Izzy
SE2
this is to remove all the different pdfs

Chapter 1 AWS

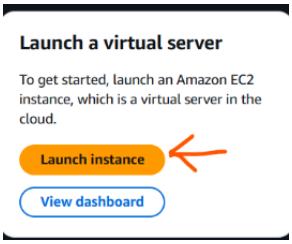
1.1 Making an EC2 instance:

1.1.1 Go to Amazon EC2

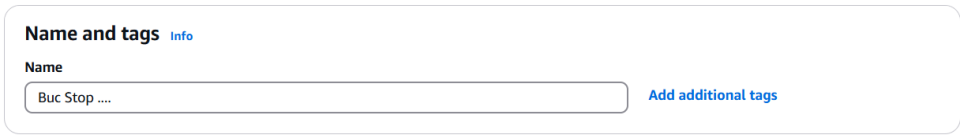
Make sure you are in the EC2 portion of AWS

1.1.2 Launch virtual server

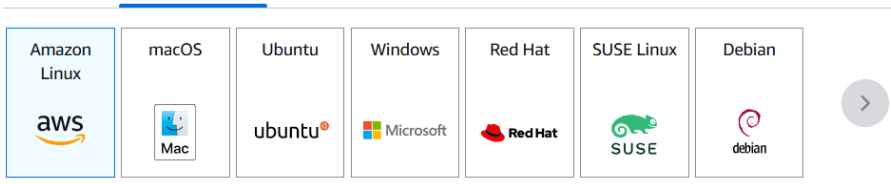
This will take you to make an instance, click on it



1.1.3 Name your instance



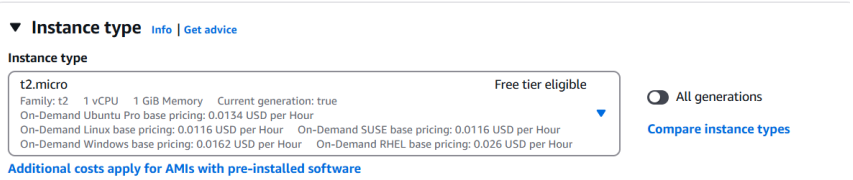
1.1.4 Locate the OS you would like to run



Izzy
SE2
this is to remove all the different pdfs

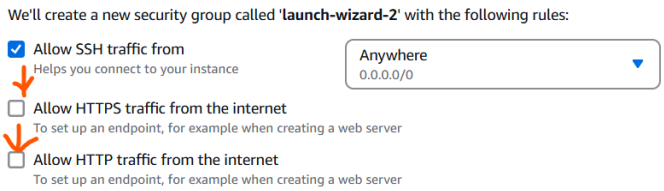
1.1.5 Choose resources

You will need to choose the number of resources you wish to have. This one has t2. Micro, which has a free tier eligible



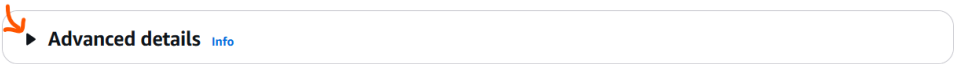
1.1.6 Allow HTPPS and HTTP

Make sure to allow HTPPS and HTTP



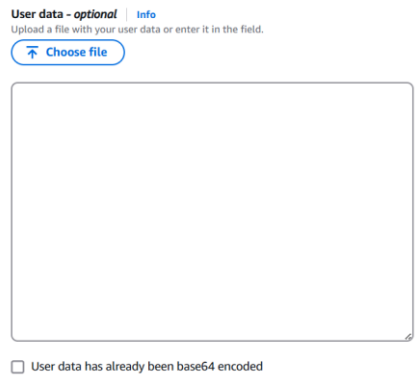
1.1.7 Click on Advanced details

Click on Advanced details and scroll all the way down.



1.1.8 Run on Initialization

“Anything you put in here runs only when the instance is initialized, and never again.” We can put all our scripts in here that we want to preconfigure the EC2 with.



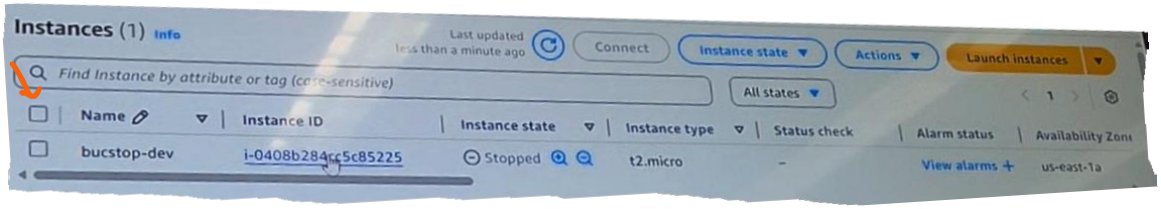
Izzy
SE2
this is to remove all the different pdfs

1.1.9 An example of this is –

1.2 Running an instance:

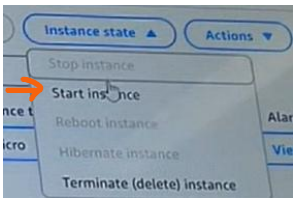
1.2.1 Current instances

This will show your current instances after they have been created. Make sure to click on the instance you would like to run



