



Protegrity Anonymization API Guide 1.1.0.0

Created on: Nov 19, 2024

Copyright

Copyright © 2004-2024 Protegrity Corporation. All rights reserved.

Protegrity products are protected by and subject to patent protections;

Patent: <https://www.protegrity.com/patents>.

Protegrity logo is the trademark of Protegrity Corporation.

NOTICE TO ALL PERSONS RECEIVING THIS DOCUMENT

Some of the product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective owners.

Windows, Azure, MS-SQL Server, Internet Explorer and Internet Explorer logo, Active Directory, and Hyper-V are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SCO and SCO UnixWare are registered trademarks of The SCO Group.

Sun, Oracle, Java, and Solaris are the registered trademarks of Oracle Corporation and/or its affiliates in the United States and other countries.

Teradata and the Teradata logo are the trademarks or registered trademarks of Teradata Corporation or its affiliates in the United States and other countries.

Hadoop or Apache Hadoop, Hadoop elephant logo, Hive, and Pig are trademarks of Apache Software Foundation.

Cloudera and the Cloudera logo are trademarks of Cloudera and its suppliers or licensors.

Hortonworks and the Hortonworks logo are the trademarks of Hortonworks, Inc. in the United States and other countries.

Greenplum Database is the registered trademark of VMware Corporation in the U.S. and other countries.

Pivotal HD is the registered trademark of Pivotal, Inc. in the U.S. and other countries.

PostgreSQL or Postgres is the copyright of The PostgreSQL Global Development Group and The Regents of the University of California.

AIX, DB2, IBM and the IBM logo, and z/OS are registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Utimaco Safeware AG is a member of the Sophos Group.

Xen, XenServer, and Xen Source are trademarks or registered trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

VMware, the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

Amazon Web Services (AWS) and AWS Marks are the registered trademarks of Amazon.com, Inc. in the United States and other countries.

HP is a registered trademark of the Hewlett-Packard Company.

HPE Ezmeral Data Fabric is the trademark of Hewlett Packard Enterprise in the United States and other countries.

Dell is a registered trademark of Dell Inc.

Novell is a registered trademark of Novell, Inc. in the United States and other countries.

POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.

Mozilla and Firefox are registered trademarks of Mozilla foundation.

Chrome and Google Cloud Platform (GCP) are registered trademarks of Google Inc.

Table of Contents

Copyright.....	2
Chapter 1 Introduction.....	6
1.1 Business Cases.....	6
1.2 Data Security and Data Privacy.....	7
1.2.1 Pseudonymization.....	7
1.2.2 Anonymization.....	8
1.3 Importance and Types of Data.....	8
1.4 Types of Privacy Models.....	9
Chapter 2 About Protegrity Anonymization API.....	10
2.1 How Protegrity Anonymization API Works.....	10
2.2 Understanding the Protegrity Anonymization API Components.....	12
2.3 Communication Between the Components.....	12
Chapter 3 Installing the Protegrity Anonymization API.....	14
3.1 Installing the Protegrity Anonymization REST API.....	14
3.1.1 Prerequisites for Deploying the Protegrity Anonymization API.....	14
3.1.2 Using Cloud Services.....	15
3.1.2.1 Anonymizing Using Amazon Elastic Kubernetes Service (EKS).....	15
3.1.2.2 Anonymizing Using Azure Kubernetes Service (AKS).....	29
3.2 Installing Using Docker Containers.....	38
3.3 Installing the Protegrity Anonymization SDK.....	41
3.3.1 Prerequisites for Deploying the Protegrity Anonymization SDK.....	41
3.3.2 Installing the Protegrity Anonymization SDK Wheel File.....	41
3.3.3 Enabling Custom Certificates From SDK.....	41
3.3.4 Verifying the Installation.....	43
Chapter 4 Using Protegrity Anonymization API.....	44
4.1 Creating Protegrity Anonymization API Requests.....	44
4.2 Working with the Protegrity Anonymization APIs.....	45
4.2.1 Understanding the Protegrity Anonymization REST APIs.....	45
4.2.1.1 Anonymization Functions.....	45
4.2.1.2 Task Monitoring APIs.....	48
4.2.1.3 Statistics APIs.....	52
4.2.1.4 Detection APIs.....	56
4.2.2 Understanding the Protegrity Anonymization SDK.....	60
4.2.2.1 Understanding the AnonElement object.....	60
4.2.2.2 Anonymization Functions.....	60
4.2.2.3 Task Monitoring APIs.....	63
4.2.2.4 Statistics APIs.....	65
4.3 Building the Anonymization request.....	68
4.3.1 Building the request using REST.....	68
4.3.1.1 Identifying the Source and Target.....	68
4.3.1.2 Specifying the Transformation.....	69
4.3.2 Building the request using SDK.....	76
4.3.2.1 Creating the Connection.....	77
4.3.2.2 Identifying the Source and Target.....	77
4.3.2.3 Specifying the Transformation.....	79
4.3.2.4 Working with Saved Anonymization Requests.....	83
4.3.2.5 Running a Sample Request.....	84
4.4 Using the Workstation Feature.....	86
4.4.1 Using the Kubernetes Workstation.....	86
4.4.2 Using the Docker Workstation.....	87

4.4.3 Using the Jupyter Notebook.....	87
Chapter 5 Using the Auto Anonymizer.....	89
5.1 Using <i>mode</i> to Auto Anonymize.....	91
5.2 Using Infer to Anonymize.....	92
Chapter 6 Appendix.....	95
6.1 Best Practices When Using the Protegrity Anonymization API.....	95
6.2 Configuring Amazon SageMaker for the Protegrity Anonymization API.....	96
6.3 Samples for Cloud-related Source and Destination Files.....	99
6.4 YAML Templates.....	101
6.4.1 cluster-aws.yaml.....	101
6.4.1.1 Cluster Metadata.....	102
6.4.1.2 Cloud Provider Details.....	102
6.4.1.3 Node Group Details.....	102
6.4.2 values.yaml.....	103
6.4.2.1 Basic Configuration.....	106
6.4.2.2 Image Configuration.....	107
6.4.2.3 REST Service Configuration.....	107
6.4.2.4 Health Check Configuration.....	108
6.4.2.5 Worker Pod Configuration.....	108
6.4.2.6 Storage Configuration.....	109
6.4.2.7 Database Configuration.....	109
6.4.3 docker-compose.yaml.....	110
6.5 Configuration Checklist.....	111
6.5.1 AWS Checklist.....	111
6.6 Setting Up Logging for the Protegrity Anonymization API.....	112
6.7 Creating a DNS Entry for the ELB Hostname in Route53.....	113
6.8 Protegrity Anonymization API Risk Metrics.....	113
6.8.1 Definitions.....	113
6.8.2 Types of Risks.....	114
6.8.3 Measuring Risks.....	114
6.8.3.1 Measuring Prosecutor Risk.....	115
6.8.3.2 Measuring Journalist Risk.....	117
6.8.3.3 Measuring Marketer Risk.....	118
6.9 Sample Data Sets.....	118
6.10 Sample Requests for Protegrity Anonymization REST API	120
6.10.1 Tree-based Aggregation for Attributes with k-Anonymity.....	120
6.10.2 Tree-based Aggregation for Attributes with k-Anonymity, l-Diversity, and t-Closeness.....	121
6.10.3 Micro-Aggregation and Generalization with Aggregates.....	123
6.10.4 Parquet File Format.....	125
6.10.5 Retaining and Redacting.....	128
6.11 Sample Requests for Protegrity Anonymization SDK.....	130
6.11.1 Tree-based Aggregation for Attributes with k-Anonymity.....	130
6.11.2 Tree-based Aggregation for Attributes with k-Anonymity, l-Diversity, and t-Closeness.....	131
6.11.3 Micro-Aggregation and Generalization with Aggregates.....	132
6.11.4 Retaining and Redacting.....	133
6.12 Troubleshooting.....	134
6.12.1 Known Issues.....	134
6.12.2 Limitations.....	136
6.12.3 Working with Certificates.....	137

Chapter 1

Introduction

1.1 Business Cases

1.2 Data Security and Data Privacy

1.3 Importance and Types of Data

1.4 Types of Privacy Models

Today's organizations collect copious amounts of personal data on data subjects. This personal information identifies data subjects and provides information about their habits, purchasing trends, health, and their likes and dislikes. The information about how the data subjects carry out their activities is integral to a business. It can help an organization formulate policies and products that put it on the path of success and increased profits. Much of this information, however, is considered highly sensitive and private. Thus, unless there is a specific requirement for personally identifiable information, it is in the organization's best interests to protect the identity of the data subject. Key identifiers could be pseudonymized (using encryption or tokenization), fully redacted, or modified in such a way that it is still useful for analytics, while no longer identifying the specific data subject. For example, the address could be generalized to list the city, state, country, or even continent. Through this generalization, the individual may no longer be specifically identifiable, but the data retains value for trend analysis.

1.1 Business Cases

Consider the following business cases:

- **Case 1:** A hospital wants to share patient data with a third-party research lab. The privacy of the patient, however, must be preserved.
- **Case 2:** An organization requires customer data from several credit unions to create training data. The data will be used to train Machine Learning models looking for new insights. The customers, however, have not agreed for their data to be used.
- **Case 3:** An organization which must be compliant with GDPR, CCPA, or other privacy regulations requires to keep some information beyond the period that meets regulations.
- **Case 4:** An organization requires raw data to train their software for Machine Learning.

In all these cases, data forms an integral part of the source for continuing the business process or analysis. Additionally, only *what was done* is required in all the cases, *who did it* does not have any value in the data. In this case, the personal information about the individual users can be removed from the data set. This removes the personal factor from the data and at the same time retains the value of the data from the business point of view. This data, since it does not have any private information, is also pulled from the legal requirements governing the data.

Thus, revisiting the cases, the data in each case can be valuable after processing it in the following ways:

- In case 1, all private information can be removed from the data and sent to the research lab for analysis.
- In case 2, all privacy data must be scrubbed from the data before the data can be used. After scrubbing, the data will be generalized in such a way that the data can be used for Machine Learning, and at the same time if a user has an issue, no one will be able to identify individuals in the source data set.
- In case 3, by anonymizing the data, the Data Subject is removed, and the data is no longer in scope for privacy compliance.

- In case 4, a generalized form of the data can be obtained. A data framework can also be built to help analyze scenarios.

Removing data manually to remove private information would take a lot of time and effort, especially if the data set consists of millions of records, with file sizes of 2 to 3 GBs. Running a find and replace or just deleting columns might remove important fields that might make the data set useless for further analysis. Additionally, a combination of remaining data (such as, date of birth, postcode, gender) may be enough to re-identify the data subject.

The Protegrity Anonymization API applies various privacy models to the data, removing direct identifiers and applying generalization to the remaining indirect identifiers, to ensure that no single data subject can be identified.

1.2 Data Security and Data Privacy

Most organizations understand the need to secure access to personally identifiable information. Sensitive values in records are often protected at rest (storage), in transit (network) and in use (fine-grained access control), through a process known as *de-identification*. De-Identification is a spectrum, where data security and data privacy issues must be balanced with data usability.

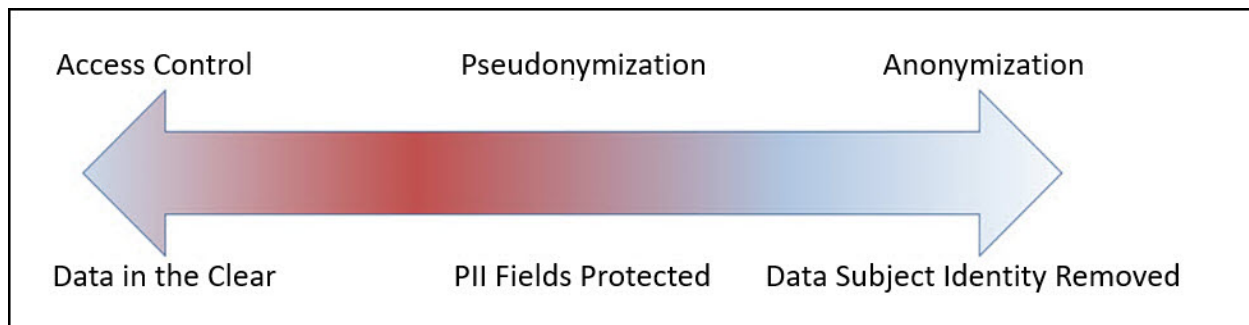


Figure 1-1: Data Protection Spectrum

1.2.1 Pseudonymization

Pseudonymization is the process of de-identification by substituting sensitive values with a consistent, non-sensitive value. This is most often accomplished through encryption, tokenization, or dynamic data masking. Access to the process for re-identification (decryption, detokenization, unmasking) is controlled, so that only users with a business requirement will see the sensitive values.

Advantages:

- The original data can be obtained again.
- It is a temporary process.
- Only authorized users can view the original data from tokenized data.
- It processes each record individually.
- This process is faster than anonymization.

Disadvantages

- The tokenized data cannot be used for analysis.
- The tokenized data is still private from the users perspective.
- Software processing is required to retrieve the data.
- Additional security is required to secure the data and the keys used for working with the data.

1.2.2 Anonymization

Anonymization is the process of de-identification which irreversibly redacts, aggregates, and generalizes identifiable information on all data subjects in the data set. This method ensures that while the data retains value for various use cases, analytics, data democratization, sharing with 3rd parties, and so on, the individual data subject can no longer be identified in the data set.

Advantages:

- The data can be used for analysis.
- An individual user cannot be identified from the anonymized data set.
- No approval is required from the user to use anonymized data.
- Any user can view the anonymized data that is generalized.
- It processes all the records together to obtain the anonymized data.
- Additional security is not required for the anonymized data.
- Data is not governed by GDPR
- Can be used by HIPAA

Disadvantages

- The original data cannot be obtained again.
- It is a permanent process.
- This process is slower than pseudonymization because multiple passes must be made on the set to anonymize it.

1.3 Importance and Types of Data

A record consists of all the information pertaining to a user. This record consists of different fields of information, such as, first name, last name, address, telephone number, age, and so on. These records might be linked with other records, such as, income statements or medical records to provide valuable information. The various fields as a whole, called a record, is private and is user-centric. However, the individual fields may or may not be personal. Accordingly, based on the privacy level, the following data classifications are available:

- **Direct Identifier:** Identity Attributes can identify an individual with the value alone. These attributes are unique to an individual in a data set and at times even in the world. It is personal and private to the user. For example, name, Social Security Number (SSN), mobile number, and so on.
- **Quasi-Identifier:** Quasi-Identifying Attributes are identifying characteristic about a data subject. However, you cannot identify an individual with the quasi-identifier alone. For example, date of birth or an address. Moreover, the individual pieces of data in a quasi-identifier might not be enough to identify a single individual. Take the example of date of birth, the year might be common to many individuals and would be difficult to narrow down to a single individual. However, if the data set is small, then it might be easy to identify an individual using this information.
- **Data about data subject:** Data about the data subject is typically the data that is being analyzed. This data might exist in the same table or a different related table of the data set. It provides valuable information about the data set and is very helpful for analysis. This data without supporting information might just be random data. This data may or might not be private to an individual. For example, salary, account balance, or credit limit. However, like quasi-identifiers, in a small data set, this data might be unique to an individual. Additionally, this data can be classified as follows:
 - **Sensitive Attributes:** This data may disclose something like a health condition which in a small result set may identify a single individual.
 - **Insensitive Attributes:** This data is not associated with a privacy risk and is common information, such as, the type of bank accounts in a bank, individual or business.

A sample data set is shown in the following figure:

First Name	Last Name	Address	City	State	E-Mail Address	SSN / NID	DOB	Account Balance	Credit Limit	Medical Code	Type
Jason	Watson	12 West St	Albany	NY	J.Watson@Company.com	392-11-4442	10/30/65	\$400,000	\$801,840	756	Individual
Kate	Harris	46 Main St	Trenton	NJ	K.Harris@Company.com	240-02-0023	12/28/47	\$381,116	\$279,128	502	Corporate
Donna	Lopez	6 Ash Lane	Westport	CT	D.Lopez@Company.com	480-65-4320	07/09/75	\$257,637	\$197,982	651	Corporate
Jon	Lewis	9 Sterling Av	Lompoc	CA	J.Lewis@Company.com	347-98-3305	11/26/84	\$750,000	\$14,453	521	Individual
Doris	Long	3 Blend Ln	Ashland	KY	D.Long@Company.com	209-33-8865	11/23/70	\$482,699	\$970,045	700	Individual
Jake	Flores	78 Main St	Bend	OR	J.Flores@Company.com	401-09-2310	10/09/62	\$151,012	\$456,820	830	Individual
Chuck	Howard	19 Rosa Lane	Miami	FL	C.Howard@Company.com	325-11-5631	09/04/65	\$161,562	\$408,627	537	Corporate
Larry	Clark	99 Ogden Rd	Waco	TX	L.Clark@Company.com	399-29-1192	08/23/65	\$5,295	\$100	700	Individual
Kate	Smith	56 Dover Ln	Mobile	AL	K.Smith@Company.com	389-65-2289	12/28/45	\$1,500,200	\$750,000	700	Corporate
Harry	Bell	45 Dent Av	Flint	MI	H.Bell@Company.com	349-55-1020	10/30/65	\$250,000	\$50,000	502	Individual
Jon	Chu	67 Harbor St	Butte	MT	J.Chu@Company.com	667-45-1123	07/30/70	\$100,000	\$10,000	756	Corporate

Figure 1-2: Sample Data Set

Based on the type of data, the columns in the above table can be classified as follows:

Type	Field Names	Description
Direct Identifier	First Name, Last Name, Address with city and state, E-Mail Address, SSN / NID	The data in these fields are enough to identify an individual.
Quasi-Identifier	City, State, Date of Birth	The data in these fields could be the same for more than one individual. Note: Address could be a direct identifier if a single individual is present from a particular state.
Sensitive Attribute	Account Balance, Credit Limit, Medical Code	The data is important for analysis, however, in a small data set it is easy to de-identify an individual.
Insensitive Attribute	Type	The data is general information making it difficult to de-identify an individual.

1.4 Types of Privacy Models

The privacy models are techniques for anonymizing data.

- **k-anonymity:** In this, data across the table is standardized in such a way that the same data seems applicable to k or more individuals in the data set. It is also known as "hiding in the crowd". For example, if a data set is k-anonymized to 5. Then, after anonymization processing, five records will be similar. If an attacker tries to re-identify data, then the same record would be applicable for five individuals, making it difficult to identify a single individual from the data set.
- **l-diversity:** In this, a k-anonymous data table is taken and the data that is generic in the entire table is broken down further. This process ensures that the data is not applicable to a specific group of individuals. The sensitive data is "well-represented" over multiple quasi-identifier classes enhancing the privacy of individuals in the data set.
- **t-closeness:** In this, the equivalence classes are identified using sensitive data and the data is kept "close" to these classes. An example of this is grouping people based on the period of birth, such as pre-1950, 1951-1980, 1981-2000, and post-2000.

Note: The l-diversity and t-closeness models extends the k-anonymity model to ensure better privacy over known attacks.

Chapter 2

About Protegrity Anonymization API

2.1 How Protegrity Anonymization API Works

2.2 Understanding the Protegrity Anonymization API Components

2.3 Communication Between the Components

Protegrity Anonymization API is a software created by Protegrity to scan your data set and provide a resultant data set that can be used for anonymization of various users and for analysis by other softwares. It generalizes all the personal data from the data set. If the data is too personal and cannot be generalized, then the data is removed or masked to secure the privacy of the users in the data set. Thus, at the end anonymized data that can be further used for analysis can be generated as the output. This data provides valuable information about the data without revealing the personal information about the user.

Protegrity Anonymization API can be hosted on Kubernetes. It uses pods to process data. Pods can be added when the workload increases and more processing is required. Since Protegrity Anonymization API is hosted on Kubernetes, it can be used for other cloud providers which support Kubernetes.

Note: Currently, Protegrity Anonymization API has been tested on EKS provided by AWS and AKS provided by Microsoft Azure. However, it can also be used for other cloud providers that support Kubernetes. You might need to modify the configuration and verify it for your cloud installation.

2.1 How Protegrity Anonymization API Works

In simple terms Protegrity Anonymization API can be described as a software that takes data for its input. It processes the data, removing personal information and summarizing or generalizing the remaining information. It then provides an output with the processed data that can be used for analysis, this output has generalized data using which individuals cannot be identified. The same can be seen in the following figure:

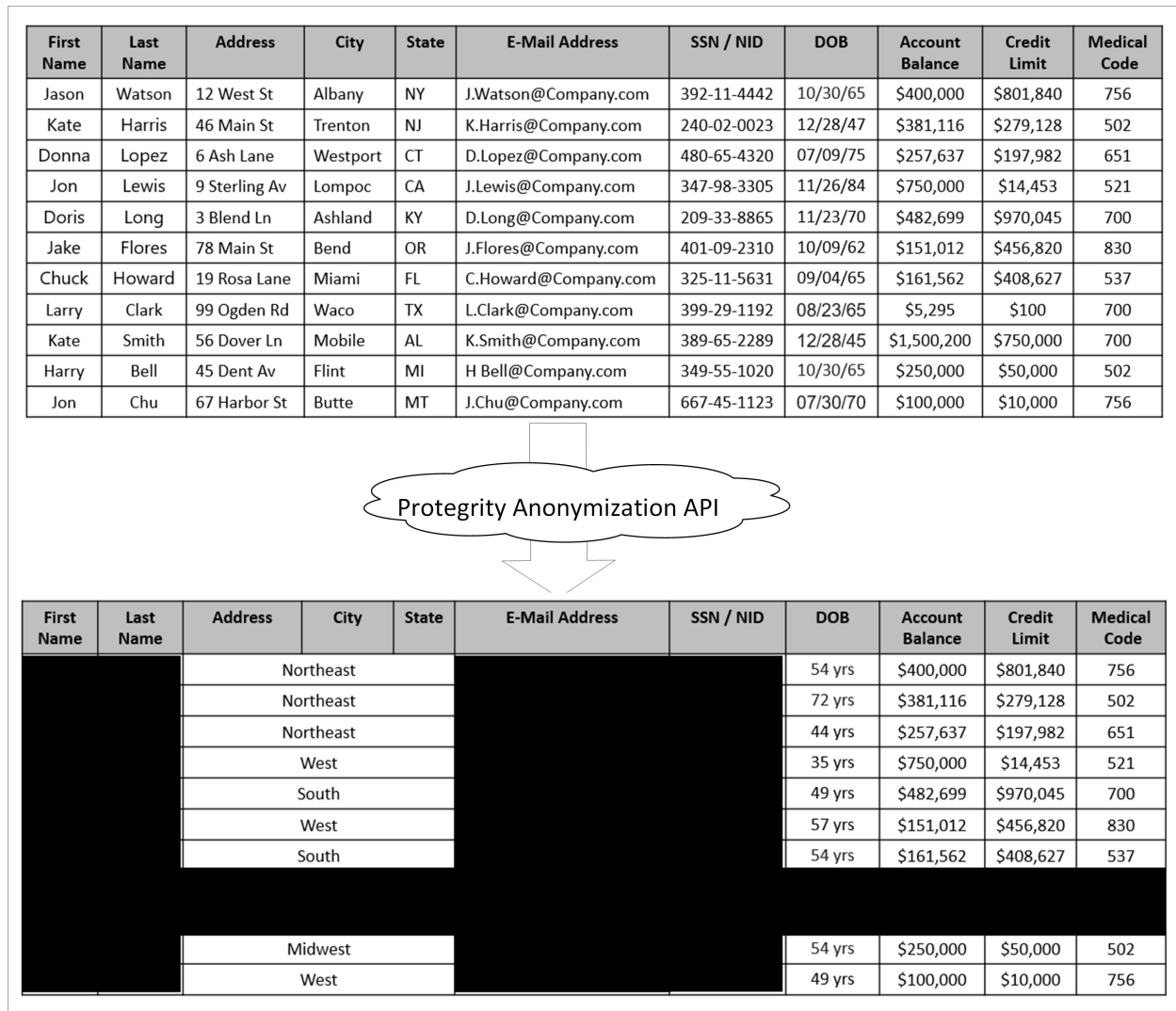


Figure 2-1: Basics of Protegrity Anonymization API

As shown in the above image, a sample table is fed as input into the Protegrity Anonymization API. The private data that can be used to identify a particular individual is removed from the table. The final table with anonymized information is provided as output.

In hiding information about the users, Protegrity Anonymization API makes the data available for further processing and at the same time protecting the users from de-identification attacks such as prosecutor, journalist, and marketer.

- In *prosecutor*, the attacker has prior knowledge about a specific person whose information is present in the data set. The attacker matches this pre-existing information with the information in the data set and identifies an individual.
- In *journalist*, the attacker uses the prior information that is available. However, this information might not be enough to identify a person in the data set. Here, the attacker might find additional information about the person using public records and narrow down the records to de-identify the individual.
- In *marketer*, the attacker tries to de-identify as many people as possible from the data set. This is a hit or miss strategy and many individuals identified might be incorrect. However, even though a lot of individuals de-identified might be incorrect, it is an issue if even few individuals are identified.

For more information about risk metrics, refer to the section [Protegrity Anonymization API Risk Metrics](#).

2.2 Understanding the Protegrity Anonymization API Components

The Protegrity Anonymization API components come together to anonymize the data. At its core, the Protegrity Anonymization API is composed of the following main components:

- *Protegrity Anonymization REST Server*: This is the core component that exposes a REST interface through which clients can interact with the system. The Protegrity Anonymization API uses an in-memory tasks queue and a persistent repository. Anonymization tasks are submitted to the queue and are handled in the order that they are received. The Protegrity Anonymization API invokes the Dask Scheduler to perform the anonymization task.

Note: Only one anonymization task is executed at a time in the Protegrity Anonymization API.

- *REST Client*: The client connects to the *Protegrity Anonymization REST Server* using an API tool, such as Postman, to create, send, and receive the anonymization request. It also provides a Swagger interface detailing the APIs available. The Swagger interface can also be used as a REST client for raising API requests.
- *Python SDK*: It is the Python programmatic interface used to communicate to the REST server. It simplifies usage of the anonymization process.
- *Anon-Storage*: It is used to read and write data to the storage using the HTTP protocol. It is an Hadoop Distributed File System (HDFS) cluster consisting of one namenode and one datanode. The namenode is to accept request for file operations. The datanode is to store data.
- *Anon-DB*: It is a PostgreSQL database that is used to store data related to the anonymization job.
- *Dask Scheduler*: This component analyzes the work load and distributes processing of the data set to one or more Dask Workers. The scheduler can invoke additional workers or reduce the number of workers required for processing the task. The Dask Scheduler analyzes the data set as a whole and allocates a small chunk of the data set to each worker.
- *Dask Worker*: This component is registered with the Dask Scheduler and processes the data set. It is the Dask library that handles the interaction and interface with the data sets and the storage. The Protegrity Anonymization API supports cloud storage, HDFS, and other storages compatible with Kubernetes. The repository can also be kept outside the container. The Dask Worker works on a subset of the entire data.
- *Jupyter Lab Workstation*: The Jupyter Lab notebook provides a ready environment to run an anonymization request using the Protegrity Anonymization SDK with minimum configuration. To use the notebook, you open the notebook, update the required parameters in the notebook, and run the request.

