NEW YORK UNIVERSITY

Cloud Computing

CSCI-GA.3033-010 - Spring 2020

Cloud Computing Project 2 Report

Submitted By:
Urvish Desai
Satyajeet Maharana
Shreya Kakkar

We hereby declare that all the work submitted in this assignment has been conducted by the team members only. We have not collaborated with anyone or copied material from other sources in finishing this assignment. All the code used in this project is hosted at: https://github.com/satyajeetmaharana/Image-Captioning-IoT-Google-Cloud

Urvish Desai
Satyajeet Maharana
Shreya Kakkar

Total in points (100 points total): _____
Professors comments:

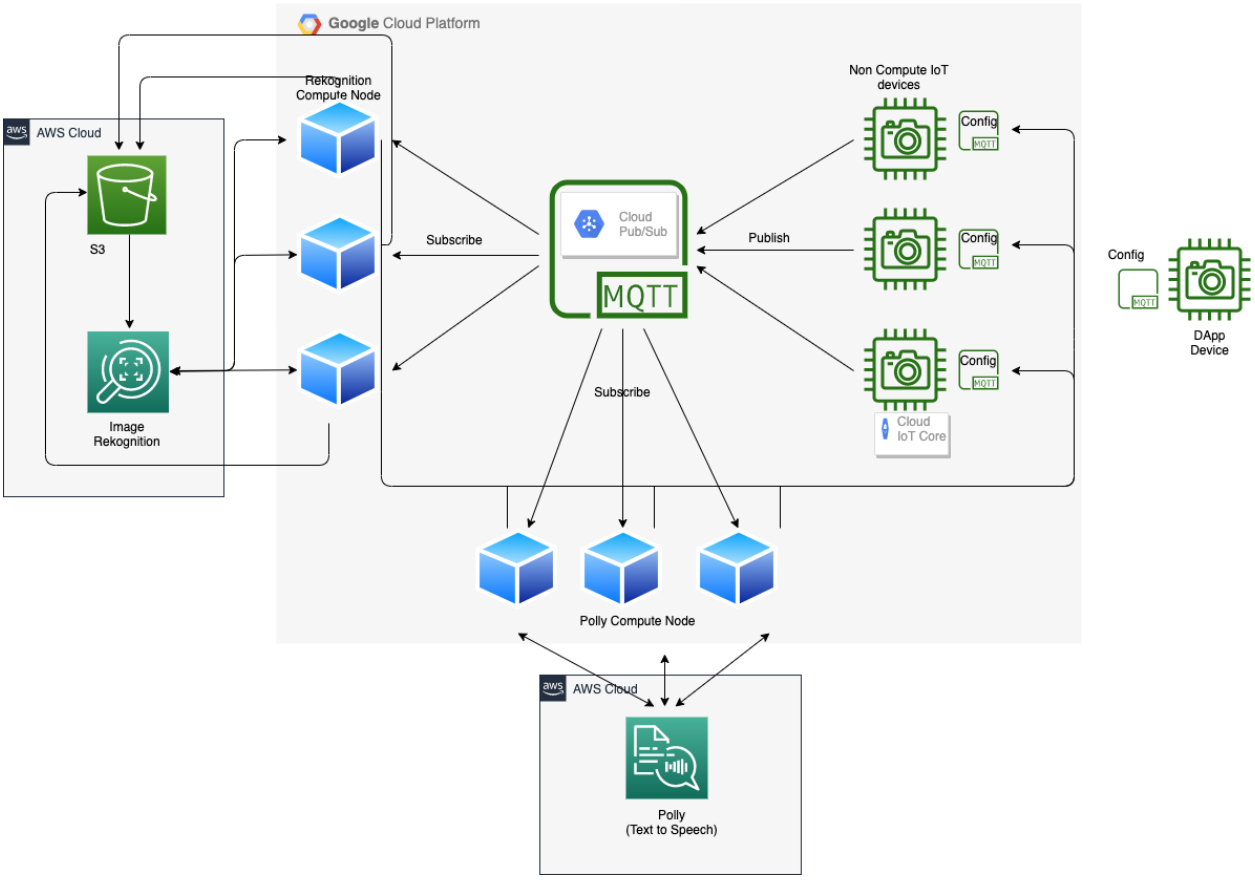
Problem Statement

Statistics show that at least 2.2 billion people have blindness or a vision impairment of some kind globally. Given that a huge ratio of our population is dealing with this handicap, our project is to make an application that helps alleviate the issues faced by visually impaired people. We have built a visual aid with the help of IoT devices and cloud resources that identifies obstacles in the IoT devices line of sight and warns the user about its presence. This application while being very simple in its idea will prove to be of utmost importance for the users.