1.2.2 Click start instance

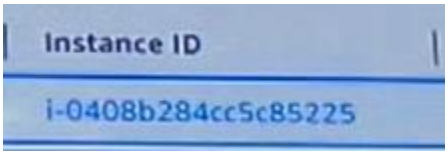
Next, go to instance state, and click start instance



1.2.3 Click instance Id

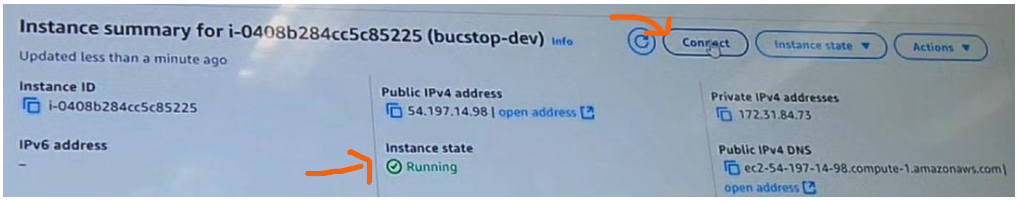
Click the instance Id to take you to the instance summary

Izzy
SE2
this is to remove all the different pdfs



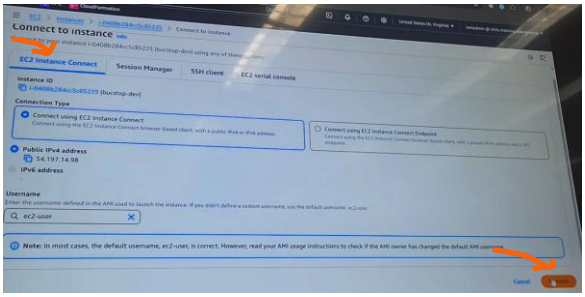
1.2.4 Launch instance

To launch the instance, make sure it is running, then click connect.

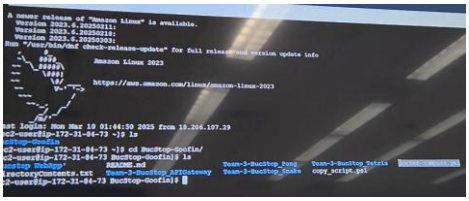


1.2.5 Different ways to run the instance

There will be different ways to run the instance choose what you would like. For this instance, we are doing EC2 instance connect. Then click connect



1.2.6 Once done, it should look like this



Izzy
SE2
this is to remove all the different pdfs

1.3 Cloud watch logging information:

1.3.1 Dashboard

The instance is still monitoring various metrics which can be viewed in both EC2 and Cloudwatch, and I have set up a custom dashboard in Cloudwatch with every basic metric that is monitored. However, none of the metrics are being logged to a log file in Cloudwatch as we have been unable to set up a Cloudwatch Agent in the EC2 instance for the above mentioned reasons.

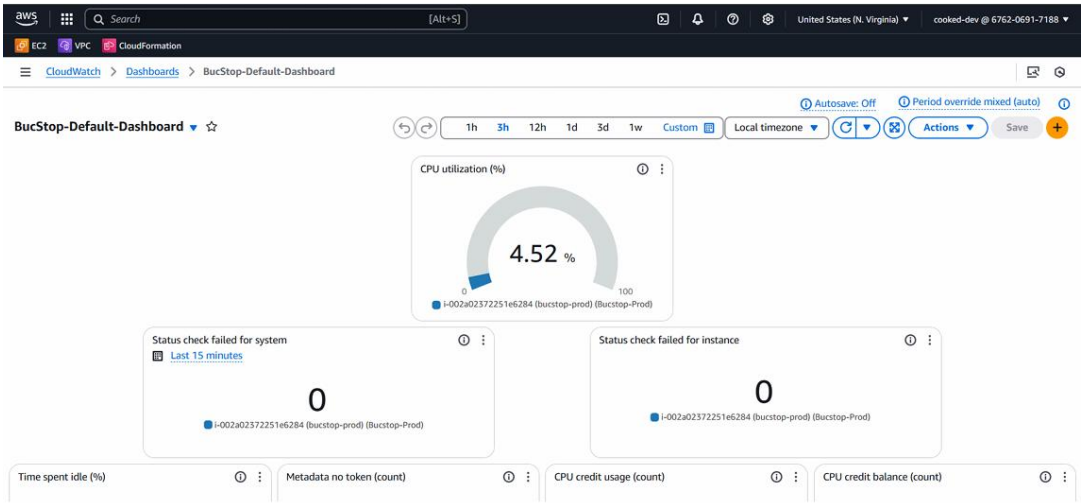


Figure 1, An image of the custom dashboard in Cloudwatch

1.3.2 Log groups

In addition, there is a Cloudwatch Log Group and Log Stream set up in Cloudwatch. The Log Stream will be where logs are accessible whenever the task is finished.