2.3 Communication Between the Components

The communication between the Protegrity Anonymization API, the Dask Scheduler, and Dask Workers is detailed in this section. An overview of the communication is shown in the following figure.

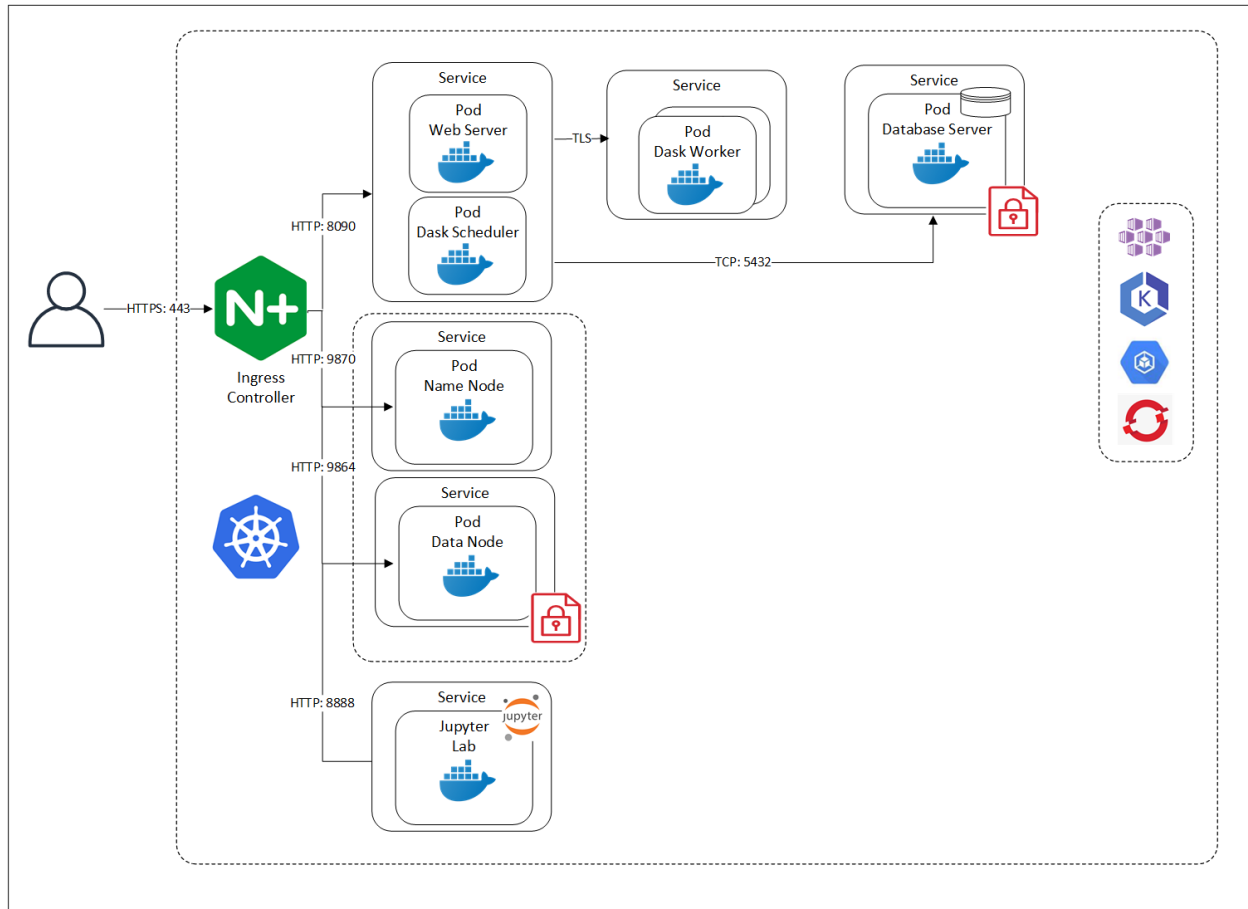


Figure 2-2: Protegrity Anonymization API Communication

Using Kubernetes, the Protegrity Anonymization API is hosted using pods. The first pod contains the Protegrity Anonymization API Web Server and the Dask Scheduler, each loaded in their own containers. This pod connects to the Dask Worker pod over TLS. If the Protegrity Anonymization API requires more processing to work with the data set, then based on the configuration, additional Dask Worker pods can be added. The processing is carried out using an internal Database Server for holding the data securely. The anonymization request is received by Nginx-Ingress. Ingress forwards the request to the Anon-App. The Anon-App processes the request and submits the tasks to the Dask Cluster. The Dask Scheduler schedules task on the Dask Workers. The Anon-app stores the metadata about the job in the Anon-DB container. Next, the Dask Workers read, write, and process the data that is stored in the Anon-Storage, the request stream, or the Cloud storage. The Anon-workstation is in-built in the SDK environment using Jupyter notebook. The communication between the Dask Scheduler and the Dask Workers is handled by the Dask Scheduler. The Dask workers run on random ports.

The Name Node and the Data Node is each hosted in their own pods. These nodes help in storing the anonymized data securely.

The *user* accesses Protegrity Anonymization API using HTTPS over the *443* port. The user requests are directed to an Ingress Controller, the controller in turn communicates with the required pods using the following ports:

- 8090: Ingress controller and the Protegrity Anonymization API Web Service.
- 9870: Ingress controller and the Name Node.
- 9864: Ingress controller and the Data Node.
- 8888: Ingress controller and the Jupyter Lab service

The entire Protegrity Anonymization API can be hosted using any Cloud-provided Kubernetes services, such as, Amazon Elastic Kubernetes Service, Google Kubernetes Engine, Azure Kubernetes Service, Red Hat OpenShift, or others.

Chapter 3

Installing the Protegrity Anonymization API

3.1 Installing the Protegrity Anonymization REST API

3.2 Installing Using Docker Containers

3.3 Installing the Protegrity Anonymization SDK

The Protegrity Anonymization API is available as a REST API that can be installed and run from any Kubernetes environment. After installing the REST API, you can use Protegrity Anonymization API to anonymize your data.

3.1 Installing the Protegrity Anonymization REST API

Complete the steps provided in this section to install the Protegrity Anonymization REST API in your Cloud environment.

3.1.1 Prerequisites for Deploying the Protegrity Anonymization API

The Protegrity Anonymization API is provided as a Docker image. Prepare your system to run commands for processing the basic Kubernetes services for setting up the Protegrity Anonymization API. Additionally, ensure that the following prerequisites are met:

- The user should be well versed with using container orchestration service like Kubernetes in different cloud services.
- Access as an Admin user is available for the cloud service used.
- A minimum of 2 nodes with the following minimum configuration:
 - RAM: 16 GB
 - CPU: 8 core
 - Hard Disk: Unlimited
- Verify the contents of the package after extracting the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tgz* and *ANON-SDK_ALL-ALL-64_x86-64_PY-3-64_1.1.0.0.x.tgz* files from the .tgz archive.
- *ANON-REST-API_1.1.0.0.x.tgz* – Installation package for the Protegrity Anonymization API. This package contains the following files:
 - *ANON-API_1.1.0.0.x.tar.gz* - This image is used to create the Protegrity Anonymization API Docker Container.
 - *cluster-aws.yaml* - This is the template configuration file for creating the cluster in the AWS Cloud environment.
 - *ANON-API_HELM_1.1.0.0.x.tgz* – This contains the Helm chart, which is used to deploy the Protegrity Anonymization API application on the Kubernetes cluster.
 - *Anon_logs.sh* - This is the script for extracting the logs from the Protegrity Anonymization API container.
 - *README.txt* - This readme contains information about the Protegrity Anonymization API.
 - *Contractual.csv* - This contains the list of libraries used in the Protegrity Anonymization API.
 - *docker/docker-compose.yaml* - This file is used to deploy the API in Docker containers.
 - *docker/ptystorage.env* - This file is used to configure the storage for the Docker containers.

- *docker/nginx.conf* - This file is used to configure nginx for Docker.
- *docker/cert/cert.pem* - This is the default self-signed certificate for the Docker container.
- *docker/cert/key.pem* - This is the key for the Docker container.
- *aws-terraform/main.tf* - This template file is used to deploy the API in AWS using Terraform.
- *aws-terraform/vars.tf* - This file is used for specifying the cluster configuration information.
- *rbac/kubconfigcmd.txt* - This file contains the commands for working with RBAC.
- *rbac/anon-service-account.yaml* - This template file contains the RBAC namespace configuration information.
- *rbac/anon-role-and-rolebinding.yaml* - This template file contains the RBAC configuration information for the roles and role binding.
- *rbac/anon-clusterrolebinding.yaml* - This file contains the RBAC configuration information for binding the roles to the cluster.
- *rbac/kubconfigcmd.txt* - This file contains the RBAC commands for retrieving tokens and assigning access to the service account.
- *ANON-STORAGE_1.1.0.0.x.tgz* - Configuration for the Protegrity Anonymization API storage. Do not extract or modify the contents of this file.
- *ANON-NOTEBOOK_1.1.0.0.x.tgz* - Docker image for the Protegrity Anonymization API Notebook workstation. Do not extract or modify the contents of this file.
- *ANON-SDK_ALL-ALL-64_x86-64_PY-3-64_1.1.0.0.x.tgz* - Contains the *Anonsdk-wheel* file that is used to install *anonsdk* in the Python environment.
- If required, then a REST client to access the REST services, such as, Postman.

3.1.2 Using Cloud Services

The Protegrity Anonymization API can be hosted in the Kubernetes service provided by various cloud platforms, such as AWS and Azure. Use the content provided in the following sections for configuring the Protegrity Anonymization API in the different cloud services.

Note: Protegrity Anonymization API is compatible for use with other Cloud providers. However, the compatibility has not been tested.

3.1.2.1 Anonymizing Using Amazon Elastic Kubernetes Service (EKS)

Set up and use the Protegrity Anonymization API on Amazon Elastic Kubernetes Service (EKS) using the information provided in this section.

Note: You can create the Kubernetes cluster on AWS using eksctl or Terraform.

3.1.2.1.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

- *Base machine* - This might be a Linux machine instance that is used to communicate with the Kubernetes cluster. This instance can be on-premise or on AWS. Ensure that Helm is installed on this Linux instance. You must also install Docker on this Linux instance to communicate with the Container Registry, where you want to upload the Docker images.

For more information about the minimum hardware requirements, refer to the section [Prerequisites for Deploying the Protegrity Anonymization API](#).

- Access to an AWS account.
- Permissions to create a Kubernetes cluster.
- IAM user:
 - Required to create the Kubernetes cluster. This user requires the following permissions:

- *AmazonEC2FullAccess* - This is a policy managed by AWS
- *AmazonEKSClusterPolicy* - This is a policy managed by AWS
- *AmazonS3FullAccess* - This is a policy managed by AWS
- *AmazonSSMFullAccess* - This is a policy managed by AWS
- *AmazonEKSServicePolicy* - This is a policy managed by AWS
- *AmazonEKS_CNI_Policy* - This is a policy managed by AWS
- *AWSCloudFormationFullAccess* - This is a policy managed by AWS
- Custom policy that allows the user to create a new role and an instance profile, retrieve information regarding a role and an instance profile, attach a policy to the specified IAM role, and so on. The following actions must be permitted on the IAM service:
 - `GetInstanceProfile`
 - `GetRole`
 - `AddRoleToInstanceProfile`
 - `CreateInstanceProfile`
 - `CreateRole`
 - `PassRole`
 - `AttachRolePolicy`
- Custom policy that allows the user to delete a role and an instance profile, detach a policy from a specified role, delete a policy from the specified role, remove an IAM role from the specified EC2 instance profile, and so on. The following actions must be permitted on the IAM service:
 - `GetOpenIDConnectProvider`
 - `CreateOpenIDConnectProvider`
 - `DeleteInstanceProfile`
 - `DeleteRole`
 - `RemoveRoleFromInstanceProfile`
 - `DeleteRolePolicy`
 - `DetachRolePolicy`
 - `PutRolePolicy`
- Custom policy that allows the user to manage EKS clusters. The following actions must be permitted on the EKS service:
 - `ListClusters`
 - `ListNodegroups`
 - `ListTagsForResource`
 - `ListUpdates`
 - `DescribeCluster`
 - `DescribeNodegroup`
 - `DescribeUpdate`
 - `CreateCluster`
 - `CreateNodegroup`
 - `DeleteCluster`
 - `DeleteNodegroup`
 - `UpdateClusterConfig`
 - `UpdateClusterVersion`
 - `UpdateNodegroupConfig`
 - `UpdateNodegroupVersion`

For more information about creating an IAM user, refer to the section *Creating an IAM User in Your AWS Account* at https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html. Contact your system administrator for creating the IAM users.

For more information about the AWS-specific permissions, refer to the API Reference document for [Amazon EKS](#).

- Access to the Amazon Elastic Kubernetes Service (EKS) to create a Kubernetes cluster.
- Access to the AWS Elastic Container Registry (ECR) to upload the Protegrity Anonymization API image.

3.1.2.1.2 Preparing the Base Machine

Use the steps provided in this section to prepare the machine for working with the EKS cluster. The steps provided here installs the software required for running the various EKS commands for setting up and working with the Protegrity Anonymization API cluster.

1. Login to your system as an administrator.
2. Open a command prompt with administrator.
3. Install the following tools to get started with creating the EKS cluster.
 - a. Install AWS CLI 2, which provides a set of command line tools for the AWS Cloud Platform.

For more information about installing the AWS CLI 2, navigate to <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>.

- b. Configure AWS CLI on your machine by running the following command.

```
aws configure
```

You are prompted to enter the AWS Access Key ID, Secret Access Key, AWS Region, and the default output format where these results are formatted.

For more information about configuring AWS CLI, refer to <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html>.

You need to specify the credentials of IAM User created in the section *Verifying the Prerequisites* to create the Kubernetes cluster..

```
AWS Access Key ID [None]: <AWS Access Key ID of the IAM User 1>
AWS Secret Access Key [None]: <AWS Secret Access Key of the IAM User 1>
Default region name [None]: <Region where you want to deploy the Kubernetes cluster>
Default output format [None]: json
```

- c. Install Kubectl version 1.22, which is the command line interface for Kubernetes.

Kubectl enables you to run commands from the Linux instance so that you can communicate with the Kubernetes cluster.

For more information about installing *kubectl*, refer to the section *Installing kubectl* in the AWS documentation.

- d. Install one of the following command line tools for creating the Kubernetes cluster on AWS (EKS):
 - **eksctl**: Install eksctl which is a command line utility to create and manage Kubernetes clusters on Amazon Elastic Kubernetes Service (Amazon EKS).

For more information about installing eksctl on the Linux instance, refer to <https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>.
 - **Terraform**: Optionally, install Terraform which is the command line to create and manage Kubernetes clusters. Use the *terraform version* command in the CLI to verify that Terraform is installed.

For more information about installing Terraform, refer to <https://www.terraform.io/downloads.html>.

- e. Install the Helm client version 3 for working with Kubernetes clusters.

For more information about installing the Helm client, refer to <https://helm.sh/docs/intro/install/>.

3.1.2.1.3 Creating the EKS Cluster

Complete the steps provided here to create the EKS cluster by running commands on the machine for the Protegrity Anonymization API.

Note: The steps listed in this procedure for creating the EKS cluster are for reference use. If you have an existing EKS cluster or want to create an EKS cluster based on your own requirements, then you can directly navigate to the section [Accessing the Cluster](#) to connect your EKS cluster and the Linux instance.

► To create an EKS cluster:

1. Login to the Linux machine.
2. Obtain and extract the Protegrity Anonymization API files to a directory on your system.
 - a. Download and extract the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tgz* file.
 - b. Verify that the following files are available in the package.
 - *ANON-REST-API_1.1.0.0.x.tgz*: The files for working with Protegrity Anonymization REST API.
 - *ANON-STORAGE_1.1.0.0.x.tgz*: The files for the Protegrity Anonymization API storage.
 - *ANON-NOTEBOOK_1.1.0.0.x.tgz*: This file contains the image for the Anon-workstation.
 - c. Extract the contents of the *ANON-REST-API_1.1.0.0.x.tgz*, *ANON-STORAGE_1.1.0.0.x.tgz*, and *ANON-NOTEBOOK_1.1.0.0.x.tgz* files to a directory.
3. Add the Cloud-related settings in the configuration files using one of the following options:

Note: Use the checklist at [AWS Checklist](#) to update the *YAML* files.

- **For eksctl:** Update the *cluster-aws.yaml* template file with the EKS authentication values for creating the EKS cluster.
- Update the following placeholder information in the *cluster-aws.yaml* file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: <cluster_name>    #(provide an appropriate name for your cluster)
  region: <Region where you want to deploy Kubernetes Cluster>    #(specify the
region to be used)
  version: "1.22"
vpc:
  id: "#Update_vpc_here#" #    (enter the vpc id to be used)
  subnets:                # (In this section specify the subnet region and subnet id
accordingly)
    private:
      <Availability zone for the region where you want to deploy your Kubernetes
cluster>:
        id: "#Update_id_here#"
      <Availability zone for the region where you want to deploy your Kubernetes
cluster>:
        id: "#Update_id_here#"
iam:
  serviceRoleARN: "arn:aws:iam::<#Update AWS Account ID#>:role/<#Update EKS SERVICE
ROLE NAME#>"
nodeGroups:
  - name: <Name of your Node Group>
```

```

instanceType: t3a.xlarge
minSize: 2
maxSize: 4          # (Set max node size according to load to be processed , for
cluster-autoscaling )
desiredCapacity: 3
privateNetworking: true
iam:
  attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
    - "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  withAddonPolicies:
    autoScaler: true
    albIngress: true
securityGroups:
  withShared: true
  withLocal: true
  attachIDs: ['#Update_security_group_id_linked_to_your_VPC_here#']
tags:
  #Add required tags (Product, name, etc.) here
  k8s.io/cluster-autoscaler/<cluster_name>: "owned"          # (Update your cluster
name in this line) ## These tags are required for
  k8s.io/cluster-autoscaler/enabled:
"true"                ## cluster-autoscaling
  Product: "Anonymization"
ssh:
  publicKeyName: '<EC2 Key Pair>'          #Add SSH key to login to
Nodes in the cluster if needed.

```

Note: In the `ssh/publicKeyName` parameter, you must specify the name of the key pair that you have created. For more information about creating the EC2 key pair, refer to <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>.

The `AmazonEKSWorkerNodePolicy` policy allows Amazon EKS worker nodes to connect to Amazon EKS Clusters. For more information about the policy, refer to [https://console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy\\$jsonEditor](https://console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy$jsonEditor).

For more information about the attached role `arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy` in the nodegroup, refer to <https://docs.aws.amazon.com/eks/latest/userguide/create-node-role.html>.

The ARN of the `AmazonEKS_CNI_Policy` policy is a default AWS policy that enables the Amazon VPC CNI Plugin to modify the IP address configuration on your EKS nodes. For more information about this policy, refer to https://console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy?section=access_advisor.

For more information about the attached role `arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy` in the nodegroup, refer to <https://docs.aws.amazon.com/eks/latest/userguide/cni-iam-role.html>.

- **For Terraform:** Update the following placeholder information in the `aws-terraform/vars.tf` file with the Terraform values for creating the cluster.

```

variable "cluster_name" {
  default = "<Cluster_name>"          ## Supply the name for your EKS cluster.
}
variable "cluster_version" {
  default = "1.22"
}
variable "aws_region" {
  default = "<Region>"                ## The region in which EKS cluster will be
created.
}
variable "role_arn" {
  default = "<Specify Role_arn>"      ## Amazon Resource Name (ARN) of the IAM
role that provides permissions for the Kubernetes control plane to make calls to AWS
API operations on your behalf.
}
variable "security_group_id" {
  default = ["<Specify security group id>"]  ## The Security Group ID for your VPC.
}

```

```

variable "subnet_ids" {
  default = ["<subnet-1 id>", "<subnet-2 id>"]    ## Supply the subnet ID's. Ensure the
  subnets should be in different Availability Zone.
}
variable "node_group_name" {
  default = "<Nodegroup Name>"                  ## Name of the nodegroup that will join the
  EKS cluster.
}
variable "node_role_arn" {
  default = "<IAM-Node ROLE ARN>"                ## Amazon Resource Name (ARN) of the IAM
  Role that provides permissions for the EKS Node Group.
}
variable "instance_type" {
  default = ["<instance_type>"]                  ## Type of Nodes in EKS cluster. Eg:
  t3a.xlarge.
}
variable "desired_nodes_count" {
  default = "<Desired node count>"                ## Desired number of Nodes Running in EKS
  cluster.
}
variable "max_nodes" {
  default = "<Max node count>"                    ## Maximum number of Nodes in EKS cluster
  can Autoscale to.
}
variable "min_nodes" {
  default = "<Min node count>"                    ## Minimum number of Nodes in EKS cluster.
}
variable "ssh_key" {
  default = "<EC2-SSH-key>"                      ## EC2-SSH Key Pair to SSH to Nodes of
  cluster.
}
output "endpoint" {
  value = aws_eks_cluster.eks_Anon.endpoint
}

```

- Run one of the the following commands to create the Kubernetes cluster. This process might take time to complete. You might need to wait for 10 to 15 minutes for it to complete:

- **For eksctl:**

```
eksctl create cluster -f cluster-aws.yaml
```

- **For Terraform:**

```
terraform init terraform plan terraform apply
```

- Deploy the Cluster Autoscaler component to enable the autoscaling of nodes in the EKS cluster.
For more information about deploying the Cluster Autoscaler, refer to the *Deploy the Cluster Autoscaler* section in the [Amazon EKS](#) documentation.
- Install the Metrics Server to enable the horizontal autoscaling of pods in the Kubernetes cluster.
For more information about installing the Metrics Server, refer to the *Horizontal Pod Autoscaler* section in the [Amazon EKS](#) documentation.

3.1.2.1.4 Accessing the Cluster

Connect to the cloud service using the steps in this section.

- Run the following command to connect your Linux instance to the Kubernetes cluster.

```
aws eks update-kubeconfig --name <Name of Kubernetes cluster> --region <Region in which
the cluster is created>
```

- Run the following command to verify that the nodes are deployed.

```
kubectl get nodes
```

Note: You can also verify that the nodes are deployed in AWS from the EKS Kubernetes Cluster dashboard.

3.1.2.1.5 Uploading the Image to AWS Container Registry (ECR)

Use the information in the following section to upload the Protegrity Anonymization API image to the AWS container registry (ECR) for running the Protegrity Anonymization API in EKS.

Before you begin

Ensure that you have set up your Container Registry.

Note: The steps listed in this section for uploading the container images to Amazon Elastic Container Repository (ECR) are for reference use. You can choose to use a different Container Registry for uploading the container images.

For more information about setting up Amazon ECR, refer to <https://docs.aws.amazon.com/AmazonECR/latest/userguide/get-set-up-for-amazon-ecr.html>.

► Install the Protegrity Anonymization API using the following steps:

- Login to the machine as an administrator to install the Protegrity Anonymization API.
- Install Docker using the steps provided at <https://docs.docker.com/engine/install/>.
- Configure docker to push the Protegrity Anonymization API images to the AWS Container Registry (ECR) by running following command:

```
aws ecr get-login-password --region us-east-1 | sudo docker login --username
<AWS_username> --password-stdin <Account.amazon>.amazonaws.com
```

- Obtain and extract the Protegrity Anonymization files to a directory on your system.
 - Download and extract the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tar.gz* file.
 - Extract the contents of the *ANON-REST-API_1.1.0.0.x.tar.gz* and *ANON-STORAGE_1.1.0.0.x.tar.gz* files to a directory.

Note: Do not extract the *ANON-API_1.1.0.0.x.tar.gz* package obtained in the directory after performing the extraction. You need to run the *docker load* command on the package obtained in the directory.

- Navigate to the directory where the *ANON-API_1.1.0.0.x.tar.gz* file is saved.
- Load the docker image into Docker by using the following command:

```
docker load < ANON-API_1.1.0.0.x.tar.gz
```

- List the images loaded by using the following command:

```
docker images
```

- Tag the image to the ECR repository by using the following command:

```
docker tag <Container image>:<Tag> <Container registry path>/<Container image>:<Tag>
```

For example:

```
docker tag ANON-API_1.1.0.0.x:anon_EKS <account_name>.dkr.ecr.region.amazonaws.com/
anon:anon_EKS
```

9. Push the tagged image to the ECR by using the following command:

```
docker push <Container_registry_path>/<Container_image>:<Tag>
```

For example:

```
docker push <account_name>.dkr.ecr.region.amazonaws.com/anon:anon_EKS
```

10. Repeat steps 5 to 9 for *ANON-STORAGE_1.1.0.0.x.tgz*.
11. Extract *ANON-NOTEBOOK_1.1.0.0.x.tgz* to obtain the *ANON-NOTEBOOK_1.1.0.0.x.tar.gz* file and then repeat the steps 5 to 9 for *ANON-NOTEBOOK_1.1.0.0.x.tar.gz*.
12. Extract *ANON-Workstation_1.1.0.0.x.tgz* to obtain the *ANON-Workstation_1.1.0.0.x.tar.gz* file and then repeat the steps 5 to 9 for *ANON-Workstation_1.1.0.0.x.tar.gz*.

The images are loaded to the ECR and are ready for deployment.

For more information about pushing container images to the ECR, navigate to <https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html>.

3.1.2.1.6 Creating an EBS Volume on AWS

Complete the steps provided here to create an EBS volume and obtain the volume ID.

Before you begin

Ensure that you select the *availability zone* of at least one node that is part of the EKS cluster.

To create the EBS volume:

1. Navigate to <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-creating-volume.html> and complete the steps provided to create the EBS volume.
2. Note the *volumeID* of the EBS volume that you created.

3.1.2.1.7 Setting up NGINX Ingress Controller

Complete the steps provided here for installing the NGINX Ingress Controller on the base machine.

1. Login to the base machine and open a command prompt.
2. Add the repository from where the Helm charts for installing the NGINX Ingress Controller must be fetched using the following command.

```
helm repo add stable https://charts.helm.sh/stable
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

3. Install the NGINX Ingress Controller using Helm charts using the following command.

```
helm install nginx-ingress --namespace <Namespace name> --set controller.replicaCount=1
--set controller.nodeSelector."beta\.kubernetes\.io/os"=linux --set
defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux ingressnginx/
ingress-nginx --set controller.publishService.enabled=true --set
controller.ingressClassResource.name=<NGINX ingress class name> --set
podSecurityPolicy.enabled=true --set rbac.create=true --set
```

```
controller.extraArgs.enablessl-
passthrough="true" --set controller.service.annotations."service\\.beta\\.kubernetes
\\.io/aws-load-balancer-internal"="true\"
```

For example,

```
helm install nginx-ingress --namespace nginx --set controller.replicaCount=1 --set
controller.extraArgs.enable-ssl-passthrough="true" --set controller.nodeSelector."beta
\\.kubernetes\\.io/os=linux --set defaultBackend.nodeSelector."beta\\.kubernetes\\.io/
os=linux ingress-nginx/ingress-nginx --set controller.publishService.enabled=true --set
controller.ingressClassResource.name=nginx-anon --set podSecurityPolicy.enabled=true --
set rbac.create=true --set controller.service.annotations."service\\.beta\\.kubernetes\\
\\.io/aws-load-balancer-internal"="true\"
```

For more information about the various configuration parameters for installing the NGINX ingress Helm charts, refer to <https://github.com/kubernetes/ingress-nginx/tree/main/charts/ingress-nginx>.

