AWS Handover & Shutdown

For Future Developers

AWS is fairly complex, so this section is designed to explain the key services used for this project, and how to use them if needed (as a warning, this section is fairly technical)

Helpful Resource For Understanding Our Setup

A particularly helpful resource for understanding our AWS setup can be seen in is YouTube video, which uses a very similar configuration as what we have:

▶ AWS Load Balancer HTTPS Setup with Route 53 and Certificate Manager & HTTP Redirect to HTTPS
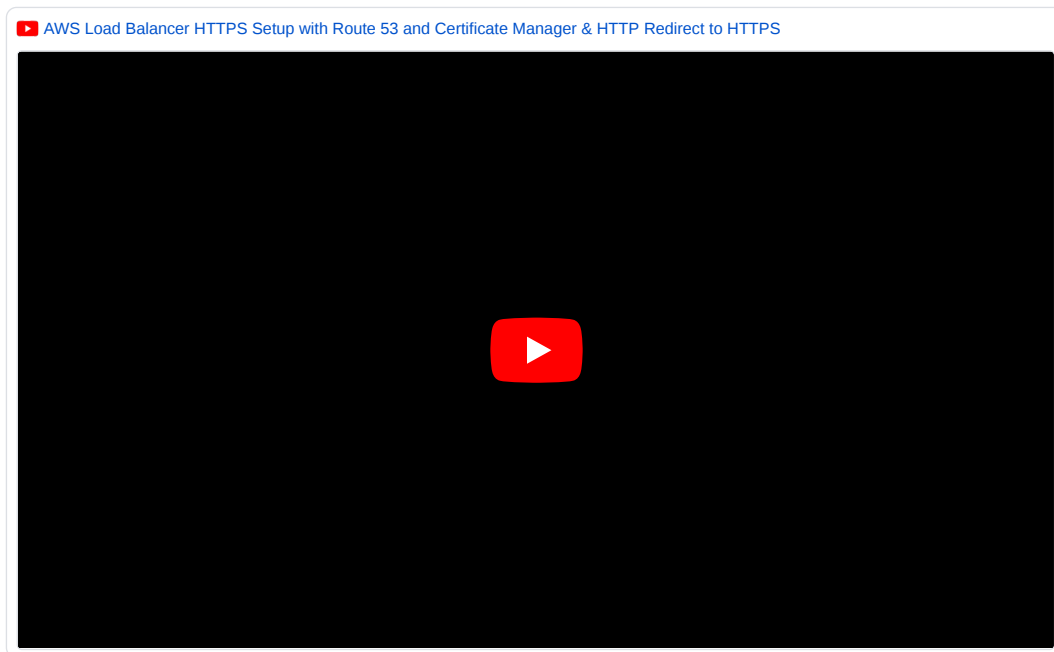
Elastic Compute Cloud (EC2)

- All of our instances run in EC2
- This is the best place to observe metrics, such as usage and resource utilisation
- If the service is crashing, for example, check in EC2 if the CPU usage is above the limit
- In general, we do not actually manage our deployments in EC2, they are instead managed by beanstalk (see below for details)

Elastic Beanstalk

- This is where we manage our actual instances and containers
- Elastic Beanstalk automatically scales our application up and down depending on usage requirements, as well as manages the groups that many other services are attached to
- Environment Properties are stored here (under configuration), including:
  - Root admin login credentials
  - Secret keys
  - Email credentials for automated emails
- Logs can also be viewed in the elastic beanstalk under the main page. This shows the deployment of all three parts of the application (MySQL, Backend, Web App). If any errors occur in production, checking these logs is typically valuable

Elastic Load Balancing

- This is also located within the beanstalk section of the console
- We use a load balancer to manage the SSL certificate and push all HTTPS requests to HTTP requests internally (to springboot and nginx)
- Not much configuration should be required here, as long as the security groups do not change, everything should remain working and attached to the correct load balancer

CodePipeline

- We used code pipeline for our CI/CD from GitHub
- If a future team wants to continue on the project, they can either gain access to our old repository if this is allowed (Potentially only possible if project is still attached to Unimelb), or create their own repository and copy in files
- In case of using a different repository, the CodePipeline will need to be configured to read changes in this repository to deploy to production
- At current, any changes main in the 'main' branch of our repository are pushed into production in the MMS application