Izzy
SE2
this is to remove all the different pdfs

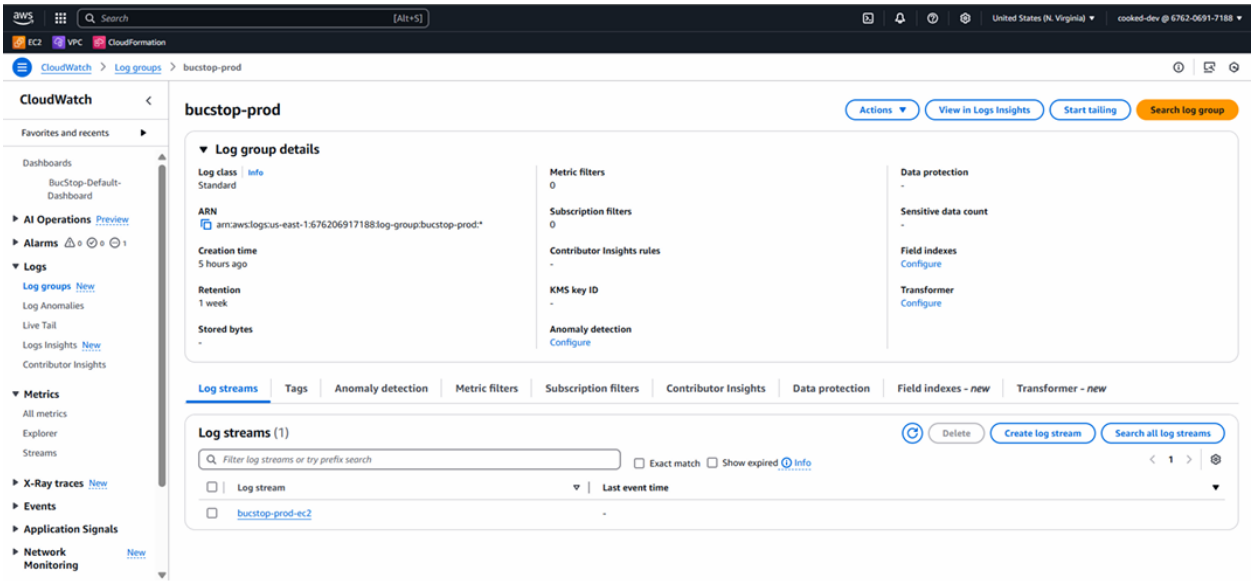


Figure 2, An image of the log group (bucstop-prod) and its log stream (bucstop-prod-ec2)

1.3.3 IAM role

Part of configuring a cloudwatch agent is setting up an IAM role to give the EC2 instance certain permissions.

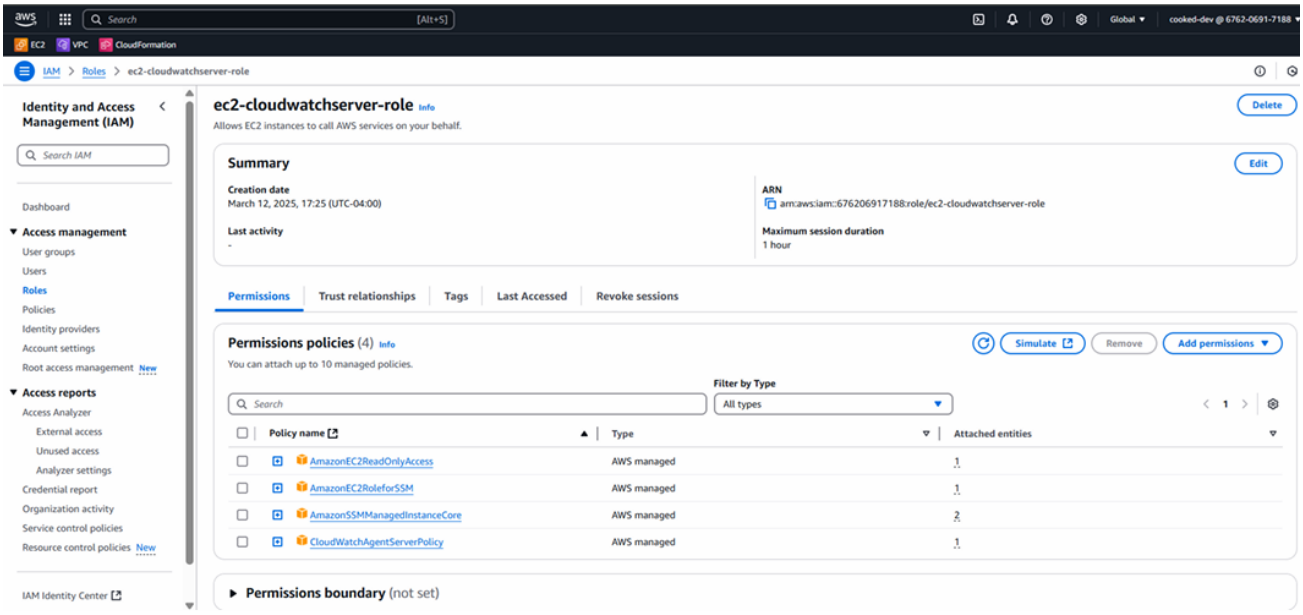


Figure 3, An image of the IAM role we created to attach to the EC2 instance

Izzy
SE2
this is to remove all the different pdfs

1.3.4 Modifying the IAM role

You can modify the IAM role attached to the EC2 instance through a security action on the EC2 Details page:

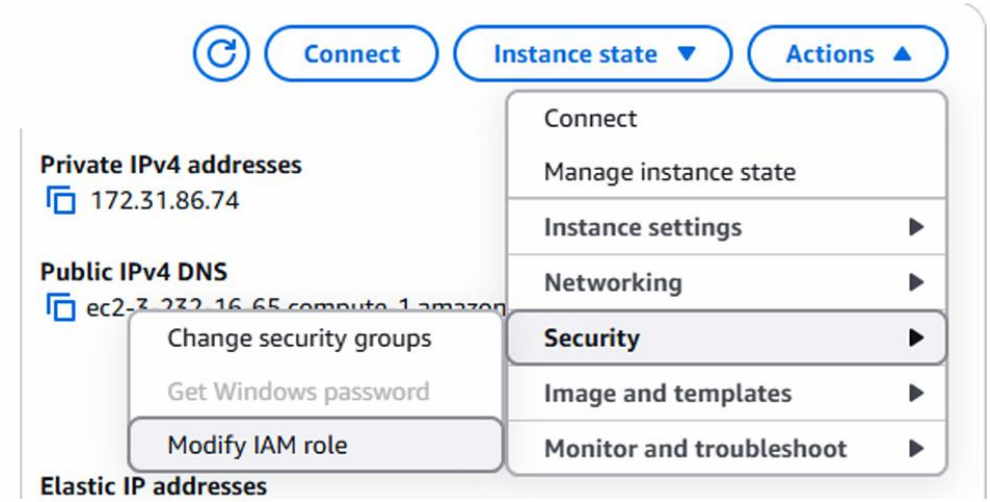


Figure 4, An image of how to modify the IAM role attached to the instance

1.3.5 Current issue

However, whenever we go to actually attach the role to the instance we run into the following issue:

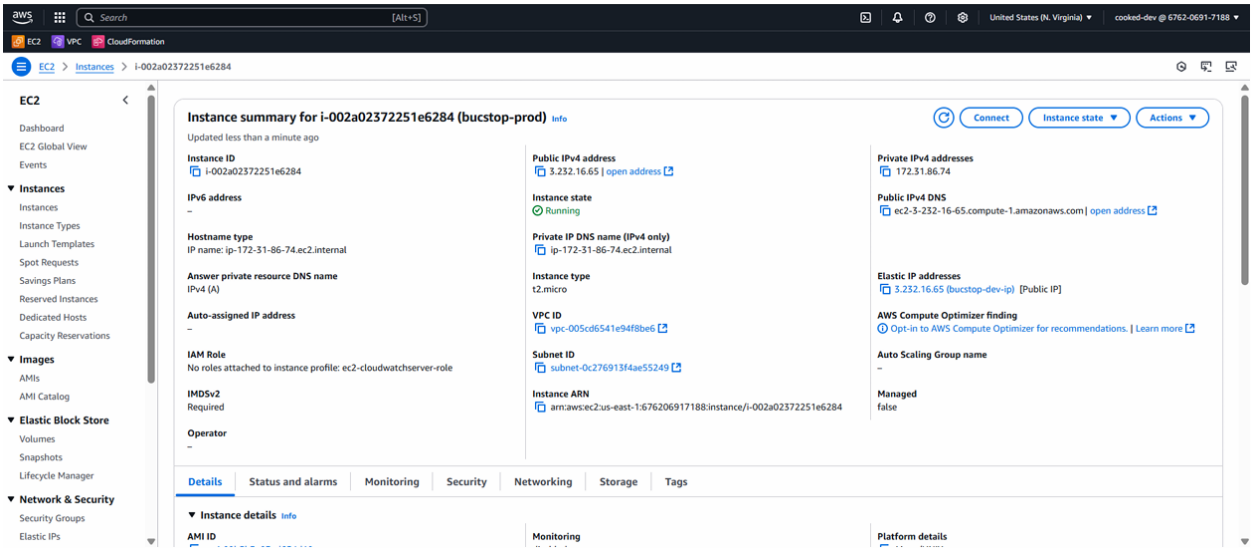


Figure 5, A full page view of the EC2 instance profile, showing the IAM role not attached

Izzy
SE2
this is to remove all the different pdfs

When attempting one of the approaches where we would configure the cloudwatch agent from the ec2 instance page under the monitoring section when the instance is running, we encounter the following error:

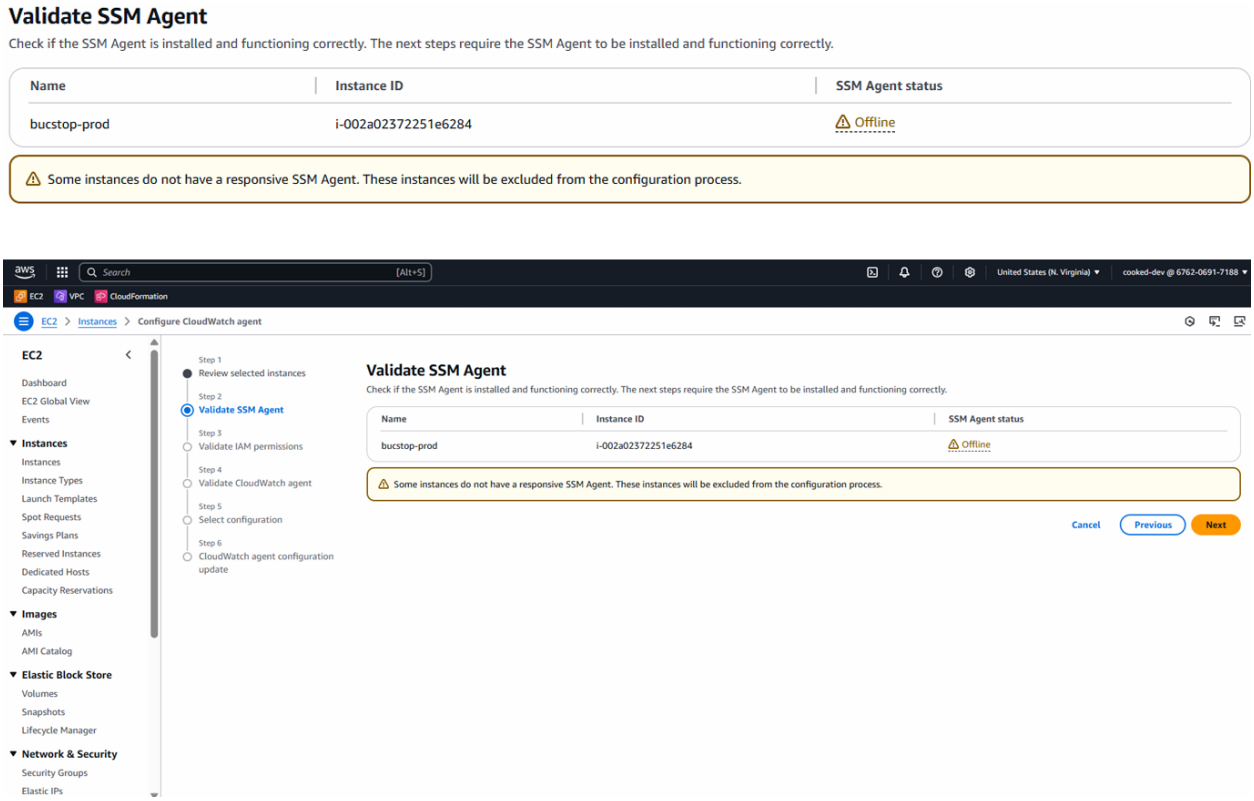


Figure 6, A full page view of the error where the SSM agent is unresponsive.

1.3.6 Resources

Some useful videos with different approaches that we have tried:
https://www.youtube.com/watch?v=hqGnDzCIMBg
https://www.youtube.com/watch?v=7UlFuwONrvQ

Chapter 2 API Gateway

2.1 Understanding a Gateway

2.1.1 What is an API Gateway:

An API (Application Programming Interface) Gateway is a midpoint between a software application and other applications or microservices that handles all communication between them. The API Gateway handles this communication by receiving an API call or

Izzy
SE2
this is to remove all the different pdfs

request from the application. The API Gateway then routes the API call to the correct microservice(s) to gather the requested data. The API Gateway then handles the logic for combining all of the data that it gathered from the microservice(s) into a single package that it can then pass to the application that originated the API call.

API Gateways come with a lot of benefits related to security, scalability, and efficiency. For security, an API Gateway can be used to monitor requests, responses, and errors, can provide authentication and authorization, and is able to help with DDoS attacks through techniques like rate limiting. With scalability, they can help with load balancing by routing their requests to different microservices to ensure that microservices do not end up being overwhelmed with traffic. With efficiency, an API Gateway can cache responses to common API calls and can reduce the complexity of a workflow to create a simpler and more efficient procedure.

They do present a single point of failure and have some scalability issues as when given too many requests to handle they can result in some higher latency.

Here is a visual from one of our resources. In this example, there is additional abstraction by using an identity provider and service registry that will likely not be needed for our project.