The user should have access to an IoT device with a camera and speaker in it. This device captures images from the camera and sends it across the network, where a trained ML model identifies the objects in this image and returns these labels. The labels are further fed into a model that converts text to speech and the final caution warning is played on the IoT's speaker to help the user. This application has a payment portal and requires a humble fee of 0.02 ether which is required in order to use the services.

Design Details

We provide a decentralized solution that is migrated completely on the cloud for the purpose of this problem. Fig 1 diagrammatically displays the architecture that we have used in order to build our application. The model can be broadly divided into four major components namely, non-compute IoT devices, rekognition compute nodes, polly compute nodes and a single DApp node. Devices interact with one another in order to ask for services and provide results over a MQTT publish subscribe model. Each of these components and the communication between them has been described in greater detail below:



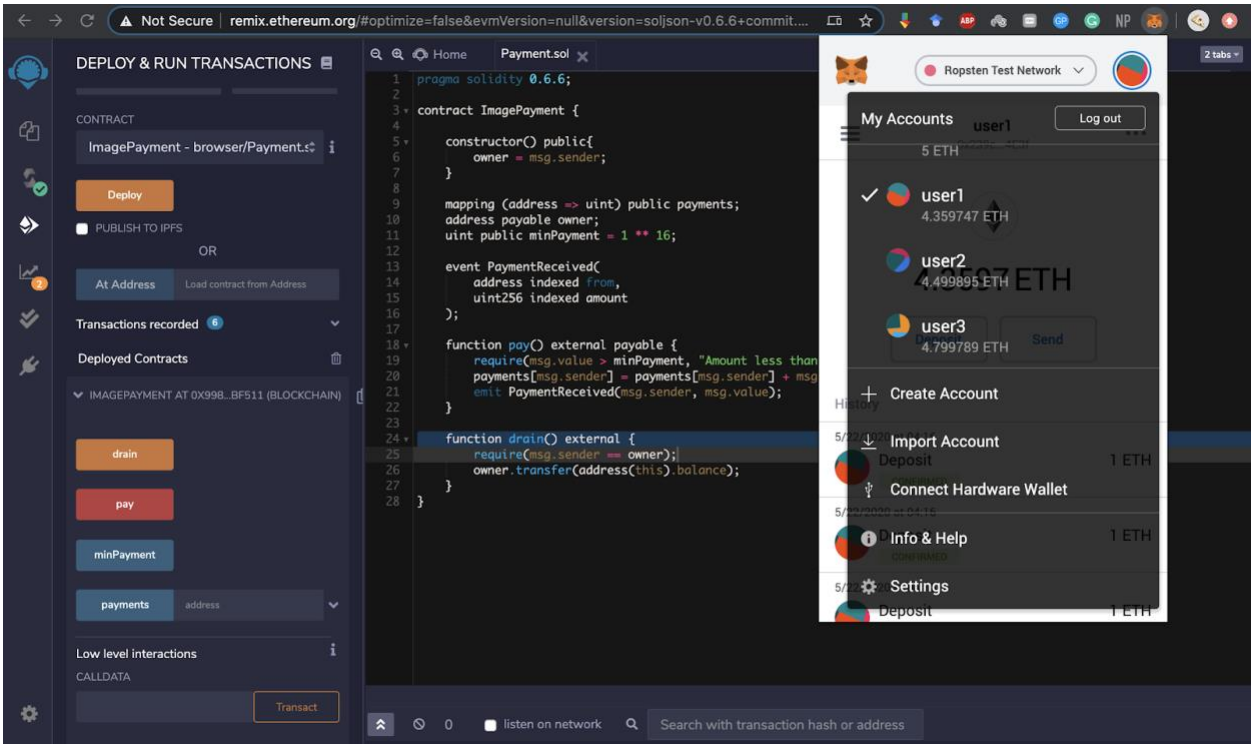
Non-Compute IoT Devices: The devices shown on the left-hand side (in a vertical queue) are the non-compute IoT devices. As per our problem statement these devices should have picture capturing and audio playing capabilities. As the name suggests this device does not have compute capabilities and relies on the network of compute nodes in order to get the final audio file for the image it publishes on the network. The non-compute IoT devices have been simulated on the Google Cloud Platform.

