Collecting Cartography Scan statistics

AWS

Task Requirements

- Examine the Cartography repository and explore what it does along with neo4j
- Explore the AWS services within in depth
- Collect statistics for scans of each sync within AWS present in the repository Like shown below for s3 and RDS.

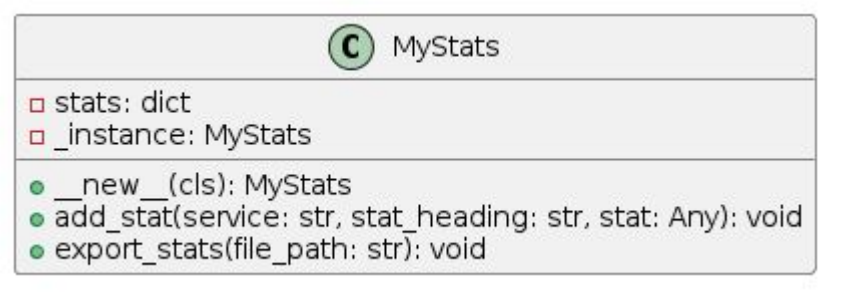
Service	Time Taken	Total Resources Scanned	Region Skipped	Errors Occurred	Status
S3	5 sec	23	us-east-1	AccessDenied	completed
RDS	10 sec	24	none	none	failed

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

General Approach

- Make a Singleton class for collecting stats through every scan
- Store a dictionary in this class which we will edit using access methods
- At the end of all the scans, store the dictionary in a json file to store details after termination of code

cartography/my_stats.py



https://github.com/Maaz-24503/Maaz_Carto_Internship.git

cartography/my_stats.py

Muhammad Maaz Karim

```python
def add_stat(self, service: str, stat_heading: str, stat: Any) -> None:
    if service not in self.stats:
        self.stats[service] = {}
    if "errors" not in self.stats[service]:
        self.stats[service]["errors"] = set()
    if "skipped regions" not in self.stats[service]:
        self.stats[service]["skipped regions"] = set()
    if stat_heading == "errors":
        self.stats[service][stat_heading].add(stat)
    elif stat_heading == "skipped regions":
        self.stats[service][stat_heading].add(stat)
    else:
        self.stats[service][stat_heading] = stat
```

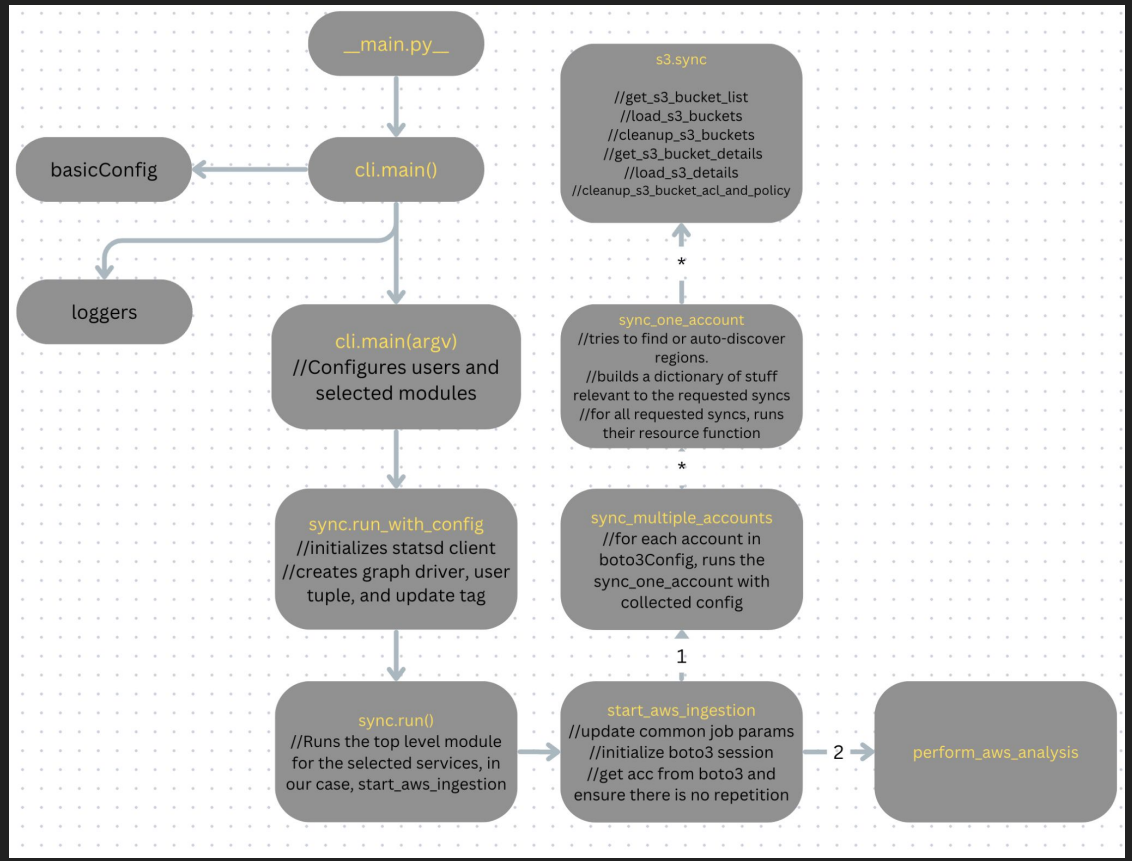
2 usages Muhammad Maaz Karim

```python
def export_stats(self, file_path: str) -> None:
    """
    Exports the stats dictionary to a JSON file.

    :param file_path: The path to the file where the JSON should be saved.
    """

    for service in self.stats:
        self.stats[service]["errors"] = list(self.stats[service]["errors"])
        self.stats[service]["skipped regions"] = list(self.stats[service]["skipped regions"])
    print("Exporting stats")
    with open(file_path, 'w') as json_file:
        json.dump(self.stats, json_file, indent=4)

    print("Exported stats")
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Taking a deeper look into one service

- To get a better understanding of where to extract our stats from, we need to get a clear view of what is happening
- This is a birds eye view of the flow of execution when running an s3 scan



https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Time Taken

cartography/intel/aws/__init__.py/_sync_one_account

- To measure the time taken for each sync, we can simply start a timer before the execution of each RESOURCE_FUNCTION, stop it after, and store the time consumed in our stats instance.