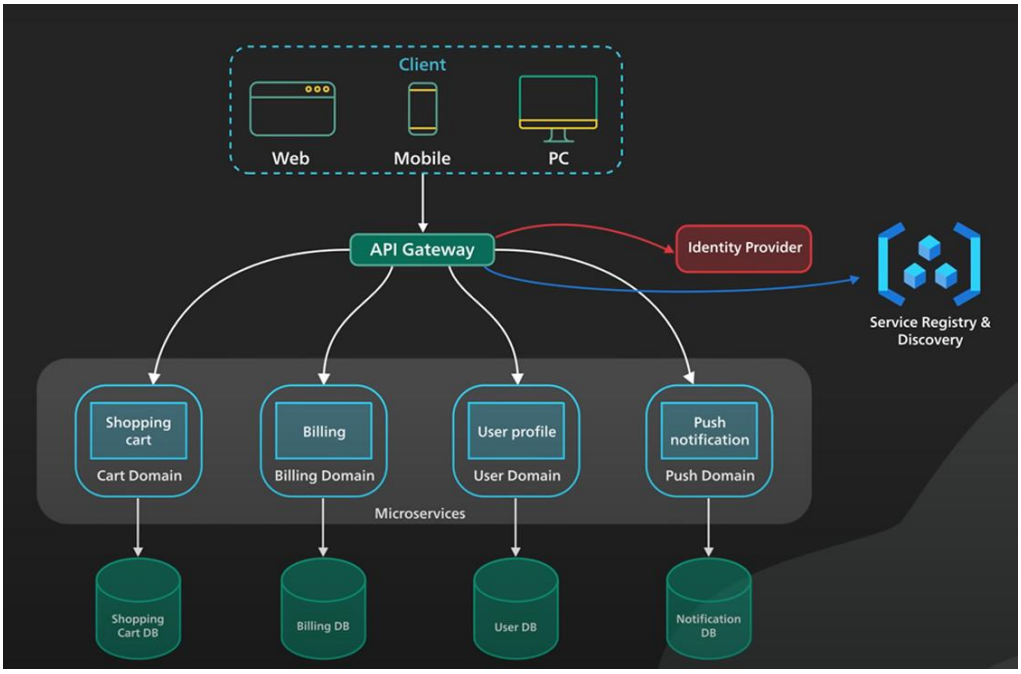


Figure 7, API diagram

Izzy
SE2
this is to remove all the different pdfs

2.1.2 What should an API Gateway do:

An API Gateway should, as mentioned above, handle requests from an application for data from other microservices. This means that the application never actually interacts with the microservices directly, instead interacting with the API Gateway which then interacts with the microservices to gather the data requested, combine it into a usable format for the application, then return it to the application.

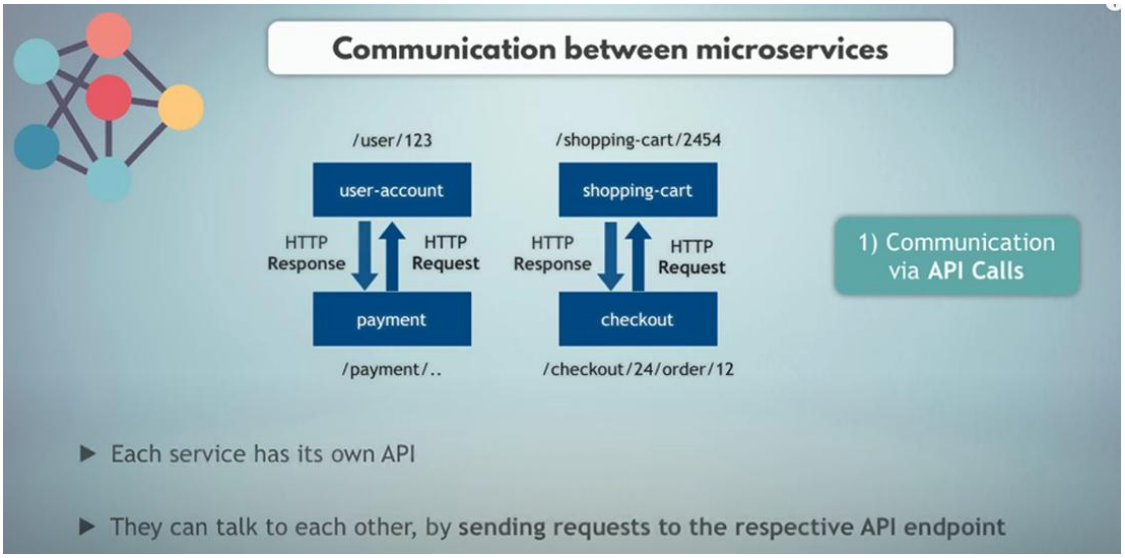


Figure 8, communication between microservices

The API Gateway can handle traffic in various ways, such as burst limits, throttling for an application or individual user, and Queues of API messages that will be handled over time.

