39: // arrowright
        movement.right = false;
        break;
      case 40: // arrowdown
        movement.down = false;
        break;
    }
  });
  socket.emit('new player', playerName);
  setInterval(function() {
    socket.emit('movement', movement);
  }, 1000 / 60);
  var canvas = document.getElementById('canvas');
  canvas.width = 1500;
  canvas.height = 650;
  var context = canvas.getContext('2d');
  socket.on('state', function(players) {
    //console.log(players);
    var colours=['silver','grey','red','black','maroon','yellow','olive','green','blue','purple','aqua'];
    context.clearRect(0, 0, 1500, 650);

    loop1: for(var id1 in players){
      var player1 =players[id1];
      for(var id2 in players){
        var player2=players[id2];
        if((id1 != id2) && ((Math.abs(player1.x-player2.x)<20) && (Math.abs(player1.y-
player2.y)<20) ))
          //if((id1 != id2) && (player1.x==player2.x && player1.y==player2.y)
&& (player1.x!=700 || player1.y!=300))
          {
            //console.log(player1.name+' wins !');
            context.clearRect(0,0,1500,650);
            context.strokeText(player1.name+' GOT a POINT! ',1300,50);
            socket.emit('score',id1);
            //player1.score=player1.score+1;
            break loop1;
          }
        }
      }
    }
    for (var id in players) {

```

```

var player = players[id];
//console.log(player.name);

context.fillStyle = colours[id.charAt(id.length-1).charCodeAt(0)%10];
context.beginPath();
context.arc(player.x, player.y, 15, 0, 2 * Math.PI);
context.fill();
context.font="15px Georgia";
//context.fillStyle = colours[id.charAt(id.length-1).charCodeAt(0)%10];
context.strokeText(player.num,player.x,player.y-20);
context.strokeText(player.name,player.x+20,player.y);
context.strokeText(player.score,player.x,player.y+25);

}
});
socket.on('winner',function(players){
    var max=0; var playername; var playernum;
    for(id in players){
        player=players[id];
        if(player.score>max){
            max=player.score;
            playername=player.name;
            playernum=player.num;
        }
    }
    if(playername!=undefined)
    document.getElementById('winner').innerHTML='WINNER      :      ['+playernum+' '+playername;
});

```