Route 53

- Route 53 manages our IP address
- The site is currently hosted on https://medsecapi.com
- If a different IP address is desired in future, this is the place to look. The new IP will need to be added to the autoscaling group in elastic beanstalk so that the load balancer knows where to push to. Similarly, the SSL certificate in Route 53 will need to be redirected
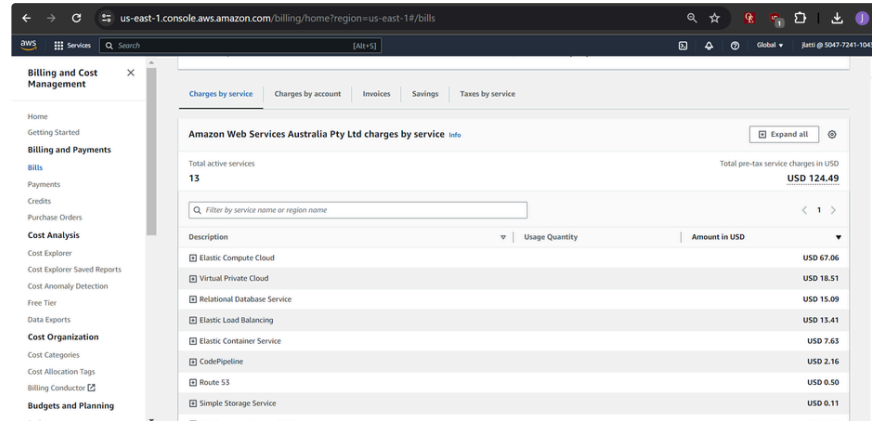
Certificate Manager

- The certificate manager can be used to check the SSL certificate we are using for HTTPS
- This should not need to be configured or edited, but is helpful as a potential resource if the project setup is changed in future

For Shutting Down Services

There are a number of services we are currently using in AWS, which are billed on a monthly basis. In the scenario that the app is ever shut down (or if the client goes with the Koala team's project), these can be easily shut down in the AWS console to save this monthly cost. At current, the wombat team is the only team using the main AWS account, as the Koala team have created their own account for prototyping purposes.

To see which services are running, the easiest way is to head to the billing tab: Biling tab

Billing Tab in AWS

Each of the services shown in the 'Billing by Service' tab will require manual shutdown. In the search bar at the top of the AWS console, search for each service with a price and navigate to that page to shut down service. Shutting each service down should be fairly self-explanatory (but a bit time consuming, sorry!). As an example, to shut down the 'Elastic Compute Cloud', search for it and click this link (it is actually called EC2 in shorthand here):
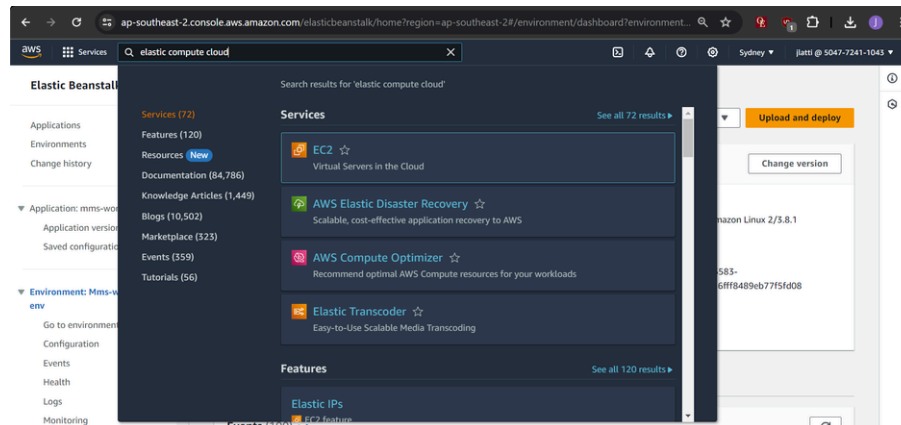
Searching for EC2

Once on the EC2 page, you will see there is 1 instance running (It could be more if the other team is also using them (In this case, ours is always the 'wombat' labelled options). Click on this instance, and delete it through the 'Actions' heading below:
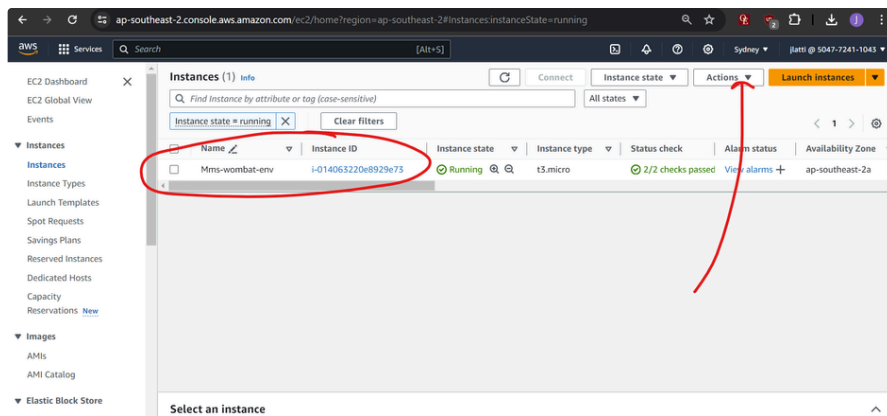
Deleting an EC2 Instance

In general, you can be confident that the 'Wombat' and 'Koala' services are separated, so if only the Wombat service is meant to be shut down, you can safely close/stop all Wombat related things (Instances, Environments, Load Balancers etc.)