4. Check the status of the nginx-ingress release and verify that all the deployments are running accurately using the following command.

```
kubectl get pods -n <Namespace name>
```

For example,

```
kubectl get pods -n nginx
```

Note: Note the pod name. It is required as a parameter in the next step.

5. View the logs on the Ingress pod using the following command.

```
kubectl logs pod/<pod-name> -n <Namespace name>
```

6. Obtain the external IP of the Nginx service by executing the following command.

```
kubectl get service --namespace <Namespace name>
```

For example,

```
kubectl get service -n nginx
```

Note: Note the IP. It is required for communicating the Protegrity Anonymization API.

3.1.2.1.8 Using Custom Certificates in Ingress

Protegrity Anonymization API uses certificates for secure communication with the client. You can use the certificates provided by Protegrity or use your own certificates. Complete the configurations provided in this section to use your custom certificates with the Ingress Controller.

Before you begin

Ensure that the certificates and keys are in the *.pem* format.

Note: Skip the steps provided in this section if you want to use the default Protegrity certificates for the Protegrity Anonymization API.

1. Login to the Base Machine where Ingress is configured and open a command prompt.
2. Copy your certificates to the Base Machine.

Note: Verify the certificates using the commands provided in the section [Working with Certificates](#).

3. Create a Kubernetes secret of the server certificate using the following command. The namespace used must be the same where the Protegrity Anonymization API application is to be deployed.

```
kubectl create secret --namespace <namespace-name> generic
<secret-name> --from-file=tls.crt=<path_to_certificate>/<certificate-name> --from-
file=tls.key=<path_to_certificate>/<certificate-key>
```

For example,

```
kubectl create secret --namespace anon-ns generic anon-protegrity-tls --from-
file=tls.crt=/tmp/cust_cert/anon-server-cert.pem --from-file=tls.key=/tmp/cust_cert/anon-
server-key.pem
```

4. Create a Kubernetes secret of the CA certificate using the following command. The namespace used must be the same where the Protegrity Anonymization API application is to be deployed.

```
kubectl create secret --namespace <namespace-name> generic <secret-name> --from-
file=ca.crt=<path_to_certificate>/<certificate-name>
```

For example,

```
kubectl create secret --namespace anon-ns generic ca-protegrity --from-file=ca.crt=/tmp/
cust_cert/anon-ca-cert.pem
```

5. Open the *values.yaml* file.
6. Add the following *host* and *secret* code for the Ingress configuration at the end of the *values.yaml* file.

```
## Refer section in documentation for setting up and configuring NGINX-INGRESS before
## deploying the application.
ingress:
  ## Add host section with the hostname used as CN while creating server certificates.
  ## While creating the certificates you can use *.protegrity.com as CN and SAN used in
  ## below example
  host: anon.protegrity.com # Update the host according to your server
  certificates.

  ## To terminate TLS on the Ingress Controller Load Balancer.
  ## K8s TLS Secret containing the certificate and key must also be provided.
  secret: anon-protegrity-tls # Update the secretName according to your
  secretName.

  ## To validate the client certificate with the above server certificate
  ## Create the secret of the CA certificate used to sign both the server and client
  ## certificate as shown in example below
  ca_secret: ca-protegrity # Update the ca-secretName according to
  your secretName.

  ingress_class: nginx-anon
```

Note: Ensure that you replace the *host*, *secret*, and *ca_secret* attributes in the *values.yaml* file with the values as per your certificate. For more information about using custom certificates, refer to the section [Updating the Configuration Files](#).

3.1.2.1.9 Updating the Configuration Files

Use the template files provided to specify the EKS settings for the Protegrity Anonymization API.

1. Extract and update the files in the *ANON-API_HELM_1.1.0.0.x.tgz* package.
The *ANON-API_HELM_1.1.0.0.x.tgz* package contains the *values.yaml* file that must be modified as per your requirements. It also contains the *templates* directory with *yaml* files.

Note: Ensure that the necessary permissions for updating the files are assigned to the *.yaml* files.

2. Navigate to the `<path_to_helm>/templates` directory and delete the `anon-db-storage-aws.yaml` file.
3. Update the `values.yaml` file.

Note: For more information about the `values.yaml` file, refer to the section [values.yaml](#).

- a. Specify a namespace for the pods.

```
namespace:
  name: anon-ns
```

- b. Specify the node name and zone information for the node as a prerequisite for the database pod. Use the node name which is running in the same zone where the external-storage is created.

```
## Prerequisite for setting up Database Pod.
## This is to handle any new DB pod getting created uses same persistence storage in
## case the running Database pod disrupts.
dbpersistence:
  ## 1. Get the list of nodes in the cluster. CMD: kubectl get nodes
  ## 2. Get the node name which is running in the same zone where the external-storage
  ## is created. CMD: kubectl describe nodes
  nodename: "<Node_name>" # Update the Node name

  ## Fetch the zone in which the node is running using the `kubectl describe node/
  ## nodename` or following command.
  ## CMD: ` kubectl describe node/<nodename> | grep topology.kubernetes.io/zone | grep
  ## -oP 'topology.kubernetes.io/zone=\K[^\s]+' `
  zone: "<Zone in which above Node is running>"

  ## Supply the volumeID of the aws-efs incase of EKS cluster
  ## Supply the subscriptionID of the azure-disk incase of AKS cluster
  storageId: "<Provide storage ID>"
```

- c. Specify the Protegrity Anonymization API information for your deployment.

```
image:
  repository: <Repo_path> # Repo path for Container Registry
  in Azure, GCP, AWS

  anonapi_tag: <AnonImage_tag> # Tag name of ANON-API Image.
  anonstorage_tag: <StorageImage_tag> # Tag name of ANON-Storage Image.
  anonworkstation_tag: <WorkstationImage_tag> # Tag name of ANON-Workstation
  Image.

  pullPolicy: Always
```

Note: Ensure that you update the `repository`, `anonapi_tag`, `anonstorage_tag`, and `anonworkstation_tag` according to your container registry.

3.1.2.1.10 Deploying the Protegrity Anonymization API to the EKS Cluster

Complete the following steps to deploy the Protegrity Anonymization API on the EKS cluster.

1. Navigate to the `<path_to_helm>/templates` directory and delete the `anon-db-storage-azure.yaml` file.
2. Create the Protegrity Anonymization API namespace using the following command.

```
kubectl create namespace <name>
```

Note: Update and use the `<name>` from the `values.yaml` file that is present in the Helm chart that you used in the previous section.

3. Run the following command to deploy the pods.

```
helm install <helm-name> /<path_to_helm> -n <namespace>
```

4. Verify that the necessary pods and services are configured and running.
 - a. Run the following command to verify the information for accessing the Protegrity Anonymization API externally on the cluster. The port mapping for accessing the UI is displayed after running the command.

```
kubectl get service -n <namespace>
```

- b. Run the following command to verify the deployment.

```
kubectl get deployment -n <namespace>
```

- c. Run the following command to verify the pods created.

```
kubectl get pods -n <namespace>
```

- d. Run the following command to verify the pods.

```
kubectl get pods -o wide -n <namespace>
```

5. If you customize the *values.yaml*, then update the configuration using the following command.

```
helm upgrade <helm name> /path/to/helmchart -n <namespace>
```

6. If required, configure logging using the steps provided in the section [Setting Up Logging for the Protegrity Anonymization API](#).
7. Execute the following command to obtain the IP address of the service.

```
kubectl get ingress -n <namespace>
```

3.1.2.1.11 Viewing Protegrity Anonymization API Using REST

Use the URLs provided here for viewing the Protegrity Anonymization API service and pod details after you have successfully deployed the Protegrity Anonymization API.

Before you begin

You need to map the IP address of Ingress in the *hosts* file with the host name set in the Ingress configuration.

For more information about updating the *hosts* file, refer to *step 2* of the section [Enabling Custom Certificates From SDK](#).

Optionally, update the hostname of the Elastic Load Balancer (ELB) that is created by the NGINX Ingress Controller using the section [Creating a DNS Entry for the ELB Hostname in Route53](#).

For more information about configuring the DNS, refer to the section [Creating a DNS Entry for the ELB Hostname in Route53](#).

1. Open a web browser.
2. Use the following URL to view basic information about the Protegrity Anonymization API.
<https://anon.protegrity.com/>
3. Use the following URL to view the Swagger UI. The various Protegrity Anonymization APIs are visible on this page.
<https://anon.protegrity.com/anonymization/api/v1/ui>
4. Use the following URL to view the contractual information for the Protegrity Anonymization API.
<https://anon.protegrity.com/about>

3.1.2.1.12 Creating Kubernetes Service Accounts and Kubeconfigs for Anonymization Cluster

A service account in the anonymization cluster namespace has access to the anonymization namespace. It might also have access to the whole cluster. These permissions for the service account allow the user to create, read, update, and delete objects in the anonymization Kubernetes cluster or the namespace. Additionally, the kubeconfig is required to access the service account using a token.

In this section, you create a Kubernetes service account and the role-based access control (RBAC) configuration manually using `kubectl`.

Ensure that the user has access to permissions for creating and updating the following resources in the Kubernetes cluster:

- Kubernetes Service Accounts
- Kubernetes Roles and Rolebindings
- Optional: Kubernetes ClusterRoles and Rolebindings

3.1.2.1.12.1 Creating the Service Account

Use the steps provided in this section to create the namespace and assign the required permissions to the cluster.

1. Create the Kubernetes Service Account using the following steps.
 - a. Navigate to the `rbac` directory of the extracted Protegrity Anonymization API package.
 - b. Open the `anon-service-account.yaml` file using a text editor.
 - c. Update the `namespace` as per your configuration in the `anon-service-account.yaml` file.
 - d. Save and close the file.
 - e. From a command prompt, navigate to the `rbac` directory and run the following command to create the service account.

```
kubectl apply -f anon-service-account.yaml
```

2. Grant the appropriate permission to the service account using any one of the following two steps.
 - Grant `cluster-admin` permissions for the service account to all the namespaces using the following steps.

Note: You need to run this step only if you want to grant the service account access to all namespaces in your cluster.

A Kubernetes ClusterRoleBinding is available at the cluster level, but the subject of the ClusterRoleBinding exists in a single namespace. Hence, you must specify the namespace for the service account.

- a. Navigate to the `rbac` directory of the extracted Protegrity Anonymization API package.
- b. Open the `anon-clusterrolebinding.yaml` file using a text editor.
- c. Update the `namespace` as per your configuration in the `anon-clusterrolebinding.yaml` file.
- d. Save and close the file.
- e. From a command prompt, navigate to the `rbac` directory and run the following command to assign the appropriate permissions.

```
kubectl apply -f anon-clusterrolebinding.yaml
```

- Grant namespace-specific permissions to the service account using the following steps.

Note: You need to run this step only if you want to grant the service account access to just the Protegrity Anonymization API namespace.

Ensure that you create a role with a set of permissions and rolebinding for attaching the role to the service account.

- a. Navigate to the *rbac* directory of the extracted Protegrity Anonymization API package.
- b. Open the *anon-role-and-rolebinding.yaml* file using a text editor.
- c. Update the *namespace*, *role*, and *service account name* as per your configuration in the *anon-role-and-rolebinding.yaml* file.
- d. Save and close the file.
- e. From a command prompt, navigate to the *rbac* directory and run the following command to assign the appropriate permissions.

```
kubectl apply -f anon-role-and-rolebinding.yaml
```

3.1.2.1.12.2 Obtaining the Tokens for the Service Account

Complete the steps provided in this section to retrieve the tokens for the Protegrity Anonymization API service account and to create a kubeconfig with access to the service account.

1. Open a command line interface on the base machine for running the configuration commands.

Note: A copy of the commands is available in the *kubconfigcmd.txt* file in the *rbac* directory of the Protegrity Anonymization API package. Use the code from the file to run the commands.

2. Set the environment variables for running the configuration commands using the following command.

```
SERVICE_ACCOUNT_NAME=anon-service-account
CONTEXT=$(kubectl config current-context)
NAMESPACE=anon-namespace
NEW_CONTEXT=anon-context

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} -n ${NAMESPACE} --
context ${CONTEXT} --namespace ${NAMESPACE} -o jsonpath='{.secrets[0].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} -n ${NAMESPACE} --context ${CONTEXT} --
namespace ${NAMESPACE} -o jsonpath='{.data.token}')
TOKEN=$(echo ${TOKEN_DATA} | base64 -d)
```

Note: Ensure that you use the appropriate values as per your configuration in the above command.

3. Set the token in the config credentials using the following command.

```
kubectl config set-credentials <username> --token=$TOKEN
```

For example,

```
kubectl config set-credentials test-user --token=$TOKEN
```

4. Retrieve the cluster name using the following command.

```
kubectl config get-clusters
```

5. Set the context in kubeconfig using the following command.

```
kubectl config set-context ${NEW_CONTEXT} --cluster=<name of your cluster> --user=test-
user
```

6. Set the current context to to use the new anonymization config using the following command.

```
kubectl config use-context ${NEW_CONTEXT}
```

7. Verify the new context using the following command.

```
kubectl config current-context
```

8. Verify the status of the pods using the following command.

```
kubectl get pods -n <name space>
```

3.1.2.2 Anonymizing Using Azure Kubernetes Service (AKS)

Set up and use the Protegrity Anonymization API on Azure using the information provided in this section.

3.1.2.2.1 Preparing the Base Machine

Install the Azure CLI and login to your account to work with Protegrity Anonymization API on the Azure Cloud.

1. Install and initialize the Azure CLI on your system.

For more information about the installation steps, refer to <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>.

2. Login to your account using the following command from a command prompt.

```
az login
```

3. Sign in to your account.

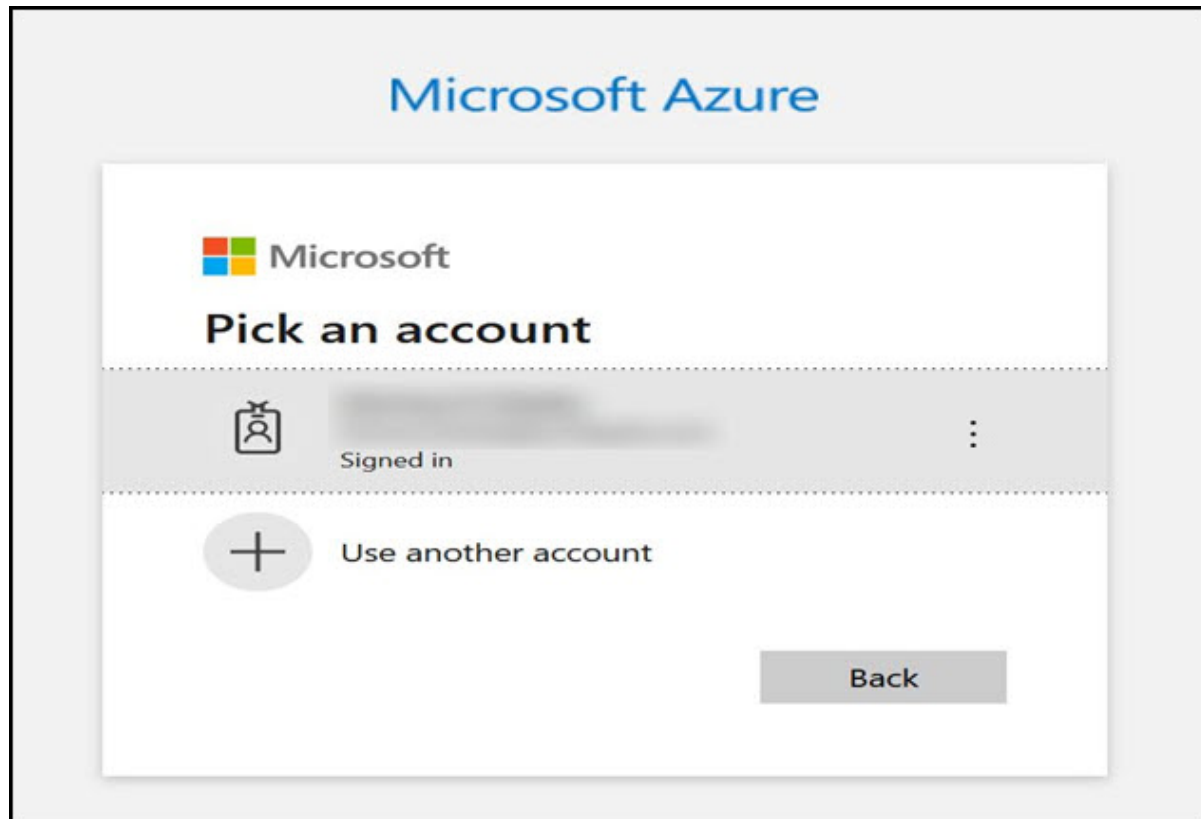


Figure 3-1: Selecting the User Account

The configuration complete message appears.

```

{
  "cloudName": "AzureCloud",
  "homeTenantId": " ",
  "id": " ",
  "isDefault": true,
  "managedByTenants": [
    {
      "tenantId": " "
    }
  ],
  "name": "Azure Cloud Platform",
  "state": "Enabled",
  "tenantId": " ",
  "user": {
    "name": " ",
    "type": "user"
  }
}

```

Figure 3-2: Azure Configuration

4. Install Kubectl version 1.22, which is the command line interface for Kubernetes.
Kubectl enables you to run commands from the Linux instance so that you can communicate with the Kubernetes cluster.
For more information about installing *kubectl*, refer to <https://kubernetes.io/docs/tasks/tools/install-kubectl/>.
5. Install the Helm client version 3 for working with Kubernetes clusters.
For more information about installing the Helm client, refer to <https://helm.sh/docs/intro/install/>.

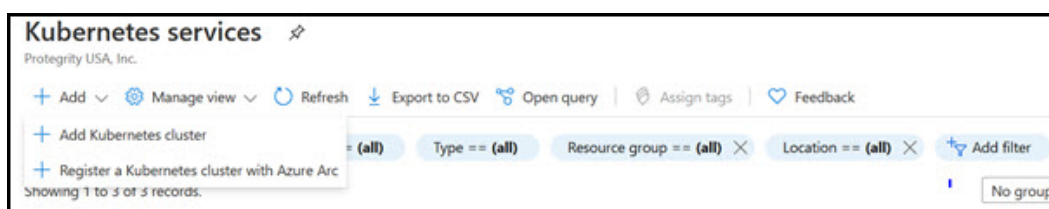
3.1.2.2.2 Creating a Kubernetes Cluster

This section describes how to create a Kubernetes Cluster on Azure.

Note: The steps listed in this procedure for creating a Kubernetes cluster are for reference use. If you have an existing Kubernetes cluster or want to create a Kubernetes cluster based on your own requirements, then you can directly navigate to the section [Accessing the Cluster](#) to connect your Kubernetes cluster and the Linux instance.

► To create a Kubernetes cluster:

1. Login to the Azure environment.
2. Click the **Portal** menu icon (☰).
The **Portal** menu appears.
3. Navigate to **All Services > Kubernetes services**.
The **Kubernetes Services** screen appears.



- Click **Add**.

The **Create Kubernetes cluster** screen appears.

- In the **Resource group** field, select the required resource group.
- In the **Kubernetes cluster name** field, specify a name for your Kubernetes cluster. Retain the default values for the remaining settings.
- Click **Review + create** to validate the configuration.
- Click **Create** to create the Kubernetes cluster. The Kubernetes cluster is created.

3.1.2.2.3 Accessing the Cluster

Connect to the cloud service using the steps in this section.

- Login to the Linux instance, and run the following command to connect your Base machine to the Kubernetes cluster.

```
az aks get-credentials --resource-group <Name_of _Resource_group> --name <Name_of Kubernetes_Cluster>
```

The Base machine is now connected with the Kubernetes cluster. You can now run commands using the Kubernetes command line interface (kubectl) to control the nodes on the Kubernetes cluster.

- Validate whether the cluster is up by running the following command.

```
kubectl get nodes
```

The command lists the Kubernetes nodes available in your cluster.

3.1.2.2.4 Uploading the Image to the Azure Container Registry

Use the information in the following section to upload the Docker image to the Azure container registry (ACR) for running the Protegrity Anonymization API in AKS.

Note: For more information about creating the Azure Container Registry, navigate to <https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-portal?tabs=azure-cli>.

► Install the Protegrity Anonymization API using the following steps:

1. Login to the machine as an administrator to install the Protegrity Anonymization API.
2. Install Docker using the steps provided at <https://docs.docker.com/engine/install/>.
3. Configure docker to push the Protegrity Anonymization API images to the Azure Container Registry (ACR) by running following command:

```
docker login <Container_registry_name>.azurecr.io
```

4. Obtain and extract the Protegrity Anonymization API files to a directory on your system.
 - a. Download and extract the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tgz* file.
 - b. Open the directory and extract the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tar* file.
 - c. Extract the contents of the *ANON-REST-API_1.1.0.0.x.tgz* and *ANON-STORAGE_1.1.0.0.x.tgz* file to a directory.

Note: Do not extract the *ANON-API_1.1.0.0.x.tar.gz* package obtained in the directory after performing the extraction. You need to run the *docker load* command on the package obtained in the directory.

5. Navigate to the directory where the *ANON-API_1.1.0.0.x.tar.gz* file is saved.
6. Load the docker image into Docker by using the following command:

```
docker load < ANON-API_1.1.0.0.x.tar.gz
```

7. List the images loaded by using the following command:

```
docker images
```

8. Tag the image to the ACR repository by using the following command:

```
docker tag <Container image>:<Tag> <Container registry path>/<Container image>:<Tag>
```

For example:

```
docker tag ANON-API_1.1.0.0.x:anon_AZ <container_registry_name>.azurecr.io/anon:anon_AZ
```

9. Push the tagged image to the ACR by using the following command:

```
docker push <Container_registry_path>/<Container_image>:<Tag>
```

For example:

```
docker push <container_registry_name>.azurecr.io/anon:anon_AZ
```

Note: Ensure that the appropriate path for the image registry along with the tag is updated in the *values.yaml* file.

10. Repeat steps 5 to 9 for *ANON-STORAGE_1.1.0.0.x.tgz*.
11. Extract *ANON-NOTEBOOK_1.1.0.0.x.tgz* to obtain the *ANON-NOTEBOOK_1.1.0.0.x.tar.gz* file and then repeat the steps 5 to 9 for *ANON-NOTEBOOK_1.1.0.0.x.tar.gz*.
12. Extract *ANON-Workstation_1.1.0.0.x.tgz* to obtain the *ANON-Workstation_1.1.0.0.x.tar.gz* file and then repeat the steps 5 to 9 for *ANON-Workstation_1.1.0.0.x.tar.gz*.

The image is loaded to the ACR and is ready for deployment.

3.1.2.2.5 Creating an Azure Disk

Complete the steps provided here to create an Azure disk and obtain the subscription ID.

► To create the Azure disk:

1. Navigate to <https://docs.microsoft.com/en-us/azure/aks/azure-disk-volume> and complete the steps provided in the section *Create an Azure disk*.

The command for creating the Azure disk is provided here, update the values according to your setup:

```
az disk create \
  --resource-group <Resource Group Name> \
  --name <Disk Name> \
  --size-gb 20 \
  --location <Location of any node in cluster> \
  --zone <Zone of the node in cluster> \
  --query id --output tsv
```

2. Note the *subscription ID* of the Azure disk that you created.

3.1.2.2.6 Setting up NGINX Ingress Controller

Complete the steps provided here for installing the NGINX Ingress Controller on the base machine.

1. Login to the base machine and open a command prompt.
2. Create a namespace where the NGINX Ingress Controller needs to be deployed using the following command.

```
kubectl create namespace <Namespace name>
```

For example,

```
kubectl create namespace nginx
```

3. Add the repository from where the Helm charts for installing the NGINX Ingress Controller must be fetched using the following command.

```
helm repo add stable https://charts.helm.sh/stable
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

4. Install the NGINX Ingress Controller using Helm charts using the following command.

```
helm install nginx-ingress --namespace <Namespace name> --
set controller.replicaCount=1 --set controller.nodeSelector."beta\.kubernetes\.io/
os"=linux --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux
ingress-nginx/ingress-nginx --set controller.publishService.enabled=true
--set controller.ingressClassResource.name=<NGINX ingress class
name> --set podSecurityPolicy.enabled=true --set
rbac.create=true --set controller.extraArgs.enable-ssl-passthrough="true" --
set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-
internal"="\true\"
```

For example,

```
helm install nginx-ingress --namespace nginx --set controller.replicaCount=1 --set
controller.extraArgs.enable-ssl-passthrough="true" --set controller.nodeSelector."beta\
\.kubernetes\io/os"=linux --set defaultBackend.nodeSelector."beta\kubernetes\io/
os"=linux ingress-nginx/ingress-nginx --set controller.publishService.enabled=true --set
controller.ingressClassResource.name=nginx-anon --set podSecurityPolicy.enabled=true --
set rbac.create=true --set controller.service.annotations."service\beta\.kubernetes\
io/azure-load-balancer-internal"="\true\"
```

For more information about the various configuration parameters for installing the NGINX ingress Helm charts, refer to <https://github.com/kubernetes/ingress-nginx/tree/main/charts/ingress-nginx>.

5. Check the status of the nginx-ingress release and verify that all the deployments are running accurately using the following command.

```
kubectl get pods -n <Namespace name>
```

For example,

```
kubectl get pods -n nginx
```

Note: Note the pod name. It is required as a parameter in the next step.

6. View the logs on the Ingress pod using the following command.

```
kubectl logs pod/<pod-name> -n <Namespace name>
```

7. Obtain the external IP of the Nginx service by executing the following command.

```
kubectl get service --namespace <Namespace name>
```

For example,

```
kubectl get service -n nginx
```

Note: Note the IP. It is required for configuring the Protegrity Anonymization API SDK.

3.1.2.2.7 Using Custom Certificates in Ingress

Protegrity Anonymization API uses certificates for secure communication with the client. You can use the certificates provided by Protegrity or use your own certificates. Complete the configurations provided in this section to use your custom certificates with the Ingress Controller.

Before you begin

Ensure that the certificates and keys are in the *.pem* format.

Note: Skip the steps provided in this section if you want to use the default Protegrity certificates for the Protegrity Anonymization API.

1. Login to the Base Machine where Ingress is configured and open a command prompt.
2. Copy your certificates to the Base Machine.

Note: Verify the certificates using the commands provided in the section [Working with Certificates](#).

3. Create a Kubernetes secret of the server certificate using the following command. The namespace used must be the same where the Protegrity Anonymization API application is to be deployed.

```
kubectl create secret --namespace <namespace-name> generic  
<secret-name> --from-file=tls.crt=<path_to_certificate>/<certificate-name> --from-  
file=tls.key=<path_to_certificate>/<certificate-key>
```

For example,

```
kubectl create secret --namespace anon-ns generic anon-protegrity-tls --from-  
file=tls.crt=/tmp/cust_cert/anon-server-cert.pem --from-file=tls.key=/tmp/cust_cert/anon-  
server-key.pem
```

4. Create a Kubernetes secret of the CA certificate using the following command. The namespace used must be the same where the Protegrity Anonymization API application is to be deployed.

