Amazon EMR RedFin

1. Launch EC2 Instance → Name (redFin-EC2) → Choose Ubuntu Image → t2.medium (Instance Type) → Create Key Pair (redFin_key_pair), key pair type (RSA), format(.pem) → Create Key → Create Security Group → Launch instance
2. (Access Keys) Go to Account → Security Credentials → Create Access Key → aws configure → Enter Access Key, Secret Access Key, Default Region
3. Installing Dependencies → python version check, sudo apt update, sudo apt install python3-pip, sudo apt install python3.10-venv → python3 -m venv redfin_venv (Create Virtual Env), source redfin_venv/bin/activate (Activate) → pip install boto3, pip install --upgrade awscli (Dependencies) → pip install apache-airflow, pip install apache-airflow-providers-amazon → airflow version (Check the version), airflow standalone (Initialize)
4. Use the public IPv4 address with 8080 port for airflow UI → In order to ensure that the UI loads the page, we need to open the ports under Security → Move to Security Groups → Edit the Inbound Rules → Add Rule → Custom TCP - 8080 - Anywhere IPv4 → Save Rules → Refresh the Airflow Page → Add credentials
5. {Optional} {Remotely SSH to VS Code} Click on the instance → Connect → SSH Client → Copy command starting with ssh → Open CMD → Paste command and choose yes
6. {Optional} Go to VS Code → Extensions → Remote SSH → Install → Click on the bottom left icon → Connect to Host → Configure SSH Hosts → Select the one with Users → Add the following and save it there → Now reconnect with the icon → Choose the desired host → Platform (Linux) → Verify if the connection is working
7. In the airflow.cfg file, there is a path for dags folder which we have to create → Create the dags folder and a file named zillow_analytics.py → Also set the load_examples to False and Refresh the server

8. Navigate to S3 Bucket (redfin-data-yml) → Choose correct region and then create folders named (store-raw-data-yml, scripts, redfin-transform-zoneyml, emr-logs-yml)
9. Connect SSH to VS Code and then Open Folder to /home/ubuntu → Move to airflow and create new folder named dags → Create new file (redfin_analytics.py)
10. Choose and create VPC → Name (redfin_emr_airflow) → No. of Private Subnets (0) with Endpoints (S3Gateway) → Create VPC → Copy SubnetID to redfin_analytics.py file under EC2SubnetID
11. Move to Key Pairs → Create Key Pair → Name (emr-keypair-airflow) → Keypair type (RSA) and format (.pem) → Check for JobFlowRole within the EC2 instance terminal → source redfin_venv/bin/activate
12. Within the terminal, we need to check for aws iam list-roles | grep 'EMR_DefaultRole\|EMR_EC2_DefaultRole' and if returns None then enter aws emr create-default-roles which will create those IAM roles.
13. Now go to Airflow UI → Check for DAG → Trigger the DAG (Check for Errors in Logs) → job_flow_id for (is_emr_cluster_created) comes from Airflow DAG → Admins → Xcom → Cluster ID (return value)
14. Create ingest.sh and transform_redfin_data.py files and upload it within the scripts folder in the S3 bucket.
15. Save the redfin_analytics file and view the DAG diagram once again before triggering the pipeline, also debug the results of the pipeline regularly for errors.
16. Log on to our Snowflake Account → Open New SQL Worksheet → Name (real_estate_snowpipe.sql) → Write the logic for exporting data from S3 to Snowflake via Snowpipe.
17. Move to S3 redfin-transform-zone-yml folder → Properties → Event Notifications → Create notification → (redfin-snowpipe-event), folder target, check all object create events → Destination SQS Queue → Enter SQS Queue ARN from Snowflake → Save changes → Debug the pipeline
18. Open PowerBI → Home → Get Data → Snowflake → Server and Warehouse from Snowflake → Under Snowflake, Account → Copy URL for Server,

Warehouse name for Warehouse → Enter Username, Password → Connect & Load Data → Create a dashboard accordingly for our analysis.