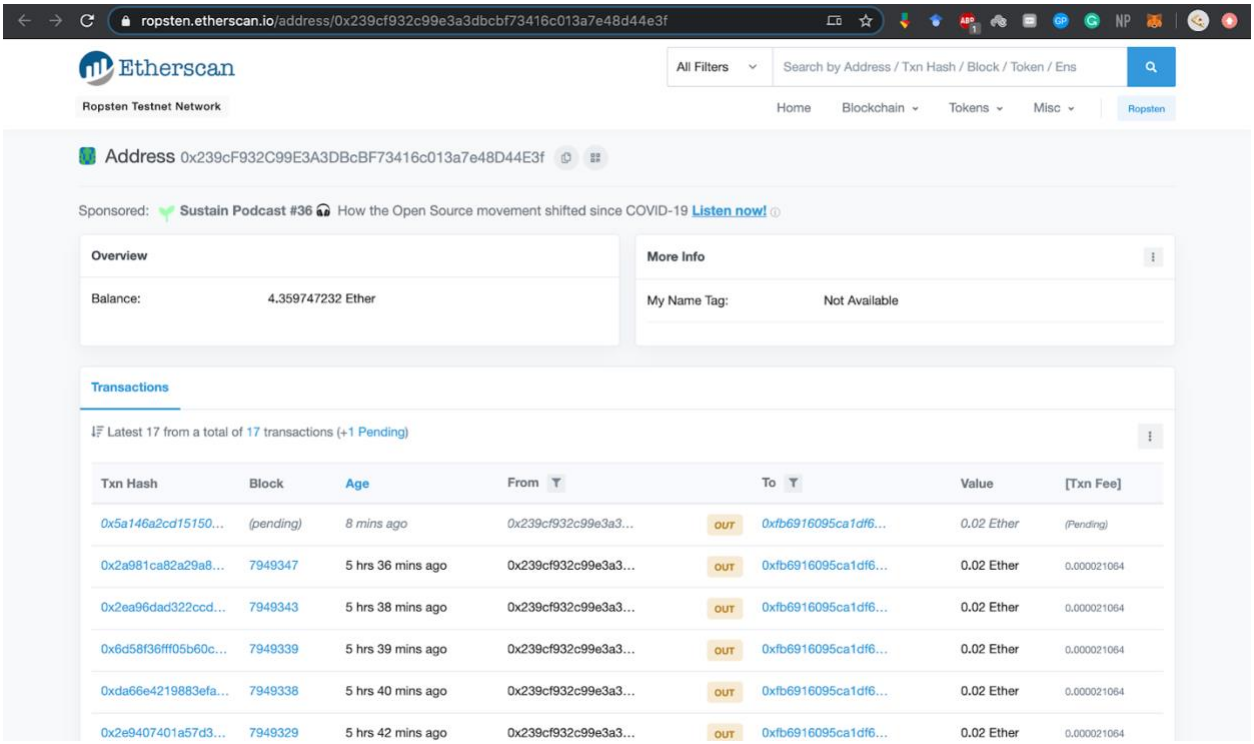
Rekognition Compute Nodes: This node has also been simulated using the Google Cloud Platform IoT Core service. The blue nodes that have been placed vertically on the left-hand

side represent the rekognition compute nodes in Fig 1. As the name suggests this node has compute capabilities. The non-compute node sends the images to this node and this node is tasked with running the image through the object detection model and returns the labels identified to the caller IoT device. This node receives the image over the MQTT queue and then stores it on the cloud. We have used AWS S3 to meet our storage needs. AWS also offers the Image Rekognition service which has a ML model pre-trained for object detection and this node uses this service on the image stored in the S3 bucket so as to get the labels in the image. The decision of which node serves which request has been made by a three-way handshake between the non-compute node and this node, such that the first node that provides an acknowledgement to the non-compute device is tasked with serving the request. Details of this communication have been presented later in the part where the MQTT communication is discussed.

Polly Compute Nodes: This node has also been simulated using the Google Cloud Platform IoT Core service. As the name suggests this node has compute capabilities. The blue nodes that have been placed horizontally at the bottom represent the polly compute nodes in Fig 1. The non-compute node sends the labels that it has received from the Rekognition Compute node to this node and this node is tasked with running the labels through the text-to-speech-model and an audio file that reads out the labels is sent to the caller IoT device. This node receives the image over the MQTT queue and then stores it on the cloud. We have used AWS S3 to meet our storage needs. AWS also offers the Image Rekognition service which has a ML model pre-trained for object detection and this node uses this service on the image stored in the S3 bucket so as to get the labels in the image. These labels are again returned using the MQTT queue. The decision of which node serves which request has been made by a three-way handshake between the non-compute node and this node, such that the first node that provides an acknowledgement to the non-compute device is tasked with serving the request. Details of this communication have been presented later in the part where the MQTT communication is discussed.