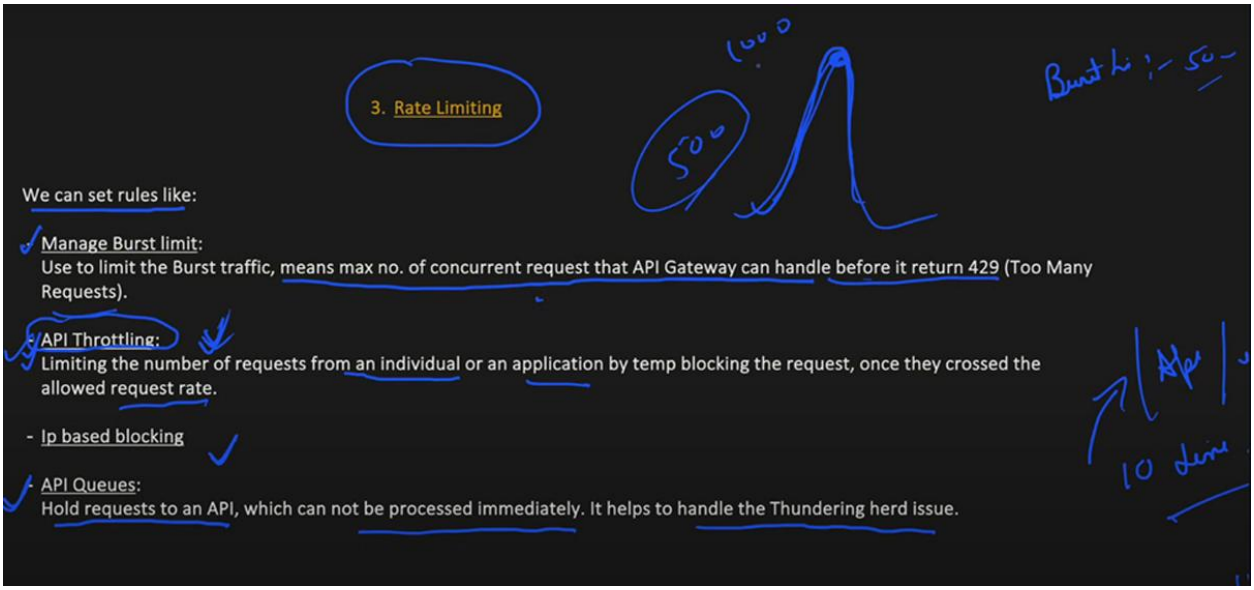


Figure 9, Rate limiting

Izzy
SE2
this is to remove all the different pdfs

2.1.3 What does our API Gateway do and not do to facilitate this workflow:

Our API Gateway Solution has a GameInfo.cs file that appears to be a class for making objects that would store collected data from each microservice. These objects would be able to pass the requested information to the application, following the workflow of an API Gateway.

Our API Gateway Solution also has a GatewayController.cs file that appears to handle the logic for communicating with the different microservices to retrieve a GameInfo object and add these objects to a list that it would return to the application. It is set up to get the game information from every microservice that is currently present (Snake, Tetris, and Pong).

Our API Gateway Solution also has a Program.cs file that appears to set up the pipeline for HTTP requests using Swagger/OpenAPI. This should allow the communication between the application and the API Gateway and the communication between the microservices and the API Gateway to function.

However, our API Gateway does not actually communicate between services. We have hardcoded values for the game info objects within the games.json file in the BucStop Repo. The files above mimic parts of the actual workflow, but are not being used so we do not know if they are functional.

So, our API Gateway has some sample code for the Application making an initial request for information on what games are available and the information related to them in order to set up a thumbnail for the carousel and have their information displayed within the app. This sample code mocks the workflow for being able to contact the microservices to gather the GameInfo objects which contain the information the application needs, combines them together, then sends this to the application.

In addition, our solution is not actually using the API Gateway for running the game as the Javascript files for the games are not stored in their own microservices but are instead stored in the BucStop Repo. The application just runs whichever game the user has selected without contacting the API Gateway. The workflow that should be followed instead would involve the Javascript files for the games being stored within their own microservices. The user would select a game which would send a request to the API Gateway to either run the game or pass the Javascript file for the game back to the application so that the application can run the game depending on how we choose to design our solution.

For our current API Gateway, there are 3 endpoints seen below to get the game data for the 3 games. As mentioned above, these are not being utilized yet.

Izzy
SE2
this is to remove all the different pdfs

```
/// <summary>
/// Handles GET requests to retrieve game information from microservices.
/// </summary>
/// <returns>A collection of GameInfo objects.</returns>
[HttpGet]
public async Task<IEnumerable<GameInfo>> Get()
{
    try
    {
        var SnakeTask = AddGameInfo("https://localhost:1948", "/Snake" ); //Snake
        var Tetristask = AddGameInfo("https://localhost:2626", "/Tetris"); //Tetris
        var PongTask = AddGameInfo("https://localhost:1941", "Pong"); //Pong
        await Task.WhenAll(SnakeTask, Tetristask, PongTask);
        return TheInfo;
    }
```

```
public async Task AddGameInfo(string baseUrl, string endpoint)
```

2.1.4 Resources:

https://www.ibm.com/think/topics/api-gateway

https://www.youtube.com/watch?v=lTAcCNbJ7KE

https://youtu.be/rv4LlmLmVWk?si=-G_E3UOgvdE7EN0

Chapter 3 Rollback

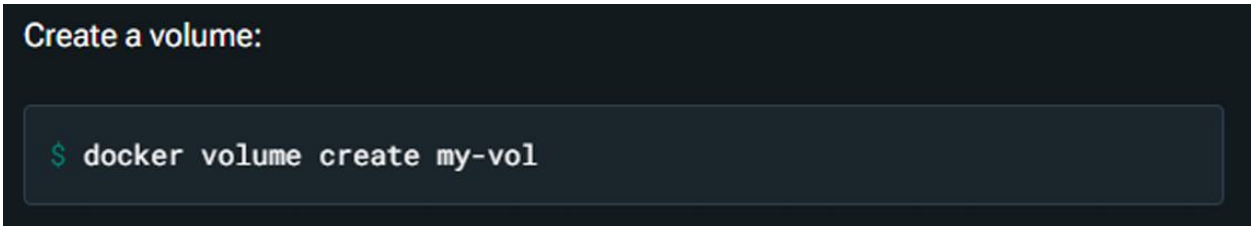
3.1 Potential solutions for keeping persistent data during rollback:

3.1.1 Docker volumes

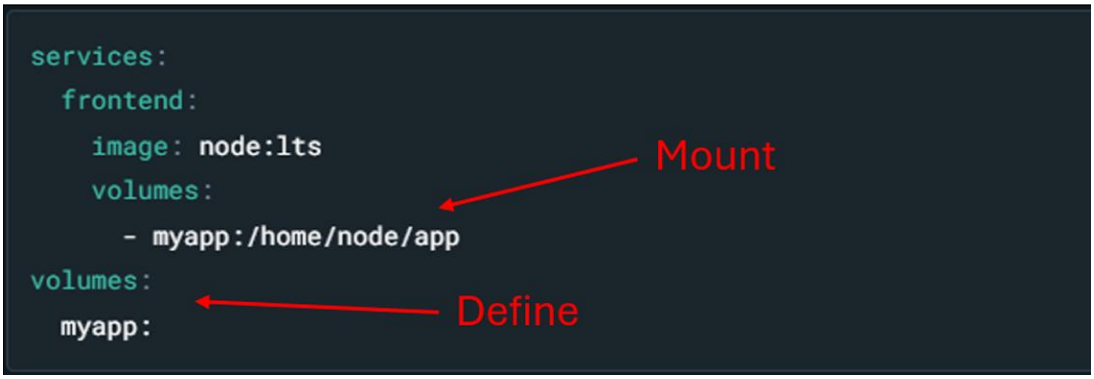
First need to mount a volume, which gives the volume access to a specific directory inside a container (ex. Playcount.json or log files)

Izzy
SE2
this is to remove all the different pdfs

Define/Create the volume



Volumes can be mounted and defined inside of docker compose. Running docker compose up for the first time creates a volume. Docker reuses the same volume when you run the command subsequently.



3.2 Some Implications for Docker Volumes:

Docker system prune command in deployment script may remove volumes

If a new EC2 instance is created, volumes will most likely be lost since they are stored on the EC2 host machine (Could use AWS Elastic Block Storage to store volumes)

3.2.1 Resources

Documentation Link: https://docs.docker.com/engine/storage/volumes/

Link to this document for editing:
https://docs.google.com/document/d/11LTixLWicBxM4XUPWyNRi4D5uFsL58xRFvQ-kOm0b9s/edit?usp=sharing

Izzy
SE2
this is to remove all the different pdfs

Chapter 4 Pong

4.1 Review

4.1.1 Overview:

The game “Pong” is integrated into a web application using a microservice architecture, where the game information is provided by the Pong API. The web app interacts with this API to fetch game metadata and then loads and runs the game via a JavaScript file (pong.js).

4.1.2 Key Dependencies:

1. The web app communicates with the Pong API through an HTTP GET (found in PongController.cs) request to the /Pong endpoint. This API responds with a GameInfo object containing details such as the game title, description, and the path to the game logic file (pong.js). The web app uses this data to load and display the game.

2. Pong is rendered on a element in the web app, where the pong.js file handles the game logic. The Content field in the API response points to the pong.js file, which should be served as a static asset by the web app, typically located in a /js/ directory (e.g., wwwroot/js/). In this case, it is located on the following file path: Bucstop WebApp/BucStop/wwwroot/js/pong.js. If these files aren’t properly delivered to the browser, the game logic and visuals wouldn’t load at all.

3. The web app must include a element with the id game, as the game logic in pong.js relies on this canvas to render the game’s graphics, such as the paddles, ball, and score display. Keyboard input (arrow keys and spacebar) is used to control the game. Without the proper HTML structure (or the appropriate script tags to load the JS), the interactive elements wouldn’t display or function.

4. The pong.js file implements the game logic, including paddle movement, ball physics, collision detection, AI paddle control, scoring, and the game loop. The game runs continuously using the requestAnimationFrame function, updating the game state and rendering to the canvas.

4.1.3 Potential Issues:

If pong.js is not correctly served from the specified directory (/js/pong.js), the game will not load. Isn't that surprising?

Izzy
SE2
this is to remove all the different pdfs

4.1.4 Conclusion:

Pong is integrated into the web app through an API that serves game metadata and assets. The web app makes requests to this API to retrieve details about the game, such as its title, description, and thumbnail. It also uses the API to access the game's JavaScript file (pong.js), which is then loaded and rendered on a canvas in the browser. For the game to function properly, it's essential that static files (like pong.js) are correctly served by the web server and that the HTML includes the proper structure to display the game, such as a canvas element for rendering.

Chapter 5 Possible files to remove?

5.1 Already documented in here

5.1.1 Rollback research

5.1.2 Cooked pong research

5.1.3 Cooked API gateway document

5.1.4 Cloud watch logging information

5.2 Not documented yet

5.2.1 Tetris Documentation

5.2.2 Snake Docs

5.2.3 Emptying games . Json

5.2.4 Docker research

5.3 Documentation in WebApp

5.3.1 Cookies research

5.3.2 Selenium documentation

5.3.3 Steps for deploying updated

5.3.4 Change Log

Izzy
SE2
this is to remove all the different pdfs