**Here are three best practices used during the implementation of this data pipeline**:

1. **Encapsulation of the code within functions**: The extract, transform, and load (ETL) steps have been implemented as separate functions. This is a good practice as it makes the code more modular, reusable, and easier to maintain. Each function performs a specific task, and the inputs and outputs of each function are well-defined.
2. **Use of try/except block**: The code that loads the data into the database has been wrapped within a try/except block. This is a good practice as it catches any errors that might occur during the loading process and logs an error message. The use of logging also helps to identify the source of the error quickly, which makes debugging easier.
3. **Use of Airflow DAG**: The data pipeline has been implemented as an Airflow DAG. This is a good practice as it enables the automation of the ETL process and the scheduling of the pipeline to run at specific intervals. Airflow DAG also provides monitoring and alerting capabilities, which can help to identify issues with the pipeline in real-time

**Given that Airflow will be used, here are some recommendations for deploying and running the pipeline on a cloud-based provider:**

1. Choose a cloud provider: The first step is to choose a cloud provider that can support Airflow. Some popular options are Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.
2. Create an Airflow environment: Once you have chosen a cloud provider, create an Airflow environment that is scalable and fault-tolerant. You can use tools like Kubernetes or Docker Swarm to manage your Airflow environment.
3. Configure the database: Airflow uses a database to store metadata about DAGs, tasks, and task instances. You will need to configure the database to run on a cloud-based provider. You can use a managed database service like Amazon RDS or Google Cloud SQL, or you can set up a self-managed database like PostgreSQL.
4. Store data on cloud storage: You will need to store your data on a cloud-based storage system like Amazon S3 or Google Cloud Storage. This will allow your pipeline to access data from anywhere and make it easy to manage and store large datasets.
5. Configure logging: Configure Airflow logging to send logs to a centralized logging service like Amazon CloudWatch or Google Cloud Logging. This will make it easier to monitor your pipeline and identify any errors or issues.
6. Secure your environment: Make sure your Airflow environment is secured with the appropriate security measures, including role-based access control, encryption, and network security.
7. Automate deployment: Use tools like Terraform or CloudFormation to automate the deployment of your Airflow environment. This will make it easier to manage and scale your pipeline over time.
8. Monitor performance: Set up monitoring and alerting for your pipeline to ensure it is running smoothly. You can use tools like Prometheus or Grafana to monitor performance metrics like task duration, CPU usage, and memory usage.

Overall, deploying and running an Airflow pipeline on a cloud-based provider requires careful planning and consideration. By following these recommendations, you can ensure your pipeline is scalable, fault-tolerant, and secure, and that it can handle large volumes of data and run reliably over time.