DApp Node: Our architecture currently supports a single DApp Node. The single IoT device present in the rightmost coroner of Fig 1 is the DApp node in the design diagram. This node is used for authentication before giving the device access to the MQTT central queue. For our project, we have used the Ropsten test network where each device has a wallet with address and private keys. The device gets access to the central queue as soon as a payment of 0.02 ether has been made via the “pay()” function in the smart contract pushed to the Ropsten test network. This is reflected in the Metamask wallet and can be seen on Etherscan as well. The user interface of the same can be seen in the figure below.


Not Secure | remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.6.6+commit....
DEPLOY & RUN TRANSACTIONS
CONTRACT
ImagePayment - browser/Payment.s
Deploy
PUBLISH TO IPFS
OR
At Address
Load contract from Address
Transactions recorded 6
Deployed Contracts
IMAGEPAYMENT AT 0X998...BF511 (BLOCKCHAIN)
drain
pay
minPayment
payments
address
Low level interactions
CALLDATA
Transact
Home
Payment.sol
pragma solidity 0.6.6;
contract ImagePayment {
constructor() public{
owner = msg.sender;
}
mapping (address => uint) public payments;
address payable owner;
uint public minPayment = 1 ** 16;
event PaymentReceived(
address indexed from,
uint256 indexed amount
);
function pay() external payable {
require(msg.value > minPayment, "Amount less than
payments[msg.sender] = payments[msg.sender] + msg
emit PaymentReceived(msg.sender, msg.value);
}
function drain() external {
require(msg.sender == owner);
owner.transfer(address(this).balance);
}
}
Ropsten Test Network
My Accounts
Log out
user1
5 ETH
user1
4.359747 ETH
user2
4.499895 ETH
user3
4.799789 ETH
Send
Create Account
Import Account
Connect Hardware Wallet
Info & Help
Settings
Deposit
1 ETH
listen on network
Search with transaction hash or address


ropsten.etherscan.io/address/0x239cf932c99e3a3dbcbf73416c013a7e48d44e3f
Etherscan
Ropsten Testnet Network
All Filters
Search by Address / Txn Hash / Block / Token / Ens
Home
Blockchain
Tokens
Misc
Ropsten
Address 0x239cF932C99E3A3DBcBF73416c013a7e48D44E3f
Sponsored: Sustain Podcast #36 How the Open Source movement shifted since COVID-19 Listen now!
Overview
Balance: 4.359747232 Ether
More Info
My Name Tag: Not Available
Transactions
Latest 17 from a total of 17 transactions (+1 Pending)
Txn Hash
Block
Age
From
To
Value
[Txn Fee]
0x5a146a2cd15150...
(pending)
8 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
(Pending)
0x2a981ca82a29a8...
7949347
5 hrs 36 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
0.000021064
0x2ea96dad322ccd...
7949343
5 hrs 38 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
0.000021064
0x6d58f36fff05b60c...
7949339
5 hrs 39 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
0.000021064
0xda66e4219883efa...
7949338
5 hrs 40 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
0.000021064
0x2e9407401a57d3...
7949329
5 hrs 42 mins ago
0x239cf932c99e3a3...
OUT
0xfb6916095ca1df6...
0.02 Ether
0.000021064

MQTT Publish/Subscribe: The devices interact between themselves using the MQTT Pub/Sub model. GCP provides a MQTT bridge for registries and we made use of that. Every device in the network (except DApp) is subscribed to the central topic ('projects/project2-277316/topics/my-topic' in our case). The compute nodes read messages from this queue and act on them only if it is a request that can be handled by it. The communication from the compute node to the non-compute device is more secure however, since the compute nodes publish the message directly to the device's config which is private to the device. The direction of communications between devices and nodes has been shown in Fig1 through arrows. As can be seen only the non-compute nodes publish information to the central topic, while the compute nodes only read from this topic. The interaction from the compute node to the non-compute device is done to the device's config directly.
An additional benefit of using a central queue across all services is that this makes the architecture very portable and easy to modify in case a new service has to be added at a later development stage.

Our solution covers all the breadth requirements of being decentralized (P2P), having IoT, Machine Learning, multiple clouds and using DApp (Blockchain). An explanation of how each one of these has been incorporated in the design has been provided below:

Decentralized (P2P) component: In order to ensure that our design has decentralized peers and compute capabilities we have tweaked the way we publish and acknowledge requests from the MQTT queue. Fig2 explains how the P2P status has been ensured.