```
kubectl create secret --namespace <namespace-name> generic <secret-name> --from-file=ca.crt=<path_to_certificate>/<certificate-name>
```

For example,

```
kubectl create secret --namespace anon-ns generic ca-protegrity --from-file=ca.crt=/tmp/cust_cert/anon-ca-cert.pem
```

5. Open the *values.yaml* file.
6. Add the following *host* and *secret* code for the Ingress configuration at the end of the *values.yaml* file.

```
## Refer section in documentation for setting up and configuring NGINX-INGRESS before
## deploying the application.
ingress:
  ## Add host section with the hostname used as CN while creating server certificates.
  ## While creating the certificates you can use *.protegrity.com as CN and SAN used in
  ## below example
  host: anon.protegrity.com # Update the host according to your server
  certificates.

  ## To terminate TLS on the Ingress Controller Load Balancer.
  ## K8s TLS Secret containing the certificate and key must also be provided.
  secret: anon-protegrity-tls # Update the secretName according to your
  secretName.

  ## To validate the client certificate with the above server certificate
  ## Create the secret of the CA certificate used to sign both the server and client
  ## certificate as shown in example below
  ca_secret: ca-protegrity # Update the ca-secretName according to
  your secretName.

  ingress_class: nginx-anon
```

Note: Ensure that you replace the *host*, *secret*, and *ca_secret* attributes in the *values.yaml* file with the values as per your certificate. For more information about using custom certificates, refer to the section [Updating the Configuration Files](#).

3.1.2.2.8 Updating the Configuration Files

Use the template files provided to specify the AKS settings for the Protegrity Anonymization API.

1. Create the Protegrity Anonymization API namespace using the following command.

```
kubectl create namespace <name>
```

Note: Update and use the *<name>* from the *values.yaml* file that is present in the Helm chart.

2. Extract and update the files in the *ANON-API_HELM_1.1.0.0.x.tgz* package.
The *ANON-API_HELM_1.1.0.0.x.tgz* package contains the *values.yaml* file that must be modified as per your requirements. It also contains the *templates* directory with *yaml* files.

Note: Ensure that the necessary permissions for updating the files are assigned to the *.yaml* files.

3. Navigate to the *<path_to_helm>/templates* directory and delete the *anon-db-storage-aws.yaml* file.

4. Update the *values.yaml* file.

Note: For more information about the *values.yaml* file, refer to the section [values.yaml](#).

a. Specify a namespace for the pods.

```
namespace:
  name: anon-ns
```

b. Specify the node name and zone information for the node as a prerequisite for the database pod. Use the node name which is running in the same zone where the external-storage is created.

```
## Prerequisite for setting up Database Pod.
## This is to handle any new DB pod getting created uses same persistence storage in
## case the running Database pod disrupts.
dbpersistence:
  ## 1. Get the list of nodes in the cluster. CMD: kubectl get nodes
  ## 2. Get the node name which is running in the same zone where the external-storage
  ## is created. CMD: kubectl describe nodes
  nodename: "<Node_name>" # Update the Node name

  ## Fetch the zone in which the node is running using the `kubectl describe node/
  ## nodename` or following command.
  ## CMD: ` kubectl describe node/<nodename> | grep topology.kubernetes.io/zone | grep
  ## -oP 'topology.kubernetes.io/zone=\K[^\s]+' `
  zone: "<Zone in which above Node is running>"

  ## Supply the volumeID of the aws-efs incase of EKS cluster
  ## Supply the subscriptionID of the azure-disk incase of AKS cluster
  storageId: "<Provide storage ID>"
```

c. Specify the Protegrity Anonymization API information for your deployment.

```
image:
  repository: <Repo_path> # Repo path for Container Registry
  in Azure, GCP, AWS

  anonapi_tag: <AnonImage_tag> # Tag name of ANON-API Image.
  anonstorage_tag: <StorageImage_tag> # Tag name of ANON-Storage Image.
  anonworkstation_tag: <WorkstationImage_tag> # Tag name of ANON-Workstation
  Image.

  pullPolicy: Always
```

Note: Ensure that you update the *repository*, *anonapi_tag*, *anonstorage_tag*, and *anonworkstation_tag* according to your container registry.

5. Extract the *values.yaml*/Helm chart from the package.6. Uncomment the following parameters and update the secret name in the *values.yaml* file.

```
## This section is if image is getting pulled from Azure Container Registry
## create image pull secrets and specify the name here.
## remove the [] after 'imagePullSecrets:' once you specify the secrets
#imagePullSecrets: []
# - name: regcred
```

7. Perform the following steps for the communication between the Kubernetes cluster and the Azure Container Registry.

a. Run the following command from a command prompt to login.

```
docker login
```

b. Specify your ACR access credentials.

8. Create the secret for Azure by using the following command:

```
kubectl create secret generic regcred --  
from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=Kubernetes.io/  
dockerconfigjson --namespace <NAMESPACE>
```

3.1.2.2.9 Deploying the Protegrity Anonymization API to the AKS Cluster

Deploy the pods using the steps in the following section.

1. Run the following command to deploy the pods.

```
helm install <helm-name> /<path_to_helm> -n <namespace>
```

2. Verify that the necessary pods and services are configured and running.
 - a. Run the following command to verify the information for accessing the Protegrity Anonymization API externally on the cluster. The port mapping for accessing the UI is displayed after running the command.

```
kubectl get service -n <namespace>
```

- b. Run the following command to verify the deployment.

```
kubectl get deployment -n <namespace>
```

- c. Run the following command to verify the pods created.

```
kubectl get pods -n <namespace>
```

- d. Run the following command to verify the pods.

```
kubectl get pods -o wide -n <namespace>
```

3. Execute the following command to obtain the IP address of the service.

```
kubectl get ingress -n <namespace>
```

The container is now ready to process Protegrity Anonymization API requests.

3.1.2.2.10 Viewing Protegrity Anonymization API Using REST

Use the URLs provided here for viewing the Protegrity Anonymization API service and pod details after you have successfully deployed the Protegrity Anonymization API.

Before you begin

You need to map the IP address of Ingress in the *hosts* file with the host name set in the Ingress configuration.

For more information about updating the *hosts* file, refer to *step 2* of the section [Enabling Custom Certificates From SDK](#).

1. Open a web browser.
2. Use the following URL to view basic information about the Protegrity Anonymization API.
<https://anon.protegrity.com/>
3. Use the following URL to view the Swagger UI. The various Protegrity Anonymization APIs are visible on this page.
<https://anon.protegrity.com/anonymization/api/v1/ui>
4. Use the following URL to view the contractual information for the Protegrity Anonymization API.
<https://anon.protegrity.com/about>

3.2 Installing Using Docker Containers

Complete the following steps to run the Protegrity Anonymization API on a host machine.

Before you begin

Ensure that you have completed the following prerequisites before deploying the Protegrity Anonymization API.

1. Install Docker using the steps provided at <https://docs.docker.com/engine/install/>.
2. Install Docker Compose using the steps provided at <https://docs.docker.com/compose/install/>.

► Install the Protegrity Anonymization API using the following steps:

1. Login to the machine as an administrator to install the Protegrity Anonymization API.
2. Obtain and extract the Protegrity Anonymization API files to a directory on your system.
 - a. Download and extract the *ANON-API_DEB-ALL-64_x86-64_Docker-ALL-64_1.1.0.0.x.tar.gz* file.
 - b. Verify that the following files are available in the package:
 - *ANON-REST-API_1.1.0.0.x.tar.gz*: The files for working with the Protegrity Anonymization REST API.
 - *ANON-STORAGE_1.1.0.0.x.tar.gz*: The files for the Protegrity Anonymization API storage.
 - *ANON-NOTEBOOK_1.1.0.0.x.tar.gz*: The files for the Protegrity Anonymization API notebook.
3. Extract the *ANON-REST-API_1.1.0.0.x.tar.gz* file.
4. Run the following command to load the API container:

```
docker load < ANON-API_1.1.0.0.x.tar.gz
```

5. Verify that the image is successfully loaded using the following command:

```
docker images
```

6. Navigate to the directory where the *ANON-STORAGE_1.1.0.0.x.tar.gz* file is saved.
7. Extract the *ANON-STORAGE_1.1.0.0.x.tar.gz* file.
8. Run the following command to load the storage container:

```
docker load < ANON-STORAGE_1.1.0.0.x.tar.gz
```

9. Verify that the image is successfully loaded using the following command:

```
docker images
```

10. Navigate to the directory where the *ANON-NOTEBOOK_1.1.0.0.x.tar.gz* file is saved.
11. Extract the *ANON-NOTEBOOK_1.1.0.0.x.tar.gz* file.
12. Run the following command to load the storage container:

```
docker load < ANON-NOTEBOOK_1.1.0.0.x.tar.gz
```

13. Verify that the image is successfully loaded using the following command:

```
docker images
```

Note the image ID for the ANON-API, ANON-STORAGE, and ANON-NOTEBOOK containers.

14. Navigate to the directory where the contents of the *ANON-REST-API_1.1.0.0.x.tar.gz* file are extracted.
15. Update the *docker/docker-compose.yaml* file for the configuration that you require, such as, the image ID.
Update the *image* tags for *scheduler*, *anon*, *anondb*, and *pty-worker* with the details of the Anon API Image.

Update the *image* tags for *datanode*, *namenode*, and *workstation* with the details of the Anon-Storage Image.

Note: If required, navigate to *pty-worker* and increase the *replicas* parameter.

An extract of the *docker-compose.yml* file with the details updated is provided here as an example. Update the file based on your configuration.

```
version: "3.1"

services:
  scheduler:
    image: anonapi-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  anon:
    image: anonapi-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  pty-worker:
    image: anonapi-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  anondb:
    image: anonapi-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  namenode:
    image: anonstorage-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  datanode:
    image: anonstorage-1.1.0.0.x:latest

  .
  <existing configuration>
  .

  workstation:
    image: anonworkstation-1.1.0.0.x:latest
    restart: unless-stopped
    hostname: workstation
    container_name: pty-workstation
    # extra_hosts: ##### Uncomment and edit this section for using jupyter-workstation
    # to send request to Protegrity Anonymization-API
    # - "anon.protegrity.com: <IP_of_host_machine>"

  .
  <existing configuration>
  .
```

Note: You can specify the *IMAGE ID* instead of the *REPOSITORY:TAG* for the *image* attribute.

16. Configure the Protegrity Anonymization API to use your custom SSL certificates, if required.

Note: The Protegrity Anonymization API provides its own set of certificates for SSL communication. Complete this step only to use custom certificates. Ensure you have the trusted CA *.pem* file, server certificate, and server key. The server certificate must be signed by the trusted CA.

Only *.pem* files are supported in the Protegrity Anonymization API.

Docker Compose mounts the certificate files from the current directory in the compose file, under the *nginx-proxy* section, as shown here.

```
./cert:/cert/:Z
```

You can mount the directory where you have obtained the trusted CA files or you can replace the certificates in the default directory.

17. Deploy the Protegrity Anonymization API to Docker using the following command.

```
docker-compose -f /path/to/docker-compose.yaml up -d
```

18. Verify that the Docker containers are running using the following command.

```
docker ps
```

19. Update the *hosts* file with an entry of the IP address to *anon.protegrity.com*.

Alternatively, update the *server_name* in the *Nginx.conf* property.

```
server_name anon.protegrity.com;
```

20. Update the host name as provided in the nginx-proxy config host name and as per your certificate.

21. Update the *hosts* file with the following code.

```
<IP of Docker Host> <host name as of nginx.conf>
```

For example,

```
192.168.1.120 anon.protegrity.com
```

The Protegrity Anonymization API is now visible using the Swagger UI. Use the URLs provided here to view the Protegrity Anonymization API using REST.

- Use the following URL to view basic information about the Protegrity Anonymization API.

<https://<Hostname>/>

Note: The default *Hostname* is *anon.protegrity.com*. Ensure that you use the *Hostname* that you provided to access the Protegrity Anonymization API.

- Use the following URL to view the Swagger UI. The various Protegrity Anonymization APIs are visible on this page.

<https://<Hostname>/anonymization/api/v1/ui>

- Use the following URL to view the contractual information for the Protegrity Anonymization API.

<https://<Hostname>/about>

3.3 Installing the Protegrity Anonymization SDK

Complete the steps provided in this section to install the Protegrity Anonymization SDK in your environment. The Protegrity Anonymization SDK communicates with the Protegrity Anonymization REST API to anonymize the data.

3.3.1 Prerequisites for Deploying the Protegrity Anonymization SDK

The Protegrity Anonymization SDK is provided as a wheel file that is installed using pip. Prepare your system for setting up the Protegrity Anonymization SDK. Additionally, ensure that the following prerequisites are met:

- Python 3.6 to 3.7 is installed.
- Protegrity Anonymization REST API is installed.

For more information about the installation steps, refer to the section [Installing the Protegrity Anonymization REST API](#).

Note: If administrator has not updated the DNS entry for ANON REST API service, then map the hostname with the IP address of Anon Service in the `hosts` file of the system.

3.3.2 Installing the Protegrity Anonymization SDK Wheel File

Install the Wheel file provided using pip to use the Protegrity Anonymization SDK.

Note:

Ensure that the prerequisites mentioned in the section [Prerequisites for Deploying the Protegrity Anonymization SDK](#) are met.

1. Create the environment using the following command. In this example, the `sdk_venv` environment is created.

```
python -m venv sdk_venv
```

2. Activate the environment using the following command:

```
source sdk_venv/bin/activate
```

3. Install the Protegrity Anonymization SDK by extracting the contents of the `ANON-SDK_ALL-ALL-64_x86-64_PY-3-64_1.1.0.0.x.tgz` file and running the following command.

```
pip install anonsdk-1.1.0.0.x-py3.whl
```

4. Verify that the `anonsdk` package is installed using the following command:

```
pip list
```

5. Navigate to `https://<IP_address>/sdkapi` to view the Protegrity Anonymization SDK help content.

You can now import and use the Protegrity Anonymization SDK using Python in your environment.

3.3.3 Enabling Custom Certificates From SDK

Protegrity Anonymization API uses certificates for secure communication with the client. You can use the certificates provided by Protegrity or use your own certificates. Complete the configurations provided in this section to use your custom certificates with the SDK.

Before you begin

Ensure that the certificates and keys are in the *.pem* format.

Note: If you want to use the default Protegrity certificates for the Protegrity Anonymization API, then skip the steps to set up the certificates provided in this section.

1. Complete the configuration on the machine where the Protegrity Anonymization API SDK will be used.
 - a. Create a directory that is named *.pty_anon* in the directory from where the SDK will run.
 - b. Create *certs* in the *.pty_anon* directory.
 - c. Create *generated-certs* in the *certs* directory.
 - d. Create *ca-cert* in the *generated-certs* directory.
 - e. Create *cert* in the *generated-certs* directory.
 - f. Create *key* in the *generated-certs* directory.
 - g. Copy the client certificates and key to the respective directories in the *.pty_anon/certs/generated-certs* directory.

The directory structure will be as follows:

```
.pty_anon/certs/generated-certs/ca-cert/CA-xyz-cert.pem
.pty_anon/certs/generated-certs/key/xyz-key.pem
.pty_anon/certs/generated-certs/cert/xyz-cert.pem
```

Make sure that you are using valid certificates. Users can validate the certificates using the commands provided in the section [Working with Certificates](#).

- h. Create a *config.yaml* file in the *.pty_anon* directory with the following Ingress Endpoint defined under *CLUSTER_ENDPOINT*.

```
HTTP_SINK_SOURCE:
  CLUSTER_ENDPOINT: https://anon.protegrity.com:443
```

Note: Ensure that you replace *anon.protegrity.com* with your host name specified in *values.yaml*.

2. Updating the *hosts* file.
 - a. Login to the machine where the Protegrity Anonymization API SDK will be used.
 - b. Update the *hosts* file with the following code according to your setup.

For Kubernetes:

```
<LB-IP of Ingress> <host defined for ingress in values.yaml>
```

For Docker:

```
<LB-IP of Ingress> <server_name defined in nginx.conf>
```

For example,

```
XX.XX.XX.XX anon.protegrity.com
```

The URL can now be used while creating the Connection Object in the SDK, such as, `conn = anon sdk.Connection("https://anon.protegrity.com/")`.

3.3.4 Verifying the Installation

Run the sample script provided in this section to test that the Protegrity Anonymization SDK installation is working.

```
import pandas as pd
import anonsdk as asdk

d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male',
'Female'],
      'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-
cleaners', 'Prof-specialty', 'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-
specialty'],
      'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
      'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White',
'White'],
      'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-
spouse', 'Married-civ-spouse', 'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-
spouse', 'Never-married'],
      'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters',
'9th', 'HS-grad', 'Masters'],
      'native-country': ['United-States', 'United-States', 'United-States', 'United-States',
'Cuba', 'United-States', 'Jamaica', 'United-States', 'United-States'],
      'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private',
'Private', 'Private', 'Self-emp-not-inc', 'Private'],
      'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K',
'>50K'],
      'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5]}

df = pd.DataFrame(data=d)
treeGen = {'lvl0': [11, 13, 14, 15, 27, 28, 20],
           'lvl1': ['<15', '<15', '<15', '<20', '<30', '<30', '<25'],
           'lvl2': ['<20', '<20', '<20', '<30', '<30', '<30', '<30']}
conn = asdk.Connection('https://anon.protegrity.com/')

e = asdk.AnonElement(conn, df)

e['gender'] = asdk.Redact()
# Also works on attribute index
# e[0] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", " Might be
<30"])

# e["race"] = asdk.Gen_Mask(maskchar="*")
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
# e["bmi"] = asdk.MicroAgg(asdk.AggregateFunction.Mean)
e.config.k = asdk.K(2)

e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TCclose(tfactor=0.2)

e.config['maxSuppression'] = 0.7
e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)

job = asdk.anonymize(e)

result = job.result()
if result.df is not None:
    print("Anon Dataframe.")
    print(result.df.head())

print(job.utilityStat()) # Utility Metrics
print(job.riskStat())
```

Chapter 4

Using Protegrity Anonymization API

[4.1 Creating Protegrity Anonymization API Requests](#)

[4.2 Working with the Protegrity Anonymization APIs](#)

[4.3 Building the Anonymization request](#)

[4.4 Using the Workstation Feature](#)

The information provided in this section explains the various APIs provided by the Protegrity Anonymization API and the method for creating and running Protegrity Anonymization API requests.

4.1 Creating Protegrity Anonymization API Requests

This section walks you through the process of creating Protegrity Anonymization API requests to anonymize your data. It describes the steps for using the API and creating the Protegrity Anonymization API request.

A general overview of the process you need to follow to anonymize the data is shown in the following figure:

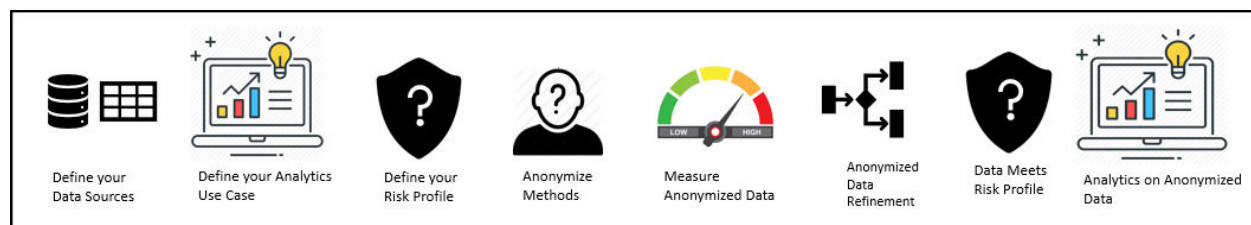


Figure 4-1: Protegrity Anonymization API Process

1. Identify the data set than needs to be anonymized.
2. Analyze and classify the various fields available in the data set. The following classifications are available:
 - Direct Identifiers
 - Quasi-Identifier
 - Sensitive Attributes
 - Non-Sensitive Attributes
3. Determine the use case by specifying the data that is required for further analysis.
4. Specify the quasi-identifiers and other fields that are not required in the data set
5. Specify the required anonymization methods for the data. The following methods are available:
 - Generalization
 - Micro-Aggregation
6. Specify the acceptable risk levels for the data fields for measuring the statistic before the anonymization job.

7. Verify that the anonymized data satisfies the acceptable risk threshold level.
8. Measure the quality of the anonymized data by comparing it with the original data. If the quality does not meet standards, then work on the data or drop the output.
9. Save the anonymized data to an output file.

The anonymized data can now be used for further analysis and as input for machine learning softwares.

4.2 Working with the Protegrity Anonymization APIs

The various APIs provided with the Protegrity Anonymization API are described here. You can use the REST APIs or install the SDK and use it by importing the *anonsdk* module.

4.2.1 Understanding the Protegrity Anonymization REST APIs

The following APIs are available with Protegrity Anonymization REST API. You can run these APIs using the command line with the *curl* command or submit the command using the default Swagger UI provided or a tool, such as, Postman.

4.2.1.1 Anonymization Functions

These functions are used to run the anonymization job.

4.2.1.1.1 Anonymize

Use this API to start an anonymize operation.

For more information about the anonymize API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Scheduling/start_anon

Ensure that you complete the following before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/anonymize

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/anonymize' \ -H 'accept: application/json' \ -H
'Content-Type: application/json' \ -d '{ "attributes": [ { "classificationType": "Quasi Identifier", "dataTransformationType":
"Generalization", "dataType": "String", "generalization": { "hierarchyType": "Rule", "rule": { "masking": { "maskChar":
"*", "maskOrder": "Right To Left", "maxDomainSize": 2 } }, "type": "Masking Based" }, "name": "age" } ], "config":
{ "maxSuppression": 0.01 }, "privacyModel": { "k": { "kValue": 50 } }, "source": { "file": { "name": "samples/adult.csv",
"props": { "sep": ";" } }, "type": "File" }, "target": { "file": { "name": "anon-adult-e1.csv" }, "type": "File" } }'
```

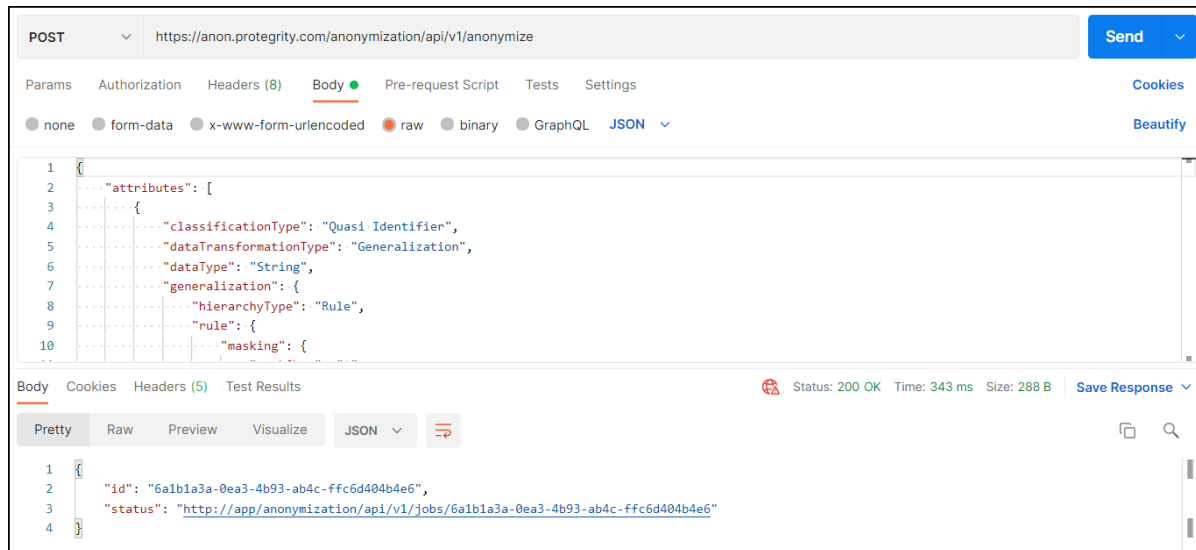


Figure 4-2: Anonymize API

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization REST API](#).

Note: When you run the job, an empty destination file is created. This file is created during processing for verifying the necessary destination permissions. Avoid using this file till the anonymization job is complete.

Ensure that the anonymized data file and the logs generated are moved to a different system before deleting your environment.

If the source file is larger than the maximum limit that is allowed on the Cloud environment, then run the anonymization request with `"additional_properties": { "single_file": "no" }`.

4.2.1.1.2 Apply Anonymize

Use this API as a template to anonymize additional entries. Using this API you can use the existing configuration to process additional data. This is especially useful in machine learning for training the system to anonymize new data points.

Note: In this API, privacy model parameters are ignored while performing the anonymization for the new entry.

For more information about the apply anonymize API, refer to the section <https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Scheduling/applyAnon>

Ensure that you complete the following before submitting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.
- Verify that the anonymization job exists.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

`/anonymize/{jobId}/apply`

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/anonymize/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/apply?mode=normal' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '[ { "gender": "Male", "age": "39", "race": "White", "income": "<=50K", "bmi": "12.5" } ]'
```

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization REST API](#).

4.2.1.1.3 Measure

Use this API to measure or obtain anonymization result statics for different configurations before the actual anonymization job.

For more information about the anonymize API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Scheduling/measure_anon

Ensure that you complete the following checks before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/measure

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/measure' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "element": { "attributes": [ { "classificationType": "Quasi Identifier", "dataTransformationType": "Generalization", "dataType": "String", "generalization": { "hierarchyType": "Rule", "rule": { "masking": { "maskChar": "*", "maskOrder": "Right To Left", "maxDomainSize": 2 } }, "type": "Masking Based" }, "name": "age" } ], "config": { "maxSuppression": 0.01 }, "privacyModel": { "k": { "kValue": 50 } }, "source": { "file": { "name": "samples/adult.csv", "props": { "sep": ";" }, "type": "File" }, "target": { "file": { "name": "anon-adult-e1.csv" }, "type": "File" } }, "includes": [ "string" ], "nodeConf": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" } } }
```

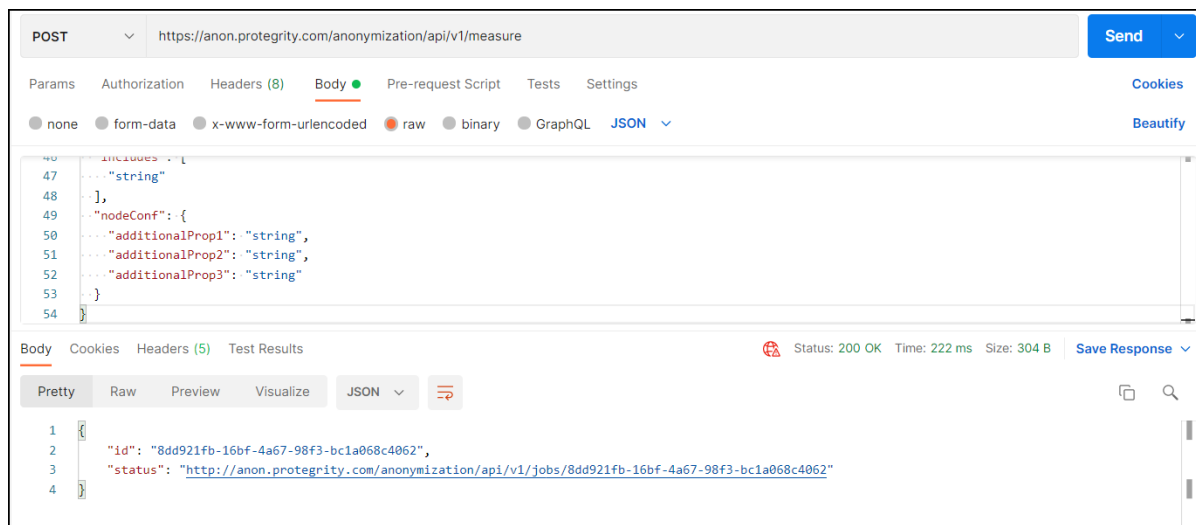


Figure 4-3: Measure API

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization REST API](#).