```python
for func_name in aws_requested_syncs:

    begin = time.time()  # Added by Maaz

    if func_name in RESOURCE_FUNCTIONS:
        # Skip permission relationships and tags for now because they rely on data already being in the graph
        if func_name not in ['permission_relationships', 'resourcegroupstaggingapi']:
            try:
                statistician.add_stat(func_name, stat_heading: "status", stat: "successful")
                RESOURCE_FUNCTIONS[func_name](**sync_args)
            except Exception as e:
                statistician.add_stat(func_name, stat_heading: "status", stat: "failed")
        else:
            continue
    else:
        raise ValueError(f'AWS sync function "{func_name}" was specified but does not exist. Did you misspell it?')

    end = time.time()   # Added by Maaz
    statistician.add_stat(func_name, stat_heading: "Time Taken", stat: f'{round(end-begin)} seconds')  # Added by Maaz



# MAP IAM permissions
if 'permission_relationships' in aws_requested_syncs:
    begin = time.time()  # Added by Maaz
    RESOURCE_FUNCTIONS['permission_relationships'](**sync_args)
    end = time.time()  # Added by Maaz
    statistician.add_stat( service: 'permission_relationships', stat_heading: "status", stat: "successful")
    statistician.add_stat( service: 'permission_relationships', stat_heading: "Time Taken", stat: f'{round(end - begin)} seconds')
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Total Resources Scanned

- In the case of s3, we will have to count the number of buckets that have been fetched in the sync function
- This statistic will be differently stored for different syncs in ways that is relevant, eg. ssm will separately have instance and patches scanned by region

cartography/intel/aws/s3.py/sync

```
Muhammad Maaz Karim
@timeit
def sync(
    neo4j_session: neo4j.Session, boto3_session: boto3.session.Session, regions: List[str], curre
    update_tag: int, common_job_parameters: Dict,
) -> None:
    logger.info( msg: "Syncing S3 for account '%s'.",  *args: current_aws_account_id)
    bucket_data = get_s3_bucket_list(boto3_session)

    total_resources = len(bucket_data["Buckets"])    # Added by Maaz
    statistician = MyStats()      # Added by Maaz
    statistician.add_stat( service: "s3",  stat_heading: "Total Resources Scanned", total_resources)

    load_s3_buckets(neo4j_session, bucket_data, current_aws_account_id, update_tag)
    cleanup_s3_buckets(neo4j_session, common_job_parameters)

    acl_and_policy_data_iter = get_s3_bucket_details(boto3_session, bucket_data)
    load_s3_details(neo4j_session, acl_and_policy_data_iter, current_aws_account_id, update_tag)
    cleanup_s3_bucket_acl_and_policy(neo4j_session, common_job_parameters)
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Regions Skipped

- In this, we must make the observation that if an error is encountered during a scan, that region is dumped altogether, so we must record this stat in the error handling function.

cartography/intel/aws/s3.py/_is_common_exception

```
6 usages   Muhammad Maaz Karim
@timeit
def _is_common_exception(e: Exception, bucket: Dict) -> bool:
    statistician = MyStats()    # Added by Maaz
    error_msg = "Failed to retrieve S3 bucket detail"
    if bucket['Region'] is not None:
        statistician.add_stat( service: "s3",  stat_heading: "skipped regions", bucket['Region'])
    # with open("bucket_check.json", 'w') as json_file:
    #     json.dump(bucket, json_file, indent=4)
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Errors

- We simply store this in our stats every time an error is caught during a sync
- For s3, this can be handled in the "_is_common_exception" function

cartography/intel/aws/s3.py/_is_common_exception

```
if "AccessDenied" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "Access Denied")  # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - Access Denied")
    return True
elif "NoSuchBucketPolicy" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "No Such Bucket Policy")  # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - NoSuchBucketPolicy")
    return True
elif "NoSuchBucket" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "No Such Bucket") # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - No Such Bucket")
    return True
elif "AllAccessDisabled" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "All Access Disabled")    # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - Bucket is disabled")
    return True
elif "EndpointConnectionError" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "Endpoint Connection Error")  # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - EndpointConnectionError")
    return True
elif "ServerSideEncryptionConfigurationNotFoundError" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "ServerSide Encryption Configuration Not Found Error")
    logger.warning(f"{error_msg} for {bucket['Name']} - ServerSideEncryptionConfigurationNotFoundError")
    return True
elif "InvalidToken" in e.args[0]:
    statistician.add_stat( service: 's3', stat_heading: "errors", stat: "Invalid Token")  # Added by Maaz
    logger.warning(f"{error_msg} for {bucket['Name']} - InvalidToken")
    return True
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Status

cartography/intel/aws/__init__.py/_sync_one_account

- If a resource function is run in the try block and the execution reaches the line below it, it means the scan was completed, we can use this to save whether or not the scan was completed

```
for func_name in aws_requested_syncs:

    begin = time.time()  # Added by Maaz

    if func_name in RESOURCE_FUNCTIONS:
        # Skip permission relationships and tags for now because they rely on data already being in the graph
        if func_name not in ['permission_relationships', 'resourcegroupstaggingapi']:
            try:
                statistician.add_stat(func_name, stat_heading: "status", stat: "successful")
                RESOURCE_FUNCTIONS[func_name](**sync_args)
            except Exception as e:
                statistician.add_stat(func_name, stat_heading: "status", stat: "failed")
        else:
            continue
    else:
        raise ValueError(f'AWS sync function "{func_name}" was specified but does not exist. Did you misspell it?')

    end = time.time()   # Added by Maaz
    statistician.add_stat(func_name, stat_heading: "Time Taken", stat: f'{round(end-begin)} seconds')  # Added by Maaz
```

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

Storing stats into a file

- We call the export_stats function after the completion of all the scans since all the stats are now stored in the instance of our MyStats class, which we need to dump into a json file to have access after termination of the code

cartography/sync.py/run_with_config

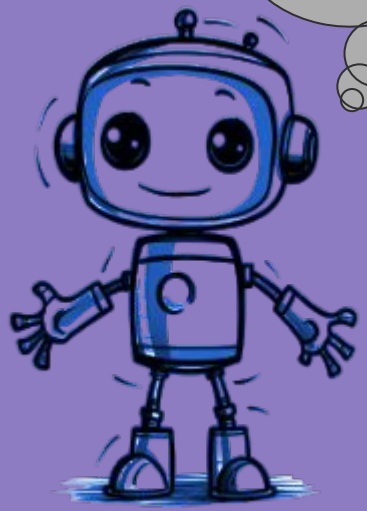
```
        }
        return STATUS_FAILURE
    default_update_tag = int(time.time())
    if not config.update_tag:
        config.update_tag = default_update_tag
    temp = sync.run(neo4j_driver, config)

    MyStats().export_stats("cartography/statistics_file.json")
    return temp
```

Results stored in cartography/statistics_file.json

https://github.com/Maaz-24503/Maaz_Carto_Internship.git

voilà

And just like that, we have the required stats from our AWS scan



https://github.com/Maaz-24503/Maaz_Carto_Internship/blob/main/cartography/statistics_file.json


```
{
    "iam": {
        "errors": [],
        "skipped regions": [],
        "status": "successful",
        "Total Users Scanned": 260,
        "Total Groups Scanned": 2,
        "Total Roles Scanned": 995,
        "Time Taken": "4233 seconds"
    },
    "s3": {
        "errors": [
            "No Such Bucket Policy",
            "Access Denied"
        ],
        "skipped regions": [
            "us-west-2",
            "us-east-2"
        ],
        "status": "successful",
        "Total Resources Scanned": 101,
        "Time Taken": "85 seconds"
    },
```


https://github.com/Maaz-24503/Maaz_Carto_Internship.git