AWS_SolutionsArchitect_Task_1

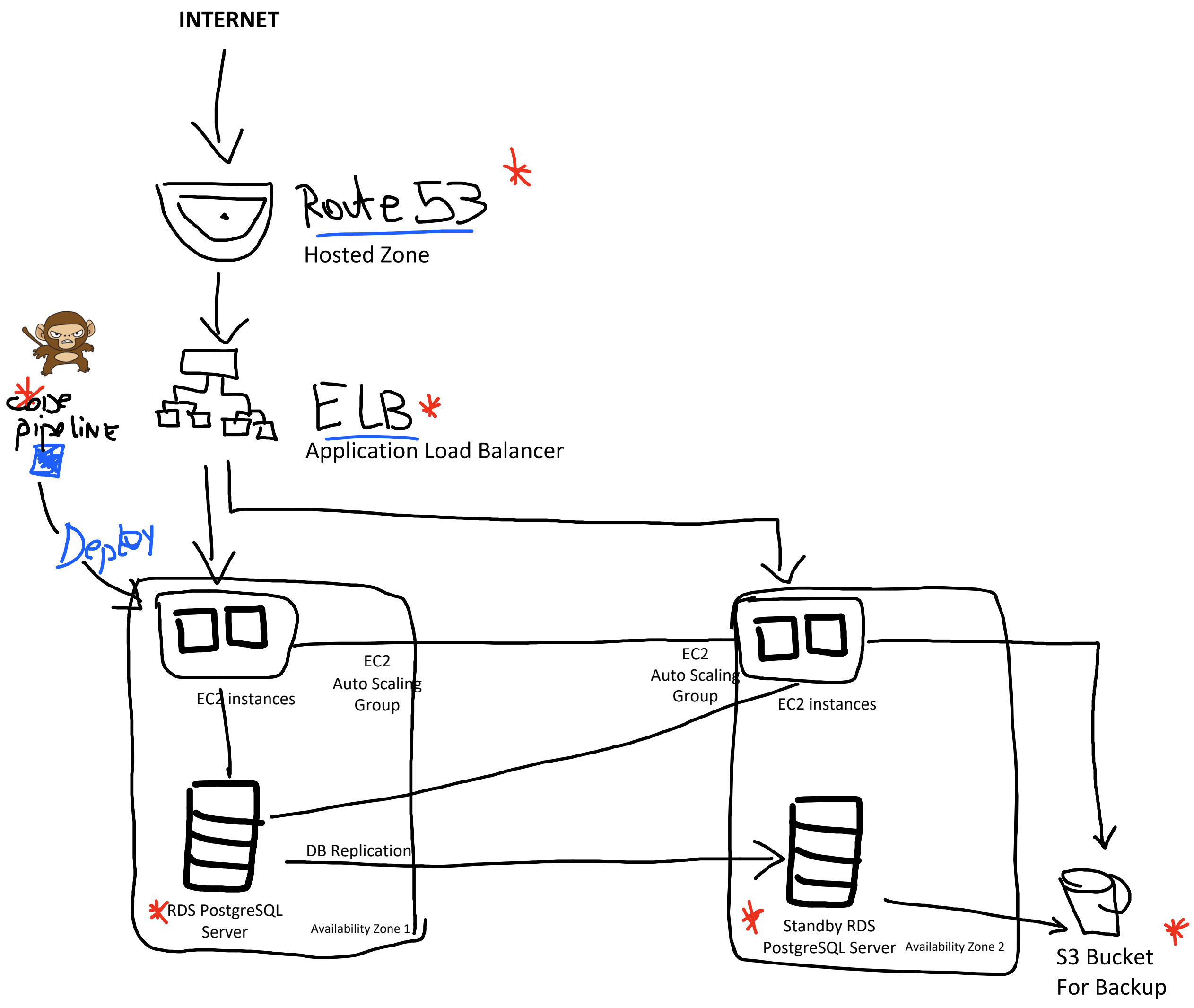
Saturday, 19 July 2025 19:23

Task 1 is about drafting an email in reply to a client's request about designing an architecture of ideal scenario to support their growth, which has been in a consistent path for a while, especially while getting advertised on recent news which helped their brand awareness.

We need to show her an architecture diagram and explain:

- What each part of the architecture is
- Why it was chosen
- Costs rough estimate

In 600-700 words



Re: Scaling and Deployment Solutions for Your Python Web Application

Hi Lilly, thank you for your message and for giving us the opportunity to assess your current situation.

The details you shared are exactly what most of the companies (which generally are in extreme growth and demand) that reach us face the same sort of issues.

AWS has a wide variety of products and I've gone through a deep dive with the best minds on our team to come up with a solution that will eliminate those problems straight from the root.

I'd be delighted to make myself available any day next week to go through what we have prepared for you. What is the best time for a quick 30min sync call on your schedule?
You can reply on this email or click here to schedule (calendly/luisfaria).

In case you want to take a sneak peak in the Architecture Diagram, you can refer to notes taken below:

'''
Our final suggestion is to use **AWS Elastic Beanstalk**, which is AWS' PaaS that simplifies deploying/managing web apps and services with:
- infrastructure provisioning
- load balancing
- auto-scaling
- health monitoring

This will allow your team to focus on development and let we take care of making it available.

An overview of the main components can be found below:
1. **Elastic Beanstalk**: zero-downtime deployments + auto scaling based on traffic patterns.
2. **EC2 instances**: Auto scales up during traffic pikes and scales down during quiet periods. 2 instances for each Availability Zone
3. **ELB**: Elastic Loading Balancer to distribute traffic across multiple app instances, making response times lightning fast
4. **RDS**: Automated backups + point-in-time recovery, database redundancy, security.
5. **S3**: a bucket for static content hosting, cost-effective storage.
6. **CodePipeline**: No down-time, enables CI/CD, rollback capabilities.
7. **Route53**: DNS service for high availability, health checks, traffic routing

The architecture diagram can be found attached.

Next steps:
1. We'll help you containerize your Python application
2. Set up the Elastic Beanstalk environment
3. Migrate your database to RDS with minimal downtime
4. Implement the CI/CD pipeline
5. Gradually shift traffic to the new infrastructure

Note: we kindly named this project "Fastier Architecture Revamp", that we'd like to develop as a Case for our series "Success Stories" (https://aws.amazon.com/solutions/case-studies)

Warm regards,
Luis Faria - Solutions Architect
AWS - Amazom Web Services