4.2.1.2 Task Monitoring APIs

These APIs are used to monitor the anonymization job. Use these APIs to obtain the job status, retrieve a job, and abort a job.

4.2.1.2.1 Get Job IDs

Use this API to get the job IDs of the last 20 anonymization operations run. You can then use the required job ID with the other APIs to work with the anonymization job. For more information about the job ID API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Monitoring/all_jobs

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs

Method

GET

Parameters

- *Running*: Displays a list of jobs that are running.
- *InQueue*: Displays a list of jobs that are queued.
- *Completed*: Displays a list of jobs that are complete.

Sample Request

`curl -X 'GET' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs?status=Running|InQueue|Completed&limit=10'`

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `https://anon.protegrity.com/anonymization/api/v1/jobs?status=Running|InQueue|Completed&limit=10`
- Query Params:**

KEY	VALUE	DESCRIPTION
status	Running InQueue Completed	
limit	10	
Key	Value	Description
- Response:** Status: 200 OK, Time: 844 ms, Size: 7.9 KB
- Body (JSON):**

```

{
  "completed": {
    "stat": {
      "result": {
        "exploratory": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/exploratory/",
        "risk": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/riskmetric/",
        "utility": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/utility/"
      },
      "source": {
        "exploratory": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/exploratory/",
        "risk": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/riskmetric/",
        "utility": "http://app/anonymization/api/v1/jobs/6a1ba3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/utility/"
      }
    }
  }
}

```

Figure 4-4: Jobs API

4.2.1.2.2 Get Job Status

Use this API to get the status of an anonymize operation that is running, in queue, or complete. It shows the percentage of job completed. Use the information provided here to monitor if a job is running or stalled. For more information about the job status API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Monitoring/get_job_status

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs/{jobId}

In the URL, the jobId is a string that specifies the ID of the job of which the status must be retrieved. The jobId is obtained in the request body when an anonymization job is scheduled.

Method

GET

Sample Request

curl -X 'GET' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6'

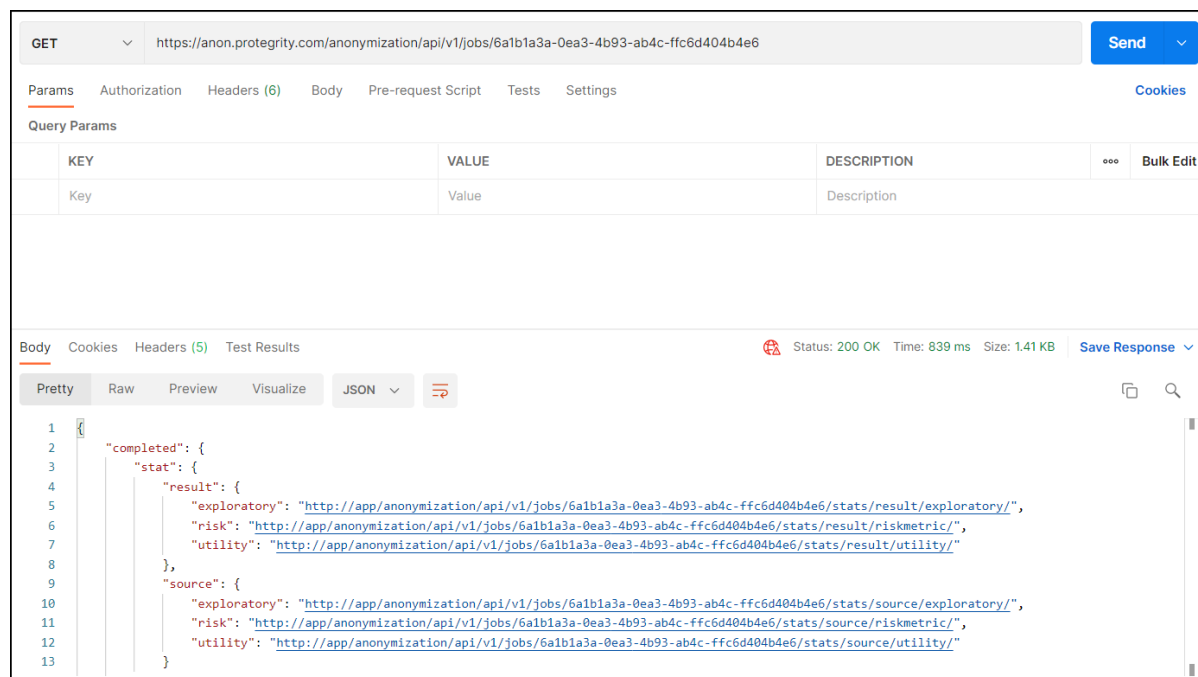


Figure 4-5: Get Job Status API

4.2.1.2.3 Get Metadata

Use this API to retrieve the metadata for the existing job. This API is useful when you need to view the configuration available for a job. It displays the fields, configuration, and the data that is used to run the anonymization job.

For more information about the meta API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Monitoring/get_job_metadata

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs/{jobId}/meta

In the URL, the `jobId` is a string that specifies the ID of the job for which the metadata must be retrieved.

Method

GET

Sample Request

```
curl -X 'GET' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/meta?include=*
```

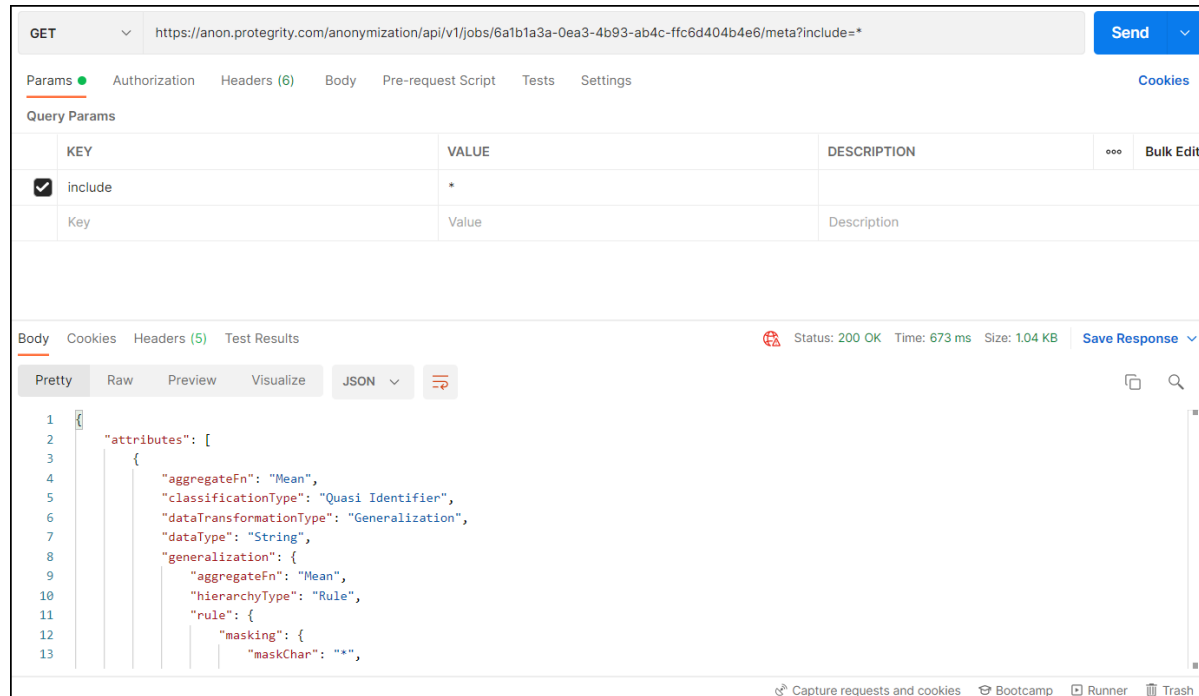


Figure 4-6: Jobs Metadata API

4.2.1.2.4 Abort

Use this API to abort a running anonymization job. You can abort jobs if you need to modify the parameters or if the job is stalled or taking too much time or resources to process. For more information about the abort API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Monitoring/abort_job

Note: After aborting the task, it might take time before all the running processes are stopped.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

`/jobs/{jobId}/abort`

In the URL, the `jobId` is a string that specifies the ID of the job that must be aborted.

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/ce5793b2-4434-4104-acff-26651de018d8/abort'
```

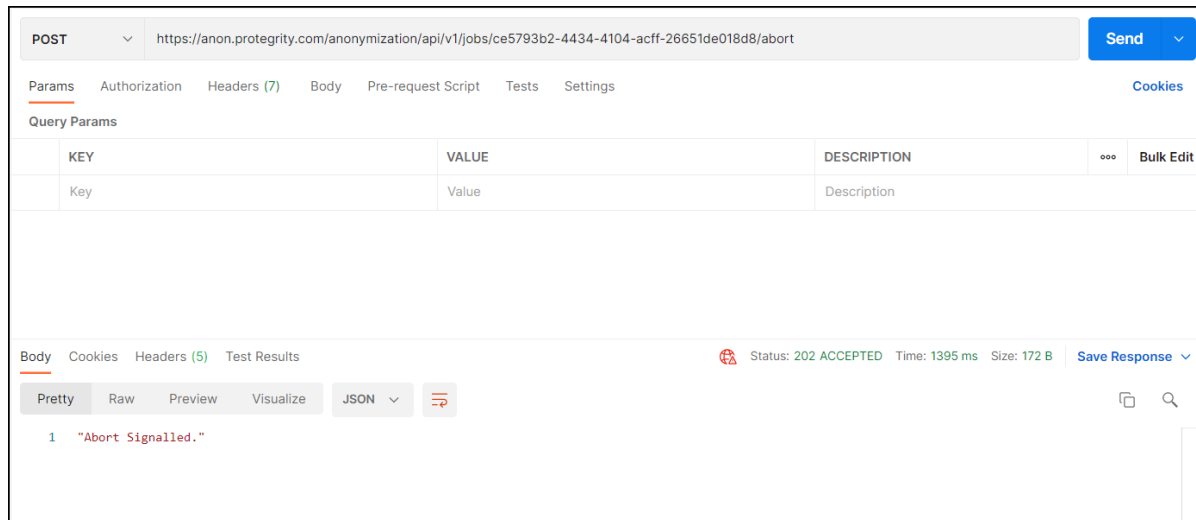


Figure 4-7: Abort Job API

4.2.1.2.5 Delete

Use this API to delete an existing job that is no longer required. For more information about the delete API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Task%20Monitoring/delete_job

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs/{jobId}

In the URL, the jobId is a string that specifies the ID of the job that must be deleted.

Method

DELETE

Sample Request

`curl -X 'DELETE' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/842aac73-1ec9-4450-af26-935e33791216'`

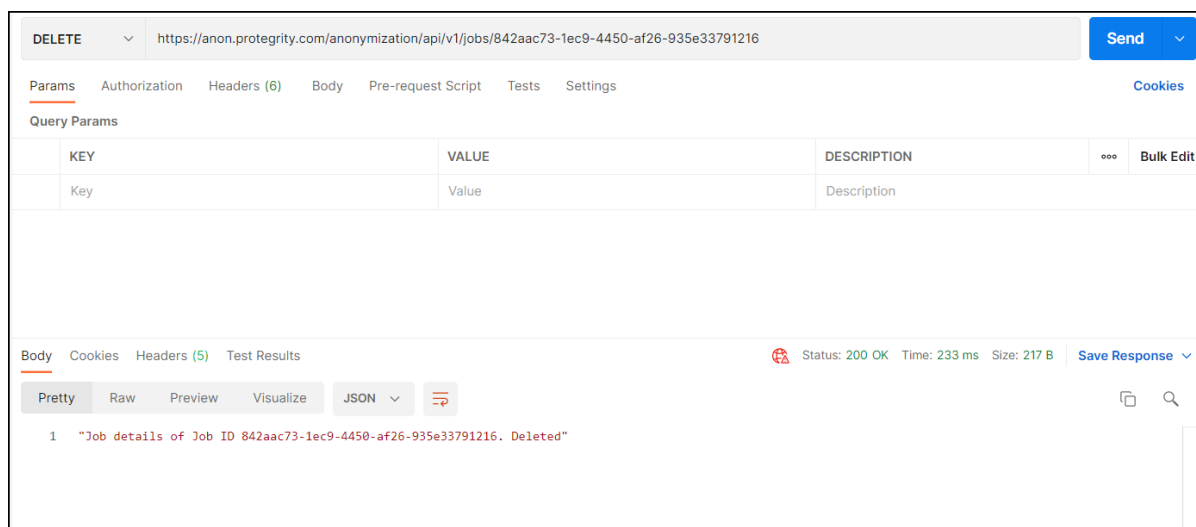


Figure 4-8: Delete Job API

4.2.1.3 Statistics APIs

These APIs are used to obtain information about the anonymization data. Use these APIs to obtain the risk and utility information about the anonymization. The user needs to access these APIs to measure the utility benefits and risk of publishing the anonymized data. If these configurations are not satisfactory, then the user can re-submit the anonymization job after modifying some parameters based on these results.

4.2.1.3.1 Get Exploratory Statistics

Use this API to obtain data distribution statistics about a completed anonymization job. The information includes information about both, the source and the target distribution. For more information about the exploratory statistic API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Stats/exploratory_stats

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs/{jobId}/stats/{type}/exploratory/

In the URL, the *jobId* is a string that specifies the ID of the job of which the information must be retrieved.

In the URL, the *type* is a string that specifies *source* for evaluating the source file and *result* for evaluating the output file.

Method

GET

Sample Request

```
curl -X GET 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/exploratory/?attr=*
```

```
curl -X GET 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/exploratory/?attr=*
```

This provides the data distribution of the attribute, which is all unique values of an attribute and its occurrence count. This can be used to build data histogram of all attributes in the data set.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/exploratory/?attr=*`
- Query Params:**

KEY	VALUE	DESCRIPTION
attr	*	
Key	Value	Description
- Status:** 200 OK, Time: 845 ms, Size: 5.27 KB
- Response Body (JSON):**

```
{
  "attributes": [
    {
      "distribution": [
        {
          "data": "17",
          "frequency": 328
        },
        {
          "data": "18",
          "frequency": 447
        }
      ]
    }
  ]
}
```

Figure 4-9: Source Exploratory

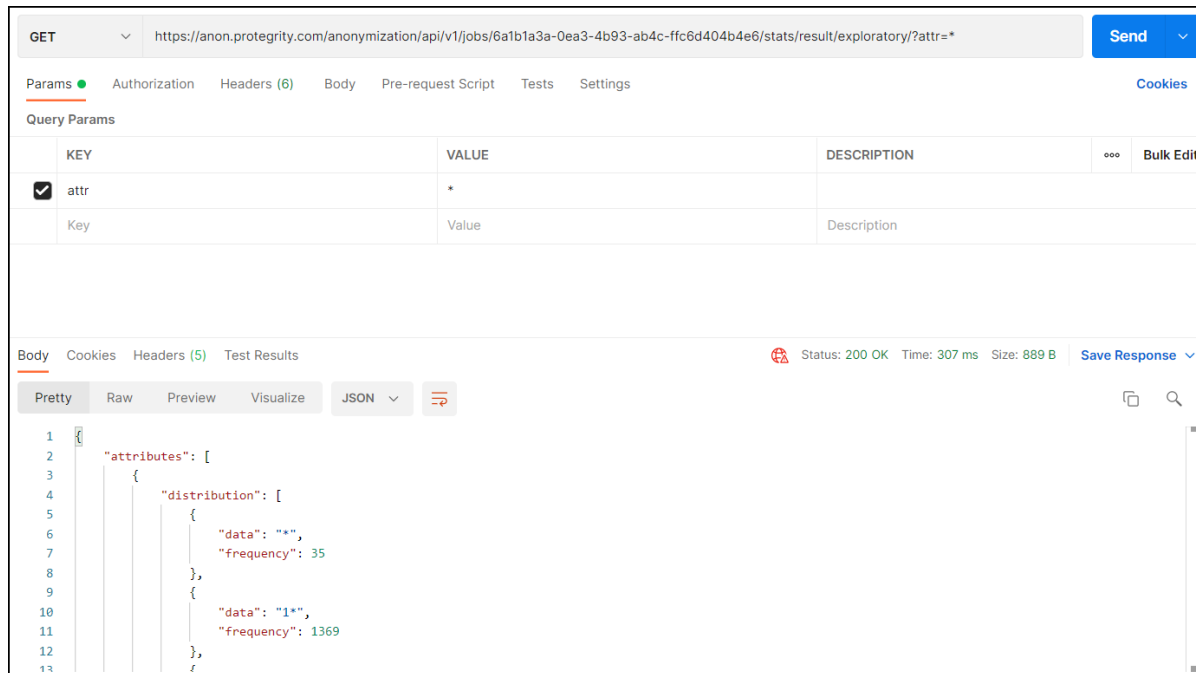


Figure 4-10: Result Exploratory

4.2.1.3.2 Get Risk Metric

Use this API to ascertain the risk of the anonymized data. It shows the risk of the data against attacks such as journalist, marketer, and prosecutor. For more information about the risk metric API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Stats/risk_metric

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/jobs/{jobId}/stats/{type}/riskmetric/

In the URL, the *jobId* is a string that specifies the ID of the job of which the information must be assessed for risk.

In the URL, the *type* is a string that specifies *source* for evaluating the source file and *result* for evaluating the output file.

Method

GET

Sample Request

```
curl -X 'GET' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/riskmetric/'
```

```
curl -X 'GET' \ 'https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/riskmetric/'
```

The following values appear:

- **avgRecordIdentification:** This value displays the average probability for identifying a record in the anonymized dataset. The risk is higher when the value is closer to the value 1.
- **maxProbabilityIdentification:** This displays the maximum probability value that a record can be identified from the dataset. The risk is higher when the value is closer to the value 1.
- **riskAboveThreshold:** This value displays the number of records that are at a risk above the risk threshold. The default threshold is 10%. The threshold is the maximum value set as a boundary. Any values beyond the threshold are a risk and might be easy to identify. For this result, the value 0 is preferred.

GET <https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/source/riskmetric/> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1057 ms Size: 554 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "journalist": {
3     "avgRecordIdentification": 0.0024,
4     "maxProbabilityIdentification": 1.0,
5     "riskAboveThreshold": 0.0009
6   },
7   "marketer": {
8     "avgRecordIdentification": 0.0024,
9     "maxProbabilityIdentification": 0.0,
10    "riskAboveThreshold": 0.0
11  },
12  "prosecutor": {
13    "avgRecordIdentification": 0.0024
14  }
15 }
```

Figure 4-11: Source Risk Metric

GET <https://anon.protegrity.com/anonymization/api/v1/jobs/6a1b1a3a-0ea3-4b93-ab4c-ffc6d404b4e6/stats/result/riskmetric/> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 236 ms Size: 554 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "journalist": {
3     "avgRecordIdentification": 0.0003,
4     "maxProbabilityIdentification": 0.0286,
5     "riskAboveThreshold": 0.0
6   },
7   "marketer": {
8     "avgRecordIdentification": 0.0003,
9     "maxProbabilityIdentification": 0.0,
10    "riskAboveThreshold": 0.0
11  },
12  "prosecutor": {
13    "avgRecordIdentification": 0.0003
14  }
15 }
```

Figure 4-12: Result Risk Metric

4.2.1.3.3 Get Utility Statistics

Use this API to check the usability of the anonymized data. For more information about the utility statistics API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Stats/utility_stats

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

`/jobs/{jobId}/stats/{type}/utility/`

In the URL, the *jobId* is a string that specifies the ID of the job of which the information must be analyzed.

In the URL, the *type* is a string that specifies *source* for evaluating the source file and *result* for evaluating the output file.

Method

GET

Sample Request

```
curl -X GET https://anon.protegrity.com/anonymization/api/v1/jobs/<job_id>/stats/source/utility/
curl -X GET https://anon.protegrity.com/anonymization/api/v1/jobs/<job_id>/stats/result/utility/
```

The following values appear:

- **ambiguity**: This value displays how well a record is hidden in all the records. This captures the ambiguity of records.
- **average_class_size**: This measures the average size of groups of indistinguishable records. A smaller class size is more favourable for retaining the quality of the information. A larger class size increases anonymity at the cost of quality.
- **discernibility**: This measures the size of groups of indistinguishable records with penalty for records which have been completely suppressed. Discernibility metrics measures the cardinality of the equivalent class. Discernibility metrics considers only the number of records in the equivalent class and does not capture information loss caused by generalization.
- **generalization_intensity**: Data transformation from the original records to anonymity is performed using generalization and suppression. This measures the concentration of generalization and suppression on attribute values.
- **infoLoss**: This value displays the probability of information lost with the data transformation from the original records. Larger the value, lesser the quality for further analysis.

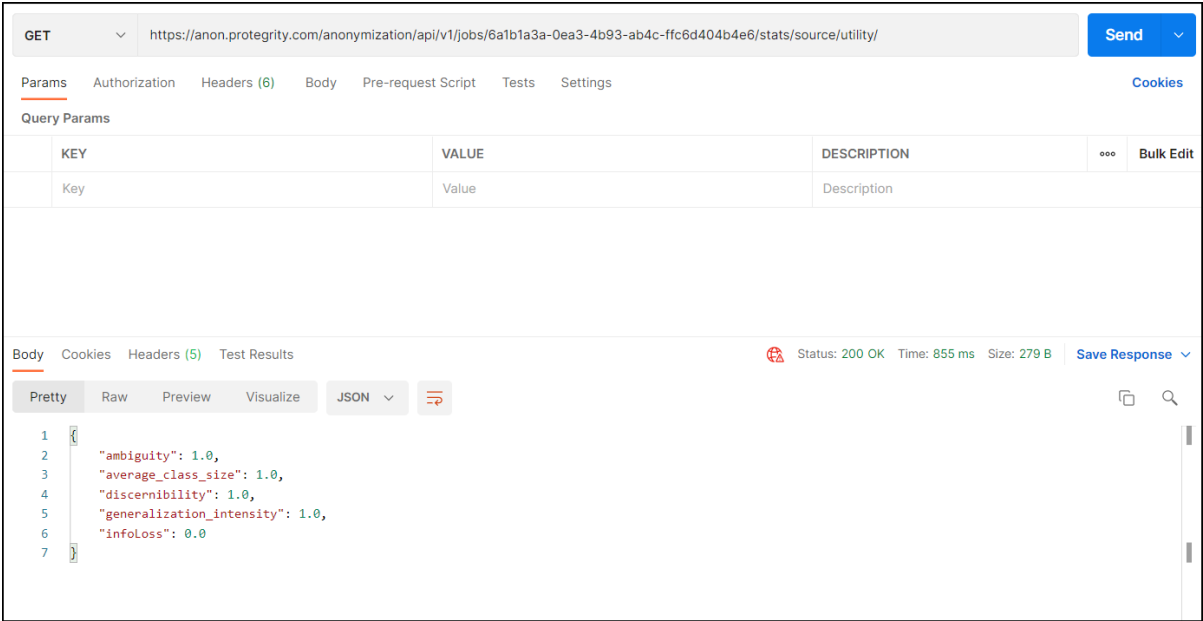


Figure 4-13: Source Utility

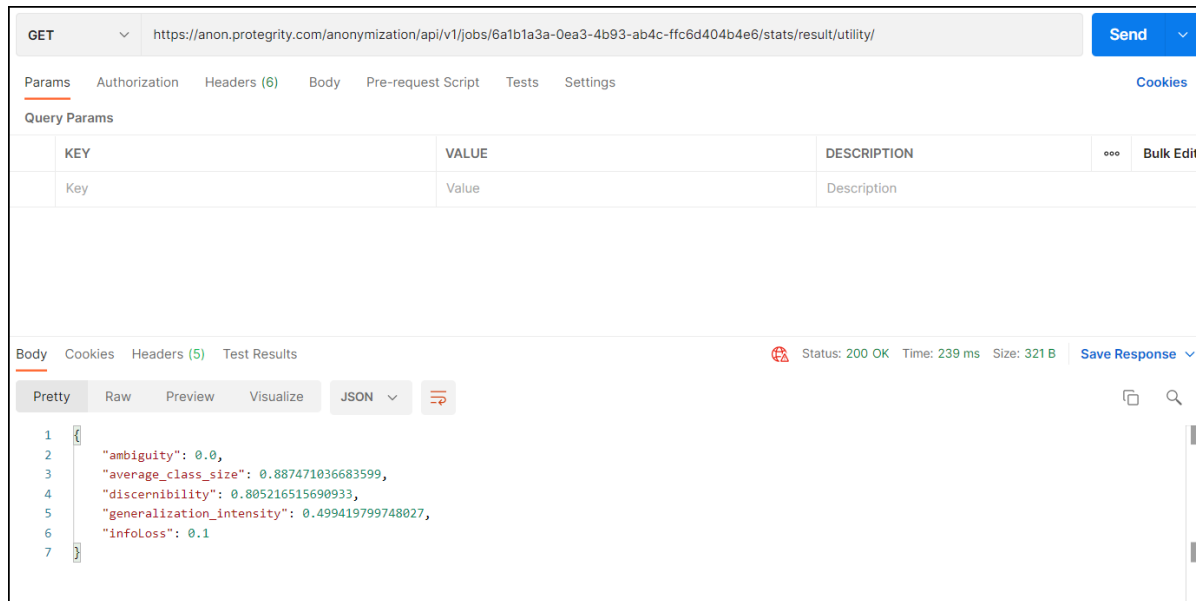


Figure 4-14: Result Utility

4.2.1.4 Detection APIs

The following APIs are available for analyzing and classifying data in the Protegrity Anonymization API.

4.2.1.4.1 Get Data Domains

Use this API to obtain a list of data domains supported.

For more information about the API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Detect/get_data_domains.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/datadomains

Method

GET

Sample Request

`curl -X GET 'https://anon.protegrity.com/anonymization/api/v1/datadomains'`

The following is the list of data domains available for auto anonymization in this release are listed here:

- age
- gender
- ccn
- phoneno
- email
- state
- country
- postcode
- diagnosis

- name

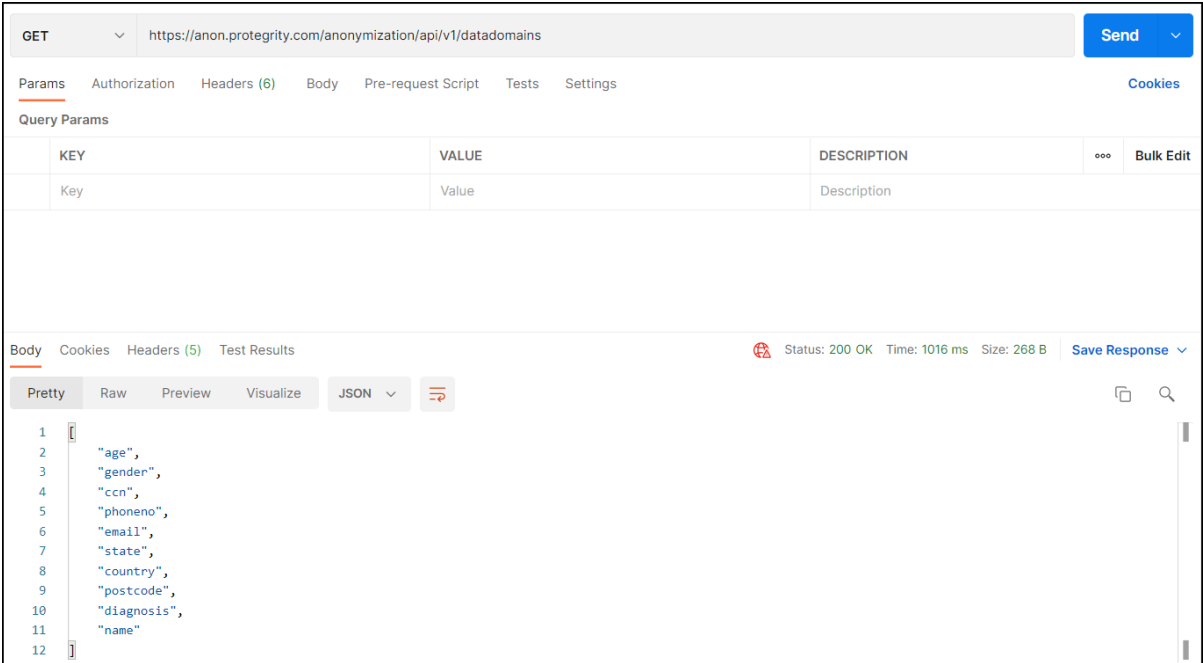


Figure 4-15: Get Data Domains

4.2.1.4.2 Detect Anonymization Information

Use this API to detect the data domain, classification type, hierarchy, and privacy models for the data set.

For more information about the API, refer to the section <https://anon.protegrity.com/anonymization/api/v1/ui/#/Detect/detect>.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/detect

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/detect' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "attributes": [ { "classificationType": "Quasi Identifier", "dataTransformationType": "Generalization", "dataType": "String", "generalization": { "hierarchyType": "Rule", "rule": { "masking": { "maskChar": "*", "maskOrder": "Right To Left", "maxDomainSize": 2 } }, "type": "Masking Based" }, "name": "age" } ], "config": { "maxSuppression": 0.01 }, "privacyModel": { "k": { "kValue": 50 } }, "source": { "file": { "name": "samples/adult.csv", "props": { "sep": ";" } }, "type": "File" }, "target": { "file": { "name": "anon-adult-e1.csv" }, "type": "File" } }'
```