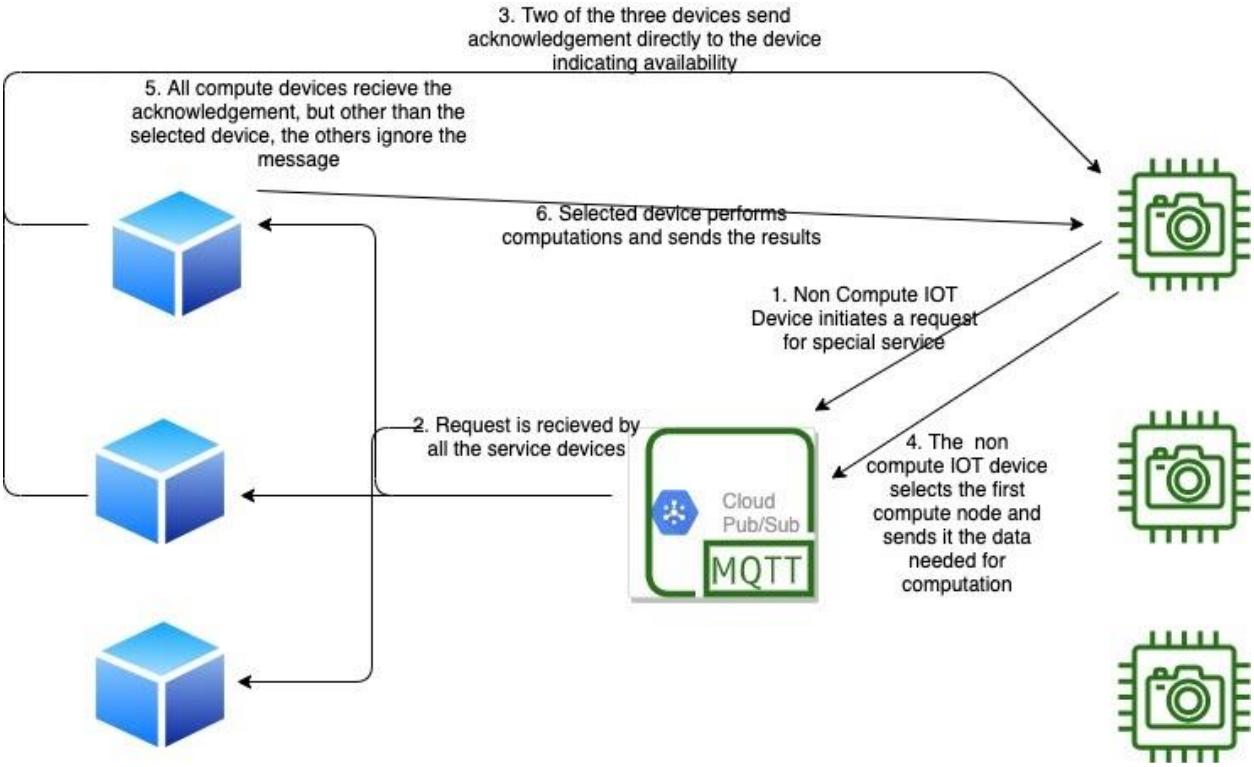


Fig2: Three-way handshake to simulate P2P architecture

IoT component: As explained above, all the devices and nodes have been simulated as IoT devices on GCP, hence covering this requirement.
Machine Learning component: Both Rekognition and Polly use ML models in order to detect objects and convert text to speech, hence covering this requirement

Multiple Clouds: While all our devices and nodes are in GCP, the ML support for the architecture has been taken from AWS Polly and Rekognition services, hence covering this breadth

DApp component: The DApp server which is a part of the IoT core in GCP, utilizes the Web3 python library to interact with the smart contract pushed on the Ropsten Network with the help of its address and ABI.

Implementation Details

The code is present on the GitHub link:
https://github.com/satyajeetmaharana/Image-Captioning-IoT-Google-Cloud.

The instructions to execute the code have been written in the Readme.md on GitHub.

Limitations of the solution

a. Fault Tolerance

The architecture that we have right now is not fault tolerant. The case where a selected compute node could fall out of the network due to failure needs to be handled. One proposed solution would be to keep track of the time when a request was made and after a pre-selected threshold, assume that the compute node is non-responsive and submit the initial request again.

b. Security

Our architecture is not secure enough to share sensitive information over the network. One solution that we would have explored if time allowed us to do it would be to encrypt and hash the data with a private and public key which would make sure that only the parties involved in the transaction are able to read the information published.