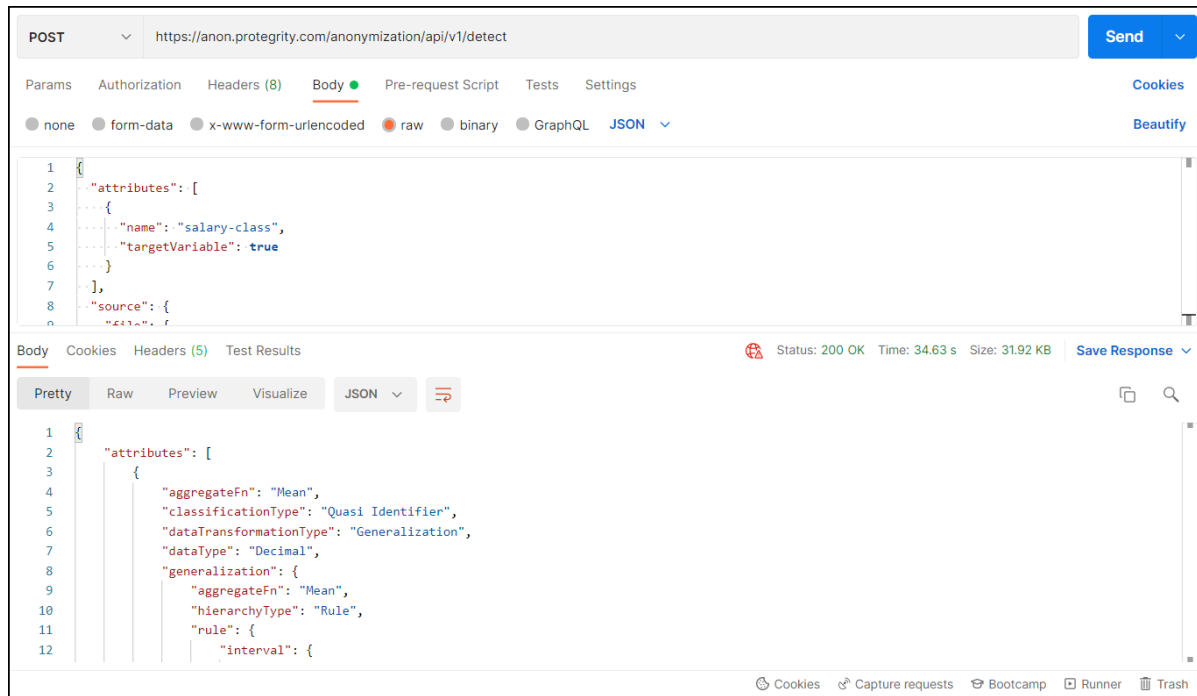


Figure 4-16: Detect API

4.2.1.4.3 Detect Classification

Use this API to detect the classification that will be used for the anonymization operation. Accordingly, you can modify the classification to match your requirements.

For more information about the API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Detect/detect_qi.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/detectClassification

Method

POST

Sample Request

```
curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/detectClassification' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "attributes": [ { "classificationType": "Quasi Identifier", "dataType": "String", "name": "age" } ], "source": { "file": { "name": "samples/adult.csv", "props": { "sep": ";" }, "type": "File" } }'
```

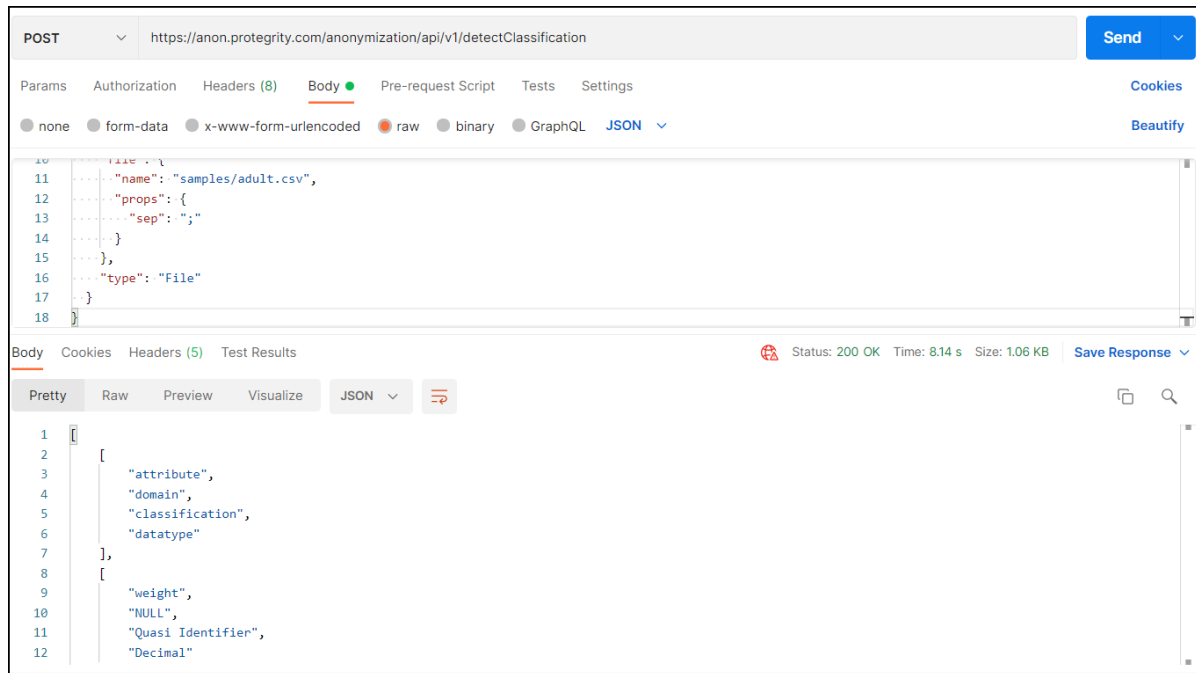


Figure 4-17: Detect Classification API

4.2.1.4.4 Detect Hierarchy

Use this API to detect the hierarchy type that will be used for the anonymization operation.

For more information about the API, refer to the section https://anon.protegrity.com/anonymization/api/v1/ui/#/Detect/detect_hierarchy.

Base URL

<https://anon.protegrity.com/anonymization/api/v1>

Path

/detectHierarchy

Method

POST

Sample Request

```

curl -X 'POST' \ 'https://anon.protegrity.com/anonymization/api/v1/detectHierarchy' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "attributes": [ { "classificationType": "Quasi Identifier", "dataType": "String", "name": "age" } ], "source": { "file": { "name": "samples/adult.csv", "props": { "sep": ";" } }, "type": "File" } }'

```

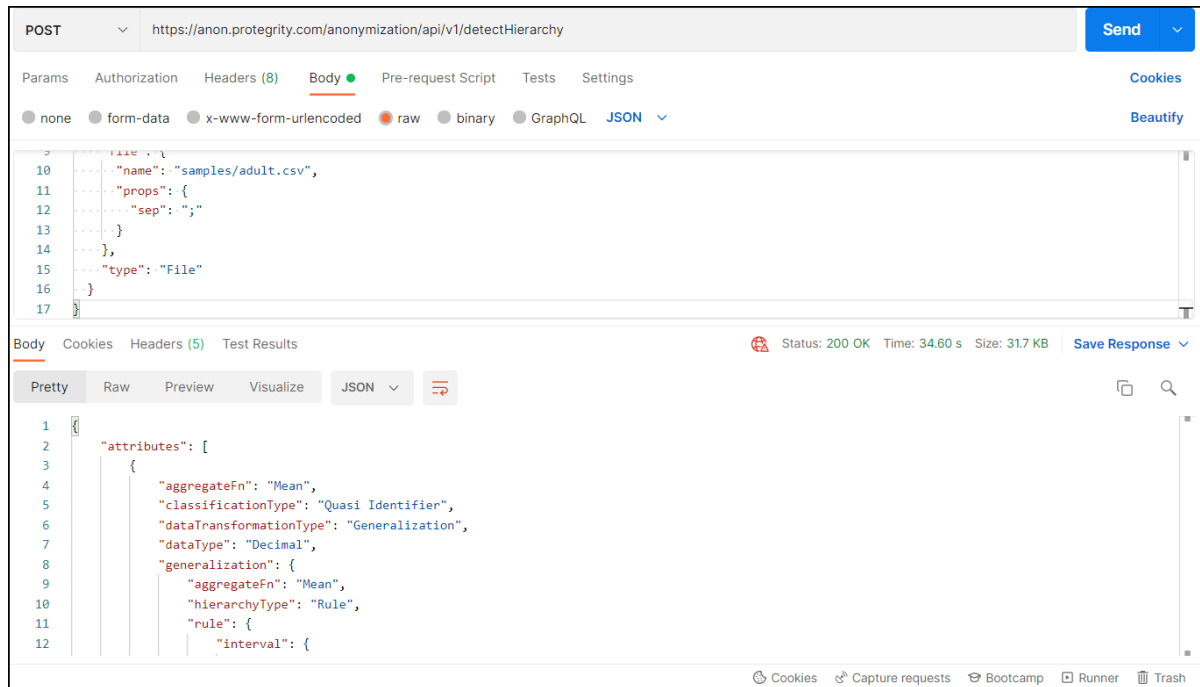


Figure 4-18: Detect Hierarchy API

4.2.2 Understanding the Protegrity Anonymization SDK

The following APIs are available with Protegrity Anonymization SDK. You can import the Protegrity Anonymization SDK in your SDK environment, pass the required parameter and data to the Protegrity Anonymization SDK API functions, and retrieve work with the anonymized output.

4.2.2.1 Understanding the AnonElement object

The *AnonElement* is an essential part of the Protegrity Anonymization SDK. It holds all information that is required for processing the anonymization request. The *AnonElement* is a part of the *anonsdk* package.

The Protegrity Anonymization SDK processes a Pandas dataframe to anonymize data using the Protegrity Anonymization REST API. It is the *AnonElement* that accepts the parameters and passes the information to the REST API. The *AnonElement* accepts the connection to the REST API, the pandas dataframe with the data that must be processed, and the optionally the source location for processing the request.

For more information about building the anonymization request, refer to the section [Building the request using SDK](#).

4.2.2.2 Anonymization Functions

These functions are used to run the anonymization job.

4.2.2.2.1 Anonymize

Use this API to start an anonymize operation.

Ensure that you complete the following before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.

- Verify that you have imported the Pythonic SDK, for example, *import anonsdk as asdk*.

Function

`anonymize(anon_object, target_datastore, force, mode)`

Parameters

anon_object: The object with the configuration for performing the anonymization request.

target_datastore: The location to store the anonymized result.

force: The boolean value to force the operation. Acceptable values: *True* and *False*. Set this flag to *true* to resubmit the same anonymized job without any modification.

mode: The value to enable auto anonymization. Acceptable value: *auto*. Do not include this parameter to skip auto anonymization.

For more information about using the Auto Anonymization, refer to the section [Using the Auto Anonymizer](#).

Return Type

A job object with which the task monitoring and task statistics can be obtained.

Sample Request

Without auto anonymization:

```
job = asdk.anonymize(anon_object,target_datastore ,force=True)
```

With auto anonymization

```
job = asdk.anonymize(anon_object,target_datastore ,force=True,mode="auto")
```

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization SDK](#).

Note: When you run the job, an empty destination file is created. This file is created during processing for verifying the necessary destination permissions. Avoid using this file till the anonymization job is complete.

Ensure that the anonymized data file and the logs generated are moved to a different system before deleting your environment.

If the source file is larger than the maximum limit that is allowed on the Cloud environment, then run the anonymization request with *"additional_properties": { "single_file": "no" }*.

If you want to bypass the Anon-Storage, then you can disable the pods by setting the *pty_storage* flag to *False*. For example, use the following code to run the anonymization request without using the storage pods

```
job=asdk.anonymize(anon_object, pty_storage=False)
```

```

In [ ]: import pandas as pd
import anonsdk as asdk

In [ ]: d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Female'],
            'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-cleaners', 'Prof-specialty',
                           'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-specialty'],
            'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
            'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White', 'White'],
            'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-spouse',
                              'Married-civ-spouse',
                              'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-spouse', 'Never-married'],
            'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters', '9th', 'HS-grad',
                          'Masters'],
            'native-country': ['United-States', 'United-States', 'United-States', 'United-States', 'Cuba', 'United-States',
                              'Jamaica', 'United-States', 'United-States'],
            'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private', 'Private', 'Private',
                          'Self-emp-not-inc', 'Private'],
            'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K', '>50K'],
            'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }

In [ ]: df = pd.DataFrame(data=d)

In [ ]: treeGen = {'lvl0': [11, 13, 14, 15, 27, 28, 20],
                  'lvl1': ["<15", "<15", "<15", "<20", "<30", "<30", "<25"],
                  'lvl2': ["<20", "<20", "<20", "<30", "<30", "<30", "<30"]}

In [ ]: conn = asdk.Connection('https://anon.protegrity.com/')

In [ ]: e = asdk.AnonElement(conn, df)

In [ ]: e.infer(targetVariable='income')

In [ ]: e.describe()

In [ ]: e['gender'] = asdk.Redact()
#Also works on attribute index
#e[0] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", "Might be <30"])

# e["race"] = asdk.Gen_Mask(maskchar="*")
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
#e["bmi"] = asdk.MicroAgg(asdk.AggregateFunction.Mean)
e.config.k = asdk.K(2)

e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TClose(tfactor=0.2)

e.config['maxSuppression'] = 0.7

In [ ]: e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)

In [ ]: e.describe()

In [ ]: job = asdk.anonymize(e)

```

Figure 4-19: SDK Anonymization Job

4.2.2.2.2 Using Infer to Anonymize

Use the *Infer* API to start auto-detecting the data-domain, classification type, hierarchies, and anonymization configuration in the Protegrity Anonymization SDK. Any user-defined configuration, such as, QI attribute assignments, hierarchy, and K value, are retained and considered while performing the auto anonymization.

Ensure that you complete the following checks before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.
- Verify that you have imported the Pythonic SDK, for example, *import anonsdk as asdk*.

Function

infer(targetVariable)

Parameters

targetVariable: The field specified here is used as a focus point for performing the anonymization.

Return Type

It returns an *anon* element with all the detected classifications and hierarchies generated.

Sample Request

`e.infer(targetVariable='income')`

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization SDK](#).

```
In [ ]: import pandas as pd
import anonsdk as asdk

In [ ]: d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Female'],
'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-cleaners', 'Prof-specialty',
'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-specialty'],
'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White', 'White'],
'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-spouse',
'Married-civ-spouse',
'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-spouse', 'Never-married'],
'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters', '9th', 'HS-grad',
'Masters'],
'native-country': ['United-States', 'United-States', 'United-States', 'United-States', 'Cuba', 'United-States',
Jamaica', 'United-States', 'United-States'],
'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private', 'Private', 'Private',
'Self-emp-not-inc', 'Private'],
'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K', '>50K'],
'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }

In [ ]: df = pd.DataFrame(data=d)

In [ ]: treeGen = {'lvl0': [11, 13, 14, 15, 27, 28, 20],
'lvl1': ['<15', '<15', '<15', '<20', '<30', '<30', '<25'],
'lvl2': ['<20', '<20', '<20', '<30', '<30', '<30']}

In [ ]: conn = asdk.Connection('https://anon.protegrity.com/')

In [ ]: e = asdk.AnonElement(conn, df)

In [ ]: e.infer(targetVariable='income')

In [ ]: e.describe()

In [ ]: e['gender'] = asdk.Redact()
#Also works on attribute index
#e[0] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", "Might be <30"])

# e["race"] = asdk.Gen_Mask(maskchar="*")
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
#e["bmi"] = asdk.MicroAgg(asdk.AggregateFunction.Mean)
e.config.k = asdk.K(2)

e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TClose(tfactor=0.2)
e.config['maxSuppression'] = 0.7

In [ ]: e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)

In [ ]: e.describe()

In [ ]: job = asdk.anonymize(e)
```

Figure 4-20: Infer Job

Note: You can use *e.measure()* to modify the request and view different outcomes of the result set.

For more information about the *measure* API, refer to the section [Measure](#).

4.2.2.3 Task Monitoring APIs

These APIs are used to monitor the anonymization job. Use these APIs to obtain the job status, retrieve a job, and abort a job.

4.2.2.3.1 Get Job Status

Use this API to get the status of an anonymize operation that is running. It shows the percentage of job completed. Use the information provided here to monitor if a job is running or stalled.

Function

`status()`

Parameters

None

Return Type

A string with the status information in the JSON format.

completed: This is information about the job, such as, data, statistics, summary, and time spent.

id: This is the job ID.

info: This is information about the job being processed, such as, the source and attributes for the job.

running: This is the completion status of the jobs being processed. It shows the percentage of the job completed.

status: This is the status of the job, such as, *running* or *completed*.

Note: This API displays all the status of the job. To obtain the ID of a job, use *job.id()*.

Sample Request

`job.status()`

```
In [10]: job = asdk.anonymize(e)

In [11]: job.id()

Out[11]: '94c9595d-385d-4c9d-b8fe-bb3e205ee0eb'

In [12]: job.status()

Out[12]: {'completed': {'data': None,
                        'stat': None,
                        'summary': None,
                        'time_spent': '9 sec(s)'},
          'id': '94c9595d-385d-4c9d-b8fe-bb3e205ee0eb',
          'info': "Source : {'EC Min Size': '[ 1, 1, 1, 1, 1]', 'EC Max Size': '[ 1, 1, "
                  "1, 1, 1]', 'EC Count': '9'}. Attribute Generalization Levels: "
                  "'age': 1, 'race': 0, 'bmi': 1}. Info Loss : 0.09563959075709949. "
                  "Outlier Records suppressed : 5. Result : {'EC Min Size': '[ 4]', "
                  "'EC Max Size': '[ 4]', 'EC Count': '1'}. ",
          'running': None,
          'status': 'Completed'}
```

Figure 4-21: SDK Job Status

4.2.2.3.2 Retrieve Anonymized Data

Use this API to retrieve the results of an anonymized job.

Function

`result()`

Parameters

None

Return Type

Returns the AnonResult element, which provides the DataFrame for the anon data.

Note: The `result.df` will be `None` if you have overridden the `resultstore` as part of anonymize method.

Sample Request

`job.result()`

Note: This is a blocking API and will stall processing till the job is complete.

```
In [13]: result = job.result()
if result.df is not None:
    print("Anon Dataframe.")
    print(result.df.head())

Anon Dataframe.
   age  race  income  bmi  gender  occupation
0  Missing  White  <=50K  10.0 - 14.99  *  *
1  Missing  White  <=50K  10.0 - 14.99  *  *
2  Missing  White  <=50K  10.0 - 14.99  *  *
3  Missing  White  >50K  10.0 - 14.99  *  *
```

Figure 4-22: SDK Retrieve Job Status

4.2.2.3.3 Abort

Use this API to abort a running anonymize operation. You can abort jobs if you need to modify the parameters or if the job is stalled or taking too much time or resources to process.

Function

`abort()`

Parameters

None

Return Type

A string with the status of the abort request.

Sample Request

`job.abort()`

```
In [26]: job = asdk.anonymize(e)

In [27]: job.id()
Out[27]: 'c29fc7f8-d48b-40f1-9545-237ef94fd28c'

In [28]: job.abort()

In [29]: job.status()
Out[29]: {'completed': None,
'id': 'c29fc7f8-d48b-40f1-9545-237ef94fd28c',
'info': 'Code:ANON_3001 Message:Job Abort Details: None. ',
'running': None,
'status': 'Aborted'}
```

Figure 4-23: Abort Job

4.2.2.4 Statistics APIs

These APIs are used to obtain information about the anonymization data. Use these APIs to obtain the risk and utility information about the anonymization. The user needs to access these APIs to measure the utility benefits and risk of publishing the anonymized data. If these configurations are not satisfactory, then the user can re-submit the anonymization job after modifying some parameters based on these results.

4.2.2.4.1 Get Exploratory Statistics

Use this API to obtain data distribution statistics about an anonymization operation.

Function`exploratoryStats()`**Parameters**

None

Return Type

A Pandas dataframe with the exploratory information of the source data and the anonymized data.

Sample Request`job.exploratoryStats()`

This provides the data distribution of the attribute, which is all unique values of an attribute and its occurrence count. This can be used to build data histogram of all attributes in the data set. The following values appear for the source and result set:

```
In [16]: job.exploratoryStats()
Out[16]:
```

	attribute	data	frequency
0	age	28	1.0
1	age	31	1.0
2	age	37	1.0
3	age	38	1.0
4	age	39	1.0
5	age	49	1.0
6	age	50	1.0
7	age	52	1.0
8	age	53	1.0
9	race	Black	3.0
10	race	White	6.0
11	bmi	11.5	2.0
12	bmi	12.5	1.0
13	bmi	13.5	1.0
14	bmi	14.5	1.0
15	bmi	16.5	2.0
16	bmi	17.5	1.0
17	bmi	18.5	1.0
18	income	<=50K	7.0
19	income	>50K	2.0

	attribute	data	frequency
0	age	Missing	4.0
1	race	White	4.0
2	bmi	10.0 - 14.99	4.0
3	income	<=50K	3.0
4	income	>50K	1.0

*Figure 4-24: SDK Exploratory Statistics***4.2.2.4.2 Get Risk Metric**

Use this API to ascertain the risk of the source data and the anonymized data. It shows the risk of the data against attacks such as journalist, marketer, and prosecutor.

Function`riskStat()`**Parameters**

None

Return Type

A Pandas dataframe with the source data and the anonymized data privacy risk information.

Note: You can customize the *riskThreshold* as part of *AnonElement* configuration.

Sample Request`job.riskStat()`

The following values appear for the source and result set:

- **avgRecordIdentification:** This value displays the average probability for identifying a record in the anonymized dataset. The risk is higher when the value is closer to the value 1.
- **maxProbabilityIdentification:** This displays the maximum probability value that a record can be identified from the dataset. The risk is higher when the value is closer to the value 1.

- **riskAboveThreshold:** This value displays the number of records that are at a risk above the risk threshold. The default threshold is 10%. The threshold is the maximum value set as a boundary. Any values beyond the threshold are a risk and might be easy to identify. For this result, the value 0 is preferred.

```
In [14]: job.riskStat() # Risk Metrics
```

```
Out[14]:
```

	journalist	marketer	prosecutor	type
avgRecordIdentification	1.00	1.00	1.00	Source
maxProbabilityIdentification	1.00	0.00	1.00	Source
riskAboveThreshold	1.00	0.00	1.00	Source
avgRecordIdentification	0.25	0.25	0.25	Result
maxProbabilityIdentification	0.25	0.00	0.25	Result
riskAboveThreshold	1.00	0.00	1.00	Result

Figure 4-25: SDK Risk Metric

4.2.2.4.3 Get Utility Statistics

Use this API to check the usability of the anonymized data. It shows the information that was lost to gain privacy protection.

Function

`utilityStat()`

Parameters

None

Return Type

A Pandas dataframe with the source and anonymized data utility information.

Sample Request

`job.utilityStat()`

The following values appear:

- **ambiguity:** This value displays how well a record is hidden in all the records. This captures the ambiguity of records.
- **average_class_size:** This measures the average size of groups of indistinguishable records. A smaller class size is more favourable for retaining the quality of the information. A larger class size increases anonymity at the cost of quality.
- **discernibility:** This measures the size of groups of indistinguishable records with penalty for records which have been completely suppressed. Discernibility metrics measures the cardinality of the equivalent class. Discernibility metrics considers only the number of records in the equivalent class and does not capture information loss caused by generalization.
- **generalization_intensity:** Data transformation from the original records to anonymity is performed using generalization and suppression. This measures the concentration of generalization and suppression on attribute values.
- **infoLoss:** This value displays the probability of information lost with the data transformation from the original records. Larger the value, lesser the quality for further analysis.

```
In [15]: job.utilityStat() # Utility Metrics
```

```
Out[15]:
```

	Source	Result
ambiguity	1.0	1.000000
average_class_size	1.0	0.000000
discernibility	1.0	0.902778
generalization_intensity	1.0	0.722222
infoLoss	0.0	0.095640

Figure 4-26: SDK Utility Statistics

4.3 Building the Anonymization request

Use the APIs provided with the Protegrity Anonymization API to create your request. You need to specify the source that must be transformed. The source can be a single row of data or multiple rows of data sent in the request, or it could be a file located on the Cloud storage.

Next, you need to specify the transformation that must be performed on the various columns in the table. Finally, after the transformation is complete, you can save the output or use it for further processing. The transformation request can be saved for processing further requests. It can also be used as an input in machine learning.

4.3.1 Building the request using REST

Use the information provided in this section to build the REST request for performing the Protegrity Anonymization API transformation.

4.3.1.1 Identifying the Source and Target

The source data set is the starting point of the transformation. In this step, you specify the source that must be transformed. Specify the target where the anonymized data will be saved.

- The following file formats are supported:
 - Comma separated values (CSV)
 - Columnar storage format: This is an optimized file format for large amounts of data. Using this file format provides faster results. For example, Parquet (gzip and snappy).
- The following data storages have been tested for the Protegrity Anonymization API:
 - Local File System
 - Amazon S3
- The following data storages can also be used for the Protegrity Anonymization API:
 - Google Cloud Storage
 - Microsoft Azure Storage
 - Data Lake Storage
 - Blob Storage
 - Hadoop Distributed File System (HDFS)
 - Other S3 Compatible Services

Use the following code to specify the source:

Note: Modify the source and destination code for your provider.

For more cloud-related sample codes, refer to the section [Samples for Cloud-related Source and Destination Files](#).

```
"source": {
  "type": "File",
  "file": {
    "name": "<Source_file_path>"
  }
}
```

```
}
}
```

Note: When uploading a file to the Cloud service, wait till the entire source file is uploaded before running the anonymization job.

Similarly, specify the target file using the following code:

```
"target": {
  "type": "File",
  "file": {
    "name": "<Target_file_path>"
  }
}
```

Specify additional parameters about the source and target file, such as, the character used to separate the values in the file, using the following properties attribute. If a property is not specified, then the default attribute shown here will be used.

```
"props": {
  "sep": ",",
  "decimal": ".",
  "quotechar": "\"",
  "escapechar": "\\",
  "encoding": "utf-8",
  "line_terminator": "\n"
}
```

If the required files are on a cloud storage, then specify the cloud-related access information using the following code:

```
"accessOptions": {
}
```

For more information and help on specifying the source and target files, refer to the Dask remote data configuration at <https://docs.dask.org/en/latest/remote-data-services.html>.

Note: If the target directory already exists, then the job fails. If the target file already exists, then the file will be overwritten. Additionally, some Cloud services have limitations on the file size. If such a limitation exists, then you can set the *single_file* switch to *no* when writing large files to the Cloud storage. This saves the output as multiple files to avoid any errors related to saving large files to the Cloud storage.

4.3.1.2 Specifying the Transformation

The data store consists of various fields. These fields need to be identified for processing data. Additionally, the type of transformation that must be performed on the fields must be specified. Also specify the type of privacy model that must be used for anonymizing the data. While specifying the rules for transformation specify the importance of the data.

4.3.1.2.1 Classifying the Fields

Specify the type of information that the fields hold. This classification must be performed carefully, leaving out important fields might lead to the anonymized data being of no value. However, including data that can identify users poses a risk of anonymization not being carried out properly.

The following four different classifications are available:

Classification	Description	Treatment
<i>Direct Identifier</i>	This classification is used for the data in fields that directly identify an individual, such as Name, SSN, phoneNo, email, and so on.	Values will be removed.
<i>Quasi Identifying Attribute</i>	This classification is used for the data in fields that does not identify an individual directly. However, it needs to be modified to avoid indirect identification. For example, age, date of birth, zip code, and so on.	Values will be transformed using the options specified.
<i>Sensitive Attribute</i>	This classification is used for the data in fields that does not identify an individual directly. However, it needs to be modified to avoid indirect identification. This data needs to be preserved to ensure further analysis or to obtain utility out of Anonymized data. In addition, ensure that records with this classification are part of a herd or group where it loses the ability to identify an individual.	No change in values, exception extreme values that might identify an individual. Values will be generalized in case of t-closeness.
<i>Non-Sensitive Attribute</i>	This classification is used for the data in fields that does not identify an individual directly or indirectly.	No change in values.

Ensure that you identify the sensitive and the quasi-identifier fields for specifying the anonymization method for hiding individuals in the data set. Use the `classificationType` attribute to specify the classification. For example, use the following code for specifying a quasi-identifier:

```
"classificationType": "Quasi Identifier",
```

The following data types are supported for working with the data in the fields:

- Integer
- Float
- String
- Date: The following date types are supported:
 - mm-dd-yyyy

Note: This is the default format.

- dd-mm-yyyy
- dd-mm-yy
- mm-dd-yy
- dd.mm.yyyy
- mm.dd.yyyy
- dd.mm.yy
- mm.dd.yy
- dd/mm/yyyy
- mm/dd/yyyy
- dd/mm/yy
- mm/dd/yy
- Time: *HH* is used to specify time in the 24-hour format and *hh* is used to specify time in the 12-hour format. The following time formats are supported:

- HH:mm:ss

Note: This is the default format.

- HH:mm:ss.ns
- hh:mm:ss
- hh:mm:ss.ns
- hh:mm:ss.ns p

Note: In this, *p* is the 12 hour format with period AM/PM.

- HH:mm:ss.ns z

Note: In this, *z* is timezone info with +- from UTC, that is, +0000,+0530,-0230.

- hh:mm:ss Z

Note: In this, *Z* is the timezone info with the name, that is, UTC,EST, CST.

Here are a few examples:

```
{
  "classificationType": "Non-Sensitive Attribute",
  "dataType": "Integer",
  "name": "index"
}
```

```
{
  "classificationType": "Sensitive Attribute",
  "dataType": "String",
  "name": "diagnosis_dup"
}
```

Note: The values present in the first row of the data set is considered for determining the format for *date*, *time*, and *datetime*. You can override the detection using *"props": {"dateformat": "<Specify_Format>"}*.

Consider the following example for date with the *mm/dd/yyyy* format:

```
10/09/2020
12/24/2020
07/30/2020
```

In this case, the data will be identified as *dd/mm/yyyy*.

You can override the using the following property:

```
"props": {"dateformat": "mm/dd/yyyy" }
```

4.3.1.2.2 Specifying the Privacy Model

The privacy model transforms the data set using one or several anonymization methods for privacy guarantee.

The following anonymization techniques are available in the Protegrity Anonymization API:

Model	Information Type	Description	Example
K-anonymity	Quasi-Identifier	Configuration of quasi-identifier tuple occurs of k records.	<pre>"privacyModel": { "k": { "kValue": 5 } }</pre>
l-diversity	Sensitive Attribute	Ensures k records in the inter-group is distributed and diverse enough to reduce the risk of identification.	<pre>"privacyModel": { "ldiversity": [{ "lFactor": 2, "name": "sex", "lType": "Distinct-l-diversity" }] }</pre>
t-closeness	Sensitive Attribute	Intra-group diversity for every sensitive attribute must be defined.	<pre>"privacyModel": { "tcloseness": [{ "name": "salary-class", "emdType": "EMD with equal ground distance", "tFactor": 0.2 }] }</pre>

4.3.1.2.3 Specifying the Hierarchy

The hierarchy specifies how the information in the data set is handled for anonymization. These hierarchical transformations are performed on Quasi-Identifiers and Sensitive Attributes. Accordingly, the data can be generalized using transformations or aggregated using mathematical functions. As we go up the hierarchy, the data is anonymized better, however, the quality of data for further analysis reduces.

Global Recoding and Full Domain Generalization

Global recoding and full domain generalization is used for anonymizing the data. When data is anonymized, the quasi-identifiers values are transformed to ensure that data fulfils the required privacy requirements. This transformation is also called as data recoding. In the Protegrity Anonymization API, data is anonymized using global recoding, that is, the same transformation rule is applied to all entries in the data set.

Consider the data in the following tables:

Table 4-1: Sample Data for Anonymization

ID	Gender	Age	Race
1	Male	45	White
2	Female	30	White
3	Male	25	Black
4	Male	30	White
5	Female	45	Black

Table 4-2: Sample Age Generalization Hierarchy

Level0	Level1	Level2	Level3	Level4
25	20-25	20-30	20-40	*

Level0	Level1	Level2	Level3	Level4
30	30-35	30-40	30-50	*
45	40-45	40-50	40-60	*

In the above example, when global recoding is used for a value such as 45, then all occurrences of age 45 will be generalized using only one generalized level as follows:

- 40-45
- 40-50
- 40-60
- *

Full-domain generalization means that all values of an attribute are generalized to the same level of the associated hierarchy level. Thus, in the first table, if age 45 gets generalized to 40-50 which is Level2, then all age values are also generalized to Level2 only. Hence, the value 30 will be generalized to 30-40.

4.3.1.2.3.1 Generalization

In Generalization, the data is grouped into sets having similar attributes.

The following transformations are available:

- **Masking Based:** In this transformation, information is hidden by masking parts of the data to form similar sets. For example, masking the last three numbers in the zip code could help group them, such as, 54892 and 54231 both being transformed as 54###.

An example of masking-based transformation is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "String",
  "generalization": {
    "hierarchyType": "Rule",
    "rule": {
      "masking": {
        "maskOrder": "Right To Left",
        "maskChar": "#",
        "maxDomainSize": 5
      }
    },
    "type": "Masking Based"
  },
  "name": "city"
}
```

Where:

- *maskOrder* is the order for masking, use *Right To Left* to mask from right and *Left To Right* for masking from the left.
- *maskChar* is the placeholder character for masking.
- *maxDomainSize* is the number of characters to mask. Default is the maximum length of the string in the column.
- **Tree Based:** In this transformation, data is aggregated by transformation to form similar sets using external knowledge. For example, in the case of address, the data can be anonymized based on the city, state, country, or continent, as required. You must specify the file containing the tree data. If the current level of aggregation does not provide adequate anonymization, then a higher level of aggregation is used. The higher the level of aggregation, the more the data is generalized. However, a higher level of generalization reduces the quality of data for further analysis.

An example of tree-based transformation is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "String",
  "generalization": {
    "type": "Tree Based",
    "hierarchyType": "Data Store",
    "dataStore": {
      "type": "File",
      "file": {
        "name": "adult_hierarchy_education.csv",
        "props": {
          "delimiter": ";",
          "quotechar": "\"",
          "header": null
        }
      },
      "format": "CSV"
    }
  },
  "name": "education"
}
```

- **Interval Based:** In this transformation, data is aggregated into groups according to a predefined interval specified.

An example of interval-based transformation is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "Integer",
  "generalization": {
    "hierarchyType": "Rule",
    "rule": {
      "interval": {
        "levels": [
          "5",
          "10",
          "50",
          "100"
        ],
        "lowerBound": "0"
      }
    },
    "type": "Interval Based"
  },
  "name": "age"
}
```

- **Aggregation Based:** In this transformation, integer data is aggregated as per the conditions specified. The available options for aggregation are *Mean*, *GMean*, and *Mode*.

Note: Geometric mean (*GMean*) is only supported for positive numbers.

An example of aggregation-based transformation is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "Integer",
  "generalization": {
    "hierarchyType": "Aggregate",
    "type": "Aggregation Based",
    "aggregateFn": "Mean"
  },
  "name": "age"
}
```

```
    "name": "age"
  }
```

- **Date Based:** In this transformation, data is aggregated into groups according to the date.

An example of date-based interval and rounding is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "Date",
  "generalization": {
    "hierarchyType": "Rule",
    "type": "Interval Based",
    "rule": {
      "daterange": {
        "levels": [
          "WD.M.Y",
          "W.M.Y",
          "FD.M.Y",
          "M.Y",
          "QTR.Y",
          "Y",
          "DEC",
          "CEN"
        ]
      }
    }
  },
  "name": "date_of_birth"
}
```

- **Time Based:** In this transformation, data is aggregated into groups according to the time. In this, Time intervals are in seconds. The LowerBound and UpperBound takes value of the format *[HH:MM:SS]*.

An example of time-based interval and rounding is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "Date",
  "generalization": {
    "hierarchyType": "Rule",
    "type": "Interval Based",
    "rule": {
      "interval": {
        "levels": [
          "30",
          "60",
          "180",
          "240"
        ],
        "lowerBound": "00:00:00",
        "upperBound": "23:59:59"
      }
    }
  },
  "name": "time_of_birth"
}
```

4.3.1.2.3.2 Micro-Aggregation

In Micro-Aggregation, mathematical formulas are used to group the data. This is used to achieve K-anonymity by forming small groups of data in the data set.

The following aggregation functions are available for micro-aggregation in the Protegrity Anonymization API:

- For numeric data types (integer and decimal):
 - Arithmetic Mean
 - Geometric Mean

Note: Micro-Aggregation using geometric mean is only supported for positive numbers.

- Median
- For all data types:
 - Mode

An example of micro-aggregation is provided here.

```
{
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Micro Aggregation",
  "dataType": "Decimal",
  "aggregateFn": "Median",
  "name": "age_ma_median"
}
```

4.3.1.2.4 Specifying Configurations

Additional configurations are available in the Protegrity Anonymization API to enhance the anonymity of the information in the data set.

The following configurations are available:

```
"config": {
  "maxSuppression": 0.1
  "suppressionData": "*"
  "redactOutliers": False
}
```

- *maxSuppression* specifies the percentage of rows allowed to be an outlier row to obtain the anonymized data. The default is 10%.
- *suppressionData* specifies the character or character set to be used for suppressing the anonymized data. The default is *.
- *redactOutliers* specifies if the outlier row should be part of the anonymized data set or not. The default is included denoted by False.

4.3.2 Building the request using SDK

Use the information provided in this section to build the request using the SDK environment for performing the Protegrity Anonymization API transformation.

To build an anonymization request using the SDK, the user first needs to import the *anonsdk* module using the following command.

```
import anonsdk as asdk
```

4.3.2.1 Creating the Connection

You need to specify the connection to the Protegrity Anonymization API REST service to set up the Protegrity Anonymization API SDK.

Note: If administrator has not updated the DNS entry for ANON REST API service, then map the hostname with the IP address of Anon Service in the *hosts* file of the system.

For example, if the Protegrity Anonymization API REST service is located at *https://anon.protegrity.com*, then you would create the following connection.

```
conn = asdk.Connection("https://anon.protegrity.com/")
```

4.3.2.2 Identifying the Source and Target

The Protegrity Anonymization API SDK is built to anonymize the data in a Pandas dataframe and return the anonymized dataframe. However, you can also specify a CSV file from various source systems for the source data.

Use the following code to specify the source.

```
e = asdk.AnonElement(conn, dataframe)
```

If the source file is located at the same place where the Protegrity Anonymization API SDK is installed, then use the following code to load the source file into a dataframe.

```
dataframe = pandas.read_csv("<file_path>")
```

- The following data storages have been tested for the Protegrity Anonymization API:
 - Local File System
 - Amazon S3

For example:

```
asdk.FileDataStore("s3://<path>/<file_name>.csv", access_options={"key":  
"<value>", "secret": "value"})
```

- The following data storages can also be used for the Protegrity Anonymization API:
 - Google Cloud Storage

For example:

```
asdk.FileDataStore("gs://<path>/<file_name>.csv", access_options={'token': '<path>/  
adc.json', 'project': '<value>'})
```

Note:

Ensure that you log in and update the *adc.json* file. For more information, refer to the section [Samples for Cloud-related Source and Destination Files](#).

- Microsoft Azure Storage
 - Data Lake Storage

For example:

```
asdk.FileDataStore("adl://<path>/<file_name>.csv", access_options={"tenant_id":
"<value>", "client_id": "<value>", "client_secret": "<value>"})
```

- Blob Storage

For example:

```
asdk.FileDataStore("abfs://<path>/<file_name>.csv", access_options={"account_name":
"<value>", "account_key": "<value>"})
```

- Hadoop Distributed File System (HDFS)
- Other S3 Compatible Services

Note: When uploading a file to the Cloud service, wait till the entire source file is uploaded before running the anonymization job.

For more information about using remote sources, navigate to <https://docs.dask.org/en/latest/how-to/connect-to-remote-data.html>.

If required, you can directly specify data in a list using the following format:

```
d = {'<column1_name>': ['value1', 'value2', 'value3', ...],
     '<column2_name>': [number1, number2, number3, ...],
     '<column3_name>': ['value1', 'value2', 'value3', ...],
     ...}
```

For example:

```
d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male',
'Female'],
     'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-
cleaners', 'Prof-specialty', 'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-
specialty'],
     'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
     'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White',
'White'],
     'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-
spouse', 'Married-civ-spouse', 'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-
spouse', 'Never-married'],
     'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters',
'9th', 'HS-grad', 'Masters'],
     'native-country': ['United-States', 'United-States', 'United-States', 'United-
States', 'Cuba', 'United-States', 'Jamaica', 'United-States', 'United-States'],
     'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private',
'Private', 'Private', 'Self-emp-not-inc', 'Private'],
     'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K',
'>50K'],
     'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }
```

The anonymized data is returned to the user as Pandas dataframe. Optionally, you can specify the required target file system and provide the target using the following code.

```
asdk.anonymize(e, resultStore=<targetFile>)
```

Specify additional parameters about the source and target file, such as, the character used to separate the values in the file, using the various properties attribute. If a property is not specified, then the default attributes will be used.

Note: Some Cloud services have limitations on the file size. If such a limitation exists, then you can set *single_file* to *no* when writing large files to the Cloud service, . This saves the output as multiple files to avoid any errors related to saving large files to the Cloud storage.

For more information and help on specifying the source and target files, refer to the Dask remote data configuration at <https://docs.dask.org/en/latest/remote-data-services.html>.

4.3.2.3 Specifying the Transformation

The data store consists of various fields. These fields need to be identified for processing data. Additionally, the type of transformation that must be performed on the fields must be specified. Also specify the type of privacy model that must be used for anonymizing the data. While specifying the rules for transformation specify the importance of the data.

The Protegrity Anonymization API uses Pandas to build and work with the data frame. You need to import the library for Pandas and store the source data that must be transformed in Pandas.

```
import pandas as pd

d = <source_data>
df = pd.DataFrame(data=d)
```

To build the transformation, you need to specify the AnonElement that holds the connection, data frame, and the source.

For example:

```
e = asdk.AnonElement(conn,df,source=datastore)
```

You need to specify the columns that must be included for processing the anonymization request and the column classification before performing the anonymization.

```
e["<column>"] = asdk.<transformation>
```

Where:

- *column*: Specify the column name or column ID.
- *transformation*: Specifies the processing to be applied for the column.

Note: By default, all the columns are set to ignore processing. The data is redacted and not included in the anonymization process. You need to manually set the column classification to include it in the anonymization process.

Specify multiple columns with *assign* using commas.

```
e.assign(["<column1>","<column2>"],asdk.Transformation()))
```

You can view the configuration provided using the *describe* function.

```
e.describe()
```

4.3.2.3.1 Classifying the Fields

Specify the type of information that the fields hold. This classification must be performed carefully, leaving out important fields might lead to the anonymized data being of no value. However, including data that can identify users poses a risk of anonymization not being carried out properly.

The following four different classifications are available:

Classification	Description	Function	Treatment
<i>Direct Identifier</i>	This classification is used for the data in fields that directly identify an individual, such as, Name, SSN, phoneNo, email, and so on.	<i>Redact</i>	Values will be removed.
<i>Quasi Identifying Attribute</i>	This classification is used for the data in fields that does not identify an individual directly. However, it needs to be modified to avoid indirect identification. For example, age, date of birth, zip code, and so on.	Hierarchy models	Values will be transformed using the options specified.
<i>Sensitive Attribute</i>	This classification is used for the data in fields that does not identify an individual directly. However, it needs to be modified to avoid indirect identification. This data needs to be preserved to ensure further analysis or to obtain utility out of Anonymized data. In addition, ensure that records with this classification are part of a herd or group where it loses the ability to identify an individual.	<i>LDiv, TClose</i>	No change in values, exception extreme values that might identify an individual.
<i>Non-Sensitive Attribute</i>	This classification is used for the data in fields that does not identify an individual directly or indirectly.	<i>Preserve</i>	No change in values.

Ensure that you identify the sensitive and the quasi-identifier fields for specifying the anonymization method for hiding individuals in the data set. For example, use the following code for specifying a quasi-identifier with Masking as the Transformation:

```
e[ '<column>' ] = asdk.Gen_Mask(maskchar='#', maxLength=3, maskOrder="L")
```

The following data types are supported for working with the data in the fields:

- Integer
- Float
- String
- DateTime

4.3.2.3.2 Specifying the Privacy Model

The privacy model transforms the data set using one or several anonymization methods for privacy guarantee.

The following anonymization techniques are available in the Protegrity Anonymization API:

Model	Information Type	Description	Example
K-anonymity	Quasi-Identifier	Configuration of quasi-identifier tuple occurs of k records.	<code>e.config.k=asdk.K(2)</code>
l-diversity	Sensitive Attribute	Ensures k records in the inter-group is distributed and diverse enough to reduce the risk of identification.	<code>e["<column>"]=asdk.LDiv(lfactor=2)</code>
t-closeness	Sensitive Attribute	Intra-group diversity for every sensitive attribute must be defined.	<code>e["<cloumn>"]=asdk.TClose(tfactor=0.2)</code>

4.3.2.3.3 Specifying the Hierarchy

The hierarchy specifies how the information in the data set is handled for anonymization. These hierarchical transformations are performed on Quasi-Identifiers and Sensitive Attributes. Accordingly, the data can be generalized using transformations or aggregated using mathematical functions. As we go up the hierarchy, the data is anonymized better, however, the quality of data for further analysis reduces.

Global Recoding and Full Domain Generalization

Global recoding and full domain generalization is used for anonymizing the data. When data is anonymized, the quasi-identifiers values are transformed to ensure that data fulfils the required privacy requirements. This transformation is also called as data recoding. In the Protegrity Anonymization API, data is anonymized using global recoding, that is, the same transformation rule is applied to all entries in the data set.

Consider the data in the following tables:

Table 4-3: Sample Data for Anonymization

ID	Gender	Age	Race
1	Male	45	White
2	Female	30	White
3	Male	25	Black
4	Male	30	White
5	Female	45	Black

Table 4-4: Sample Age Generalization Hierarchy

Level0	Level1	Level2	Level3	Level4
25	20-25	20-30	20-40	*
30	30-35	30-40	30-50	*
45	40-45	40-50	40-60	*

In the above example, when global recoding is used for a value such as 45, then all occurrences of age 45 will be generalized using only one generalized level as follows:

- 40-45
- 40-50
- 40-60
- *

Full-domain generalization means that all values of an attribute are generalized to the same level of the associated hierarchy level. Thus, in the first table, if age 45 gets generalized to 40-50 which is Level2, then all age values are also generalized to Level2 only. Hence, the value 30 will be generalized to 30-40.

4.3.2.3.1 Generalization

In Generalization, the data is grouped into sets having similar attributes.

The following transformations are available:

- **Masking Based:** In this transformation, information is hidden by masking parts of the data to form similar sets. For example, masking the last three numbers in the zip code could help group them, such as, 54892 and 54231 both being transformed as 54###.

An example of masking-based transformation is provided here.

```
e["zip_code"] = asdk.Gen_Mask(maskchar="#", maskOrder = "R", maxLength=5)
```

Where:

- *maskchar* is the placeholder character for masking.
 - *maskOrder* is the order for masking, use *R* to mask from right and *L* for masking from the left.
 - *maxLength* is the number of characters to mask. Default is the maximum length of the string in the column.
- **Tree Based:** In this transformation, data is aggregated by transformation to form similar sets using external knowledge. For example, in the case of address, the data can be anonymized based on the city, state, country, or continent, as required. You must specify the file containing the tree data. If the current level of aggregation does not provide adequate anonymization, then a higher level of aggregation is used. The higher the level of aggregation, the more the data is generalized. However, a higher level of generalization reduces the quality of data for further analysis.

An example of tree-based transformation is provided here.

```
treeGen = {'lv10': [11, 13, 14, 15, 27, 28, 20],
           'lv11': ["<15", "<15", "<15", "<20", "<30", "<30", "<25"],
           'lv12': ["<20", "<20", "<20", "<30", "<30", "<30", "<30"]}

e["bmi"] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", "Might be <30"])
```

You can refer to an external file for specifying the parameters for the hierarchy tree.

```
education_df = pd.read_csv('D:\\WS\\data source\\hierarchy\\
\\adult_hierarchy_education.csv', sep=';')
e['education'] = asdk.Gen_Tree(education_df)
```

- **Interval Based:** In this transformation, data is aggregated into groups according to a predefined interval specified. In addition to the list, the lowerbound and upperbound values need to be specified. Values below the lowerbound and values above the upperbound are excluded from range generation.

```
asdk.Gen_Interval([<interval_level>], <lowerbound>, <upperbound>)
```

An example of interval-based transformation is provided here.

```
e['age'] = asdk.Gen_Interval([5,10,15])
e['age'] = asdk.Gen_Interval([5,10,15],20,60)
```

- **Aggregation Based:** In this transformation, data is aggregated as per the conditions specified. The available options for aggregation are *Mean*, *GMean*, and *Mode*.

Note: Geometric mean (*GMean*) is only supported for positive numbers.

An example of aggregation-based transformation is provided here.

```
e['salary'] = asdk.Gen_Agg(AggregateFunction.Mode)
```

An example of aggregation-based transformation is provided here.

```
e['age'] = asdk.Gen_Agg(AggregateFunction.GMean)
```

- **Rounding Based:** In this transformation, data is rounded to groups according to a predefined rounding factor specified.

An example of date-based transformation is provided here.

```
e['DateOfBirth'] = asdk.Gen_Rounding(["H.M4", "WD.M.Y", "M.Y"])
```

An example of numeric-based transformation is provided here.

```
e['Interest_Rate'] = asdk.Gen_Rounding([0.05,0.10,1])
```

4.3.2.3.2 Micro-Aggregation

In Micro-Aggregation, mathematical formulas are used to group the data. This is used to achieve K-anonymity by forming small groups of data in the data set.

The following aggregation functions are available for micro-aggregation in the Protegrity Anonymization API:

- For numeric data types (integer and decimal):
 - Arithmetic Mean
 - Geometric Mean

Note: Micro-Aggregation using geometric mean is only supported for positive numbers.

- Median
- For all data types:
 - Mode

An example of micro-aggregation is provided here.

```
e['income'] = asdk.MicroAgg(asdk.Mean)
```

4.3.2.4 Working with Saved Anonymization Requests

The *save* method provides interoperability with the REST API. It generates the required JSON payload that can be used as part of curl or any REST client.

Use the following command to save the anonymization request.

```
e.save("<file_path>\\fileName.json")
```

4.3.2.4.1 Applying Anonymization to Additional Rows

You can use the *applyAnon* method to anonymize any additional rows using the saved request. Use the following command to anonymize using a previous anonymization job.

```
asdk.applyAnon(<conn>, job.id(), <single_row_data>)
```

Note: Use this function to anonymize only a few rows. You need to specify the row information using the key-value pair and ensure that all the required columns are present.

An example of a single and multi row data is shown here.

```
single_row_data = [{'ID': '1', 'Name': 'Wilburt Daniel', 'Address': '4 Sachtjen Plaza',
'City_Name': 'Ambato', 'Gender': 'Male', 'date': '18-04-2008': '9'}]
multi_row_data = [{'ID': '1', 'Name': 'Wilburt Daniel', 'Address': '4 Sachtjen Plaza',
'City_Name': 'Ambato', 'Gender': 'Male', 'date': '18-04-2008': '9'}, {'ID': '2', 'Name':
'Jones Knight', 'Address': '25 Macadamia Street', 'City_Name': 'Ambato', 'Gender':
'Male', 'date': '25-11-1997': '9'}]
```

4.3.2.5 Running a Sample Request

Run the sample code provided here in and SDK. This sample is also available at https://<IP_Address>:<Port>/sdkapi.

Import the Protegrity Anonymization SDK and the Pandas package in the SDK tool.

```
import pandas as pd
import anonsdk as asdk
```

Create a variable *d* with the sample data.

```
#Sample data for Demo
d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male',
'Female'],
      'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-
cleaners', 'Prof-specialty', 'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-
specialty'],
      'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
      'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White',
'White'],
      'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-
spouse', 'Married-civ-spouse', 'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-
spouse', 'Never-married'],
      'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters',
'9th', 'HS-grad', 'Masters'],
      'native-country': ['United-States', 'United-States', 'United-States', 'United-
States', 'Cuba', 'United-States', 'Jamaica', 'United-States', 'United-States'],
      'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private',
'Private', 'Private', 'Self-emp-not-inc', 'Private'],
      'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K',
'>50K'],
      'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }
```

Load the data in a Pandas DataFrame.

```
df = pd.DataFrame(data=d)
```

Specify the additional data required per attribute to transform and obtain anonymized data. In this example, the Hierarchy Tree is specified.

```
treeGen = {'lv10': [11, 13, 14, 15, 27, 28, 20],
            'lv11': ["<15", "<15", "<15", "<20", "<30", "<30", "<25"],
            'lv12': ["<20", "<20", "<20", "<30", "<30", "<30", "<30"]}
```

Build the connection to a running Protegrity Anonymization API REST cluster instance. Ensure that the hosts file is configured and points to the REST cluster.

```
conn = asdk.Connection('https://anon.protegrity.com/')
```

Build the AnonElement passing the connection and the data as inputs for the anonymization request.

```
e = asdk.AnonElement(conn, df)
```

Note: Use the following code sample to read data from an external file store.

```
e = asdk.AnonElement(conn, dataframe, <SourceFile>)
```

Specify the transformation that is required.

```
e['gender'] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", " Might be <30"])
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
```

Specify the K-value, the L-Diversity, and the T-Closeness values.

```
e.config.k = asdk.K(2)
e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TClose(tfactor=0.2)
```

Specify the max suppression.

```
e.config['maxSuppression'] = 0.7
```

Specify the importance for the required fields.

```
e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)
```

View the details of the current configuration.

```
e.describe()
```

Anonymize the data.

```
job = asdk.anonymize(e)
```

Note: If required, save the results to a file.

```
datastore=asdk.FileDataStore("s3://...",access_options={"key": "K...", "secret": "S..."})
job = asdk.anonymize(e, resultStore=datastore)
```

View the job status.

```
job.status()
```

View the anonymized data.

```
result = job.result()
if result.df is not None:
    print("Anon Dataframe.")
    print(result.df.head())
```

View the utility and risk statistics of the data.

```
job.utilityStat()
job.riskStat()
```

Save the job configuration with the updated source and target to a JSON file.

```
e.save("/file_path/file.json", store=datastore)
```

Optional: Apply the anonymization rules of previous jobs to new data.

```
anonData = asdk.applyAnon(conn,job.id(), [{ 'gender': 'Male', 'age': '39', 'race': 'White',
'income': '<=50K', 'bmi': '12.5' }])
anonData
```

4.4 Using the Workstation Feature

The Anonymization SDK comes preinstalled inside the workstation. You can use the workstation setup to get started and perform anonymization requests. You can use the workstation in Kubernetes or Docker.

4.4.1 Using the Kubernetes Workstation

Create a Route53 entry with the host name configured in the *values.yaml* file and the IP address of the Nginx-Ingress service deployed in the Kubernetes cluster.

The *hostname* is configured in the *values.yaml* file.

Run the following command to obtain the IP address of the Ingress service on the base machine that is used to setup the Kubernetes cluster:

```
kubectl get ingress -n <namespace>
```

Run the following command to resolve the domain name:

```
nslookup <domain_name>
```

For more information about creating the Route53 entry, navigate to <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-creating.html>.

4.4.2 Using the Docker Workstation

Complete one of the following ways to submit the request to the Anonymization API from the workstation.

- Updating the *docker-compose.yml* file.

Edit the workstation service provided in the *docker-compose.yml* file with the following details.

```
extra_hosts:
  "anon.protegrity.com:<IP of the host machine>"
```

- Creating the Route53 entry.

Create a Route53 entry for with the *hostname* of the IP of the docker machine on which the Protegrity Anonymization API is running.

For more information about creating the Route53 entry, navigate to <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-creating.html>.

4.4.3 Using the Jupyter Notebook

A sample Jupyter notebook is provided with the workstation. You can run an anonymization request using the sample provided in the notebook. Complete the steps provided in this section to run the notebook.

1. Open a browser on the system where the Docker Workstation is installed and navigate to <https://anon.protegrity.com/lab>.
2. From the left pane, expand **sdk-samples** and double-click **AnonSDK-Sample** to open the notebook.

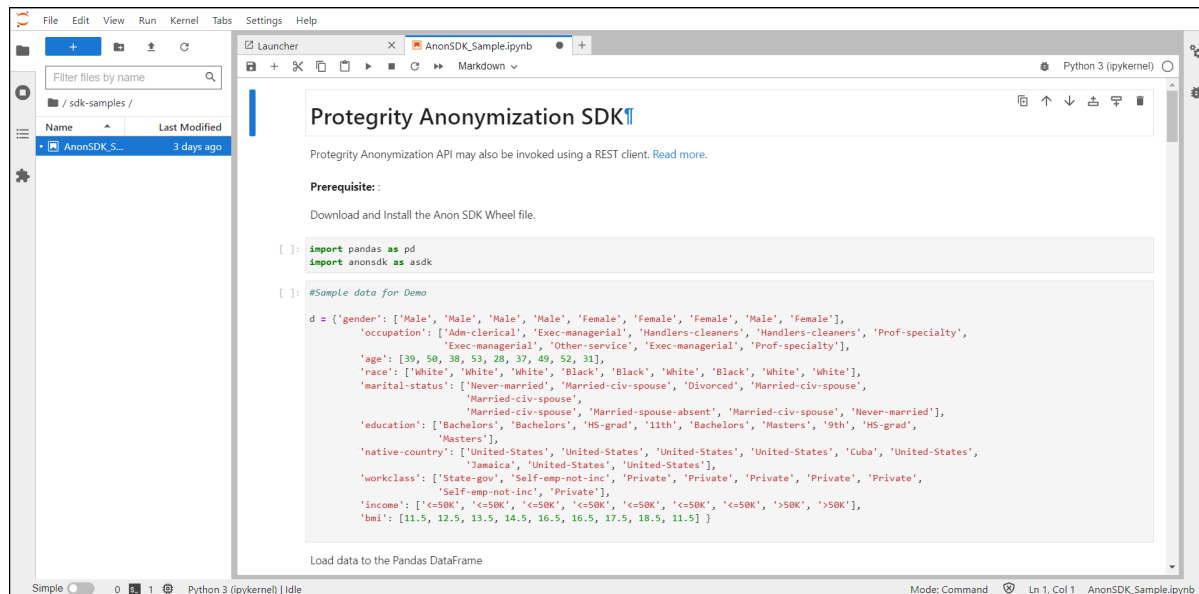



Figure 4-27: Sample Notebook

3. Click  (**Run**) to run the anonymization request.

Note: You can customize the values in the notebook before running the request.

Chapter 5

Using the Auto Anonymizer

5.1 Using mode to Auto Anonymize

5.2 Using Infer to Anonymize

The Auto Anonymizer feature of the Protegrity Anonymization API is a powerful feature for performing anonymization. It processes the data to generate a template for completing anonymization requests. It is simple and easy to configure. Moreover, it is built to analyze the data and produce an output that has a balance of both, generalization and value.

The Protegrity Anonymization API analyzes a sample of the data from the dataset. This sample is then analyzed to build a template for performing the anonymization. The template building takes time, based on the size of the dataset.

You can specify the parameters such as, the various fields for redacting, for anonymizing the data. You can use the Auto Anonymizer feature to automatically analyze the data and perform the required anonymization. This feature can also scan the data and perform the best optimization for providing high quality anonymized data. The various parameters used for performing auto anonymization are configurable and can be optimized to suite your business need or requirements. Additionally, frequently performed fields can be created and stored to enable you to build the anonymization request faster and with minimal information before runtime.

A brief flow of the steps for auto anonymization is shown in the following figure.

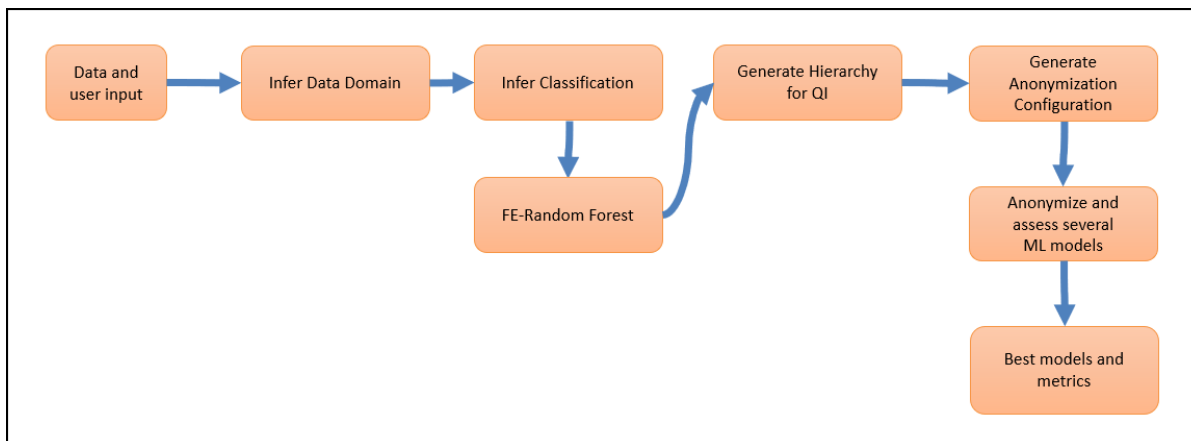


Figure 5-1: Auto Anonymization Flow

The user provides the data, column identification, and anonymization parameters, if required. The Protegrity Anonymization API analyzes the parameters provided and analyzes the data set. Various anonymization models are generated and analyzed. The parameters, such as, the K, l, and t values, along with the data available in the data set are used for processing the request. The results are compared and finally, the data set is processed using the model and parameters that have the best anonymization output.

Consider the following sample graph.

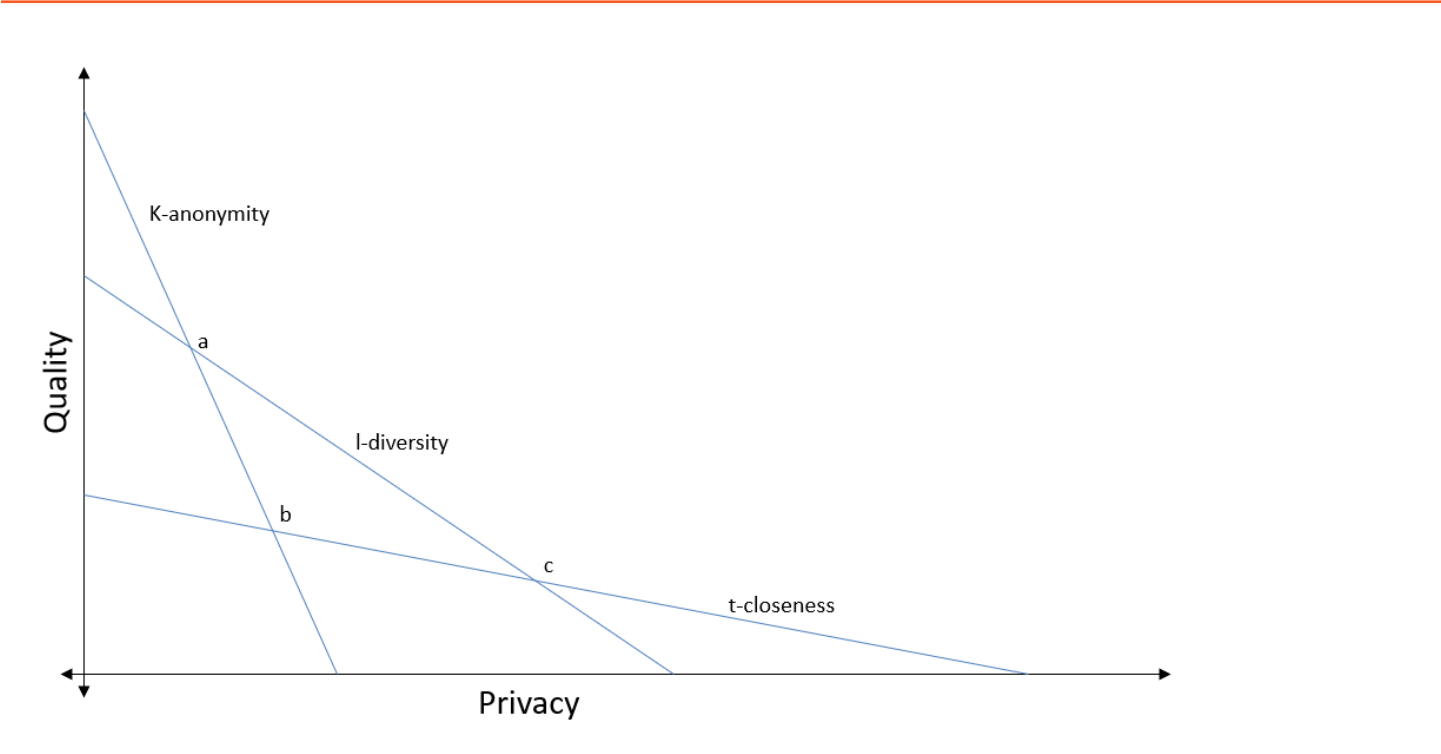


Figure 5-2: Sample Graph

The Protegrity Anonymization API will first auto assign the privacy levels for the various columns in the data set. The sensitive data will be redacted from the data set. Next, models will be created using different values for K-anonymity, l-diversity, and t-closeness. The values will be analyzed, and the best values selected, such as, the values at *point b* in the graph. The data set will then be anonymized using the values determined to complete the anonymization request.

The user can specify the values that must be used, if required. The Protegrity Anonymization API will consider the values specified by the user and continue to auto generate the remaining values accordingly.

Note: The auto anonymization runs the same request using different values, the anonymization request will take more time to complete compared to a regular anonymization request.

You can use *measure*, *mode*, and *Infer* for Auto Anonymization.

For more information about the *measure* API, refer to the section [Measure](#).

The difference between using *mode* and *Infer* is provided in the following table.

Mode	Infer
Analyzes the data set and performs the anonymization job.	Only analyzes the data set.
The result set is the output.	Updates the models used for performing the anonymization job.
You cannot retrieve the attributes for the job.	You can view the auto generated job attribute values, such as, K-anonymity, that will be used for performing the job using the <i>describe</i> method.
You can specify target variables for focusing the anonymization job with the anonymization function.	You can specify target variables for focus before performing the anonymization job or even modify the model after performing the anonymization job.

5.1 Using *mode* to Auto Anonymize

Set the *mode* to *Auto* to auto anonymize. The auto anonymization auto-detects the data-domain, classification type, hierarchies, and anonymization configuration in the Protegrity Anonymization SDK. Any user-defined configuration, such as, QI attribute assignments, hierarchy, and K value, are retained and considered while performing the auto anonymization. You can also specify the *targetVariable* that must be considered for obtaining the best possible result set in terms of quality data while performing the anonymization job.

Ensure that you complete the following checks before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.
- Verify that you have imported the Pythonic SDK, for example, *import anonsdk as asdk*.

Function

```
job = asdk.anonymize(e, targetVariable="targetVariable", mode="Auto")
```

Parameters

targetVariable: The field specified here is used as a focus point for performing the anonymization.

Return Type

It returns the result set after performing the anonymization job.

Sample Request

```
job = asdk.anonymize(e, targetVariable="date", mode="Auto")
```

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization SDK](#).

```

In [ ]: import pandas as pd
import anonsdk as asdk

In [ ]: d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Female'],
            'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-cleaners', 'Prof-specialty',
                           'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-specialty'],
            'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
            'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White', 'White'],
            'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-spouse',
                              'Married-civ-spouse',
                              'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-spouse', 'Never-married'],
            'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters', '9th', 'HS-grad',
                          'Masters'],
            'native-country': ['United-States', 'United-States', 'United-States', 'United-States', 'United-States', 'Cuba', 'United-States',
                              'Jamaica', 'United-States', 'United-States'],
            'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private', 'Private', 'Private', 'Private',
                          'Self-emp-not-inc', 'Private'],
            'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K', '>50K'],
            'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }

In [ ]: df = pd.DataFrame(data=d)

In [ ]: treeGen = {'lvl0': [11, 13, 14, 15, 27, 28, 20],
                  'lvl1': ["<15", "<15", "<15", "<20", "<30", "<30", "<25"],
                  'lvl2': ["<20", "<20", "<20", "<30", "<30", "<30", "<30"]}

In [ ]: conn = asdk.Connection('https://anon.protegrity.com/')

In [ ]: e = asdk.AnonElement(conn, df)

In [ ]: e['gender'] = asdk.Redact()
#Also works on attribute index
#e[0] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", "Might be <30"])

# e["race"] = asdk.Gen_Mask(maskchar="*")
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
#e["bmi"] = asdk.MicroAgg(asdk.AggregateFunction.Mean)
e.config.k = asdk.K(2)

e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TClose(tfactor=0.2)

e.config['maxSuppression'] = 0.7

In [ ]: e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)

In [ ]: e.describe()

In [ ]: job = asdk.anonymize(e, targetVariable="income", mode="Auto")

In [ ]: result = job.result()
if result.df is not None:
    print("Anon Dataframe.")
    print(result.df.head())

In [ ]: source, result = job.exploratoryStats() |

```

Note: You can use *e.measure()* to modify the request and view different outcomes of the result set.

For more information about the *measure* API, refer to the section [Measure](#).

Figure 5-3: SDK Auto Anonymization Job

5.2 Using Infer to Anonymize

Use the *Infer* API to start auto-detecting the data-domain, classification type, hierarchies, and anonymization configuration in the Protegrity Anonymization SDK. Any user-defined configuration, such as, QI attribute assignments, hierarchy, and K value, are retained and considered while performing the auto anonymization.

Ensure that you complete the following checks before starting the anonymization job:

- Verify that the destination file is not in use and that the required permissions are set for creating and modifying the destination file.
- Ensure that the disk is not full and enough free space is available for saving the destination file.

- Verify that you have imported the Pythonic SDK, for example, `import anonSDK as asdk`.

Function

`infer(targetVariable)`

Parameters

targetVariable: The field specified here is used as a focus point for performing the anonymization.

Return Type

It returns an *anon* element with all the detected classifications and hierarchies generated.

Sample Request

`e.infer(targetVariable='income')`

For more sample requests that you can use, refer to the section [Sample Requests for Protegrity Anonymization SDK](#).

```
In [ ]: import pandas as pd
import anonSDK as asdk

In [ ]: d = {'gender': ['Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Female'],
             'occupation': ['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners', 'Handlers-cleaners', 'Prof-specialty',
                             'Exec-managerial', 'Other-service', 'Exec-managerial', 'Prof-specialty'],
             'age': [39, 50, 38, 53, 28, 37, 49, 52, 31],
             'race': ['White', 'White', 'White', 'Black', 'Black', 'White', 'Black', 'White', 'White'],
             'marital-status': ['Never-married', 'Married-civ-spouse', 'Divorced', 'Married-civ-spouse',
                                'Married-civ-spouse',
                                'Married-civ-spouse', 'Married-spouse-absent', 'Married-civ-spouse', 'Never-married'],
             'education': ['Bachelors', 'Bachelors', 'HS-grad', '11th', 'Bachelors', 'Masters', '9th', 'HS-grad',
                             'Masters'],
             'native-country': ['United-States', 'United-States', 'United-States', 'United-States', 'Cuba', 'United-States',
                                 'Jamaica', 'United-States', 'United-States'],
             'workclass': ['State-gov', 'Self-emp-not-inc', 'Private', 'Private', 'Private', 'Private', 'Private',
                             'Self-emp-not-inc', 'Private'],
             'income': ['<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '<=50K', '>50K', '>50K'],
             'bmi': [11.5, 12.5, 13.5, 14.5, 16.5, 16.5, 17.5, 18.5, 11.5] }

In [ ]: df = pd.DataFrame(data=d)

In [ ]: treeGen = {'lvl0': [11, 13, 14, 15, 27, 28, 20],
                  'lvl1': ['<15', '<15', '<15', '<20', '<30', '<30', '<25'],
                  'lvl2': ['<20', '<20', '<20', '<30', '<30', '<30', '<30']}

In [ ]: conn = asdk.Connection('https://anon.protegrity.com/')

In [ ]: e = asdk.AnonElement(conn, df)

In [ ]: e.infer(targetVariable='income')

In [ ]: e.describe()

In [ ]: e['gender'] = asdk.Redact()
#Also works on attribute index
#e[0] = asdk.Redact()
e['occupation'] = asdk.Redact()
e['age'] = asdk.Gen_Tree(pd.DataFrame(data=treeGen), ["Missing", "Might be <30", "Might be <30"])

# e["race"] = asdk.Gen_Mask(maskchar="*")
e["bmi"] = asdk.Gen_Interval(['5', '10', '15'])
#e["bmi"] = asdk.MicroAgg(asdk.AggregateFunction.Mean)
e.config.k = asdk.K(2)

e["income"] = asdk.LDiv(lfactor=2)
e["income"] = asdk.TClose(tfactor=0.2)

e.config['maxSuppression'] = 0.7

In [ ]: e["race"] = asdk.Gen_Mask(maskchar="*", importance=0.8)

In [ ]: e.describe()

In [ ]: job = asdk.anonymize(e)
```

Figure 5-4: Infer Job

Note: You can use `e.measure()` to modify the request and view different outcomes of the result set.

For more information about the *measure* API, refer to the section [Measure](#).

Chapter 6

Appendix

- [6.1 Best Practices When Using the Protegrity Anonymization API](#)
- [6.2 Configuring Amazon SageMaker for the Protegrity Anonymization API](#)
- [6.3 Samples for Cloud-related Source and Destination Files](#)
- [6.4 YAML Templates](#)
- [6.5 Configuration Checklist](#)
- [6.6 Setting Up Logging for the Protegrity Anonymization API](#)
- [6.7 Creating a DNS Entry for the ELB Hostname in Route53](#)
- [6.8 Protegrity Anonymization API Risk Metrics](#)
- [6.9 Sample Data Sets](#)
- [6.10 Sample Requests for Protegrity Anonymization REST API](#)
- [6.11 Sample Requests for Protegrity Anonymization SDK](#)
- [6.12 Troubleshooting](#)

This appendix contains information, code blocks, and other information to help you with using the product.

6.1 Best Practices When Using the Protegrity Anonymization API

The following suggestions are provided for using the Protegrity Anonymization API efficiently:

- Ensure that the source file is *clean* based on the following checks:
 - A column contains correct data values. For example, a field with numbers, such as, salary, must not contain text values.
 - Appropriate text as per the coding selected is present in the files. Special characters or characters that cannot be processed must not be present in the source file.
- Move the anonymized data file and the logs generated to a different system before deleting your environment.
- The maximum dataframe size that can attach to an anonymization job is 100MB.

For processing a larger dataset size, users can use the different cloud storages available.

- **Run a maximum of 5 anonymization jobs in the Protegrity Anonymization API:** A maximum of 5 jobs can be run on the Protegrity Anonymization API for adequate utilization of resources. If more jobs are raised, then the job after the initial 5 jobs are rejected and are not processed. If required, increase the maximum limit for the **JOB_QUEUE_SIZE** parameter in the *config.yaml* file. For Docker, update the *config-docker.yaml* file.
- **Protegrity Anonymization API accepts a maximum of 60 requests per minute:** The Protegrity Anonymization API can accept a maximum of 60 request per minute. If more than 60 requests are raised, then the excess requests are rejected and are not processed. If required, increase the maximum limit for the **DEFAULT_API_RATE_LIMIT** parameter in the *config.yaml* file. For Docker, update the *config-docker.yaml* file.
- When executing anonymization scripts using a language, such as Python, ensure that you enclose the script in a *main()* block.

6.2 Configuring Amazon SageMaker for the Protegrity Anonymization API

Use the steps provided in this section to configure Amazon SageMaker for anonymizing data using the Protegrity Anonymization API. Install and configure Docker and install the Protegrity Anonymization API SDK wheel file in your environment. You can use this scenario for running anonymization jobs in an offline environment that does not have Internet access. However, you do require an Internet connection for installing the software and for creating and deploying the Protegrity Anonymization API containers.

Before you begin

Ensure that the following prerequisites are met:

- Install Docker on the Base Machine to build the Docker container.

For more information about installing Docker Desktop, navigate to <https://www.docker.com/products/docker-desktop>.

- Internet connectivity on the Base Machine to download the base image for Docker and the Protegrity Anonymization API SDK wheel file.
- Install the AWS CLI to push the Docker image to ECR.

For more information about installing the AWS CLI 2, navigate to <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>.

1. Login to the Base Machine and open a command prompt.
2. Create a directory for working with the container, such as, *img*. Ensure that you assign the necessary permissions to the folder.
3. Obtain and save the Protegrity Anonymization API SDK wheel file in the folder.
4. Create a *Dockerfile* with the following content. You can customize the configuration as per your requirements.

```
FROM tensorflow/tensorflow:2.3.0
RUN apt-get update
RUN pip install --upgrade pip
RUN pip install ipykernel && \
    python -m ipykernel install --sys-prefix && \
    pip install --quiet --no-cache-dir \
    'boto3>1.0<2.0' \
    'sagemaker>2.0<3.0'
COPY *.whl /
RUN pip install /*.whl &&\
    pip install pandas==1.0.2
```

5. Build the Docker image using the following command.

```
docker build -t pty-anonsdk .
```

6. Tag the Docker image with *sgstudio* before pushing it to ECR using the following command.

```
docker tag pty-anonsdk:latest <account_name>.dkr.ecr.region.amazonaws.com/sgstudio-anonsdk:pty-anonsdk
```

7. Push the Docker image to ECR using the following steps.

- a. Login to ECR using the following command.

```
aws ecr get-login-password --region <region> | docker login --username AWS --password-stdin <account_name>.dkr.ecr.region.amazonaws.com
```

- b. Push the Docker image to ECR using the following command.

```
docker push <account_name>.dkr.ecr.region.amazonaws.com/sgstudio-anonsdk:pty-anonsdk
```

8. Login to the AWS Management Console.

9. Navigate to **ECR > Repositories**.
10. Locate the image that you pushed to ECR.
11. Copy the **URI** of the image.
12. Navigate to **Amazon SageMaker > Images > Create Image**.
13. Specify the URI of the image that you copied and click **Next**.
14. Provide details for the image and click **Submit**. The image is created.

Figure 6-1: Specifying Image Details

15. Click **Create version** for the image.
16. Specify the image URI that was copied earlier and click **Submit**.
17. Attach the custom image to SageMaker using the following steps.
 - a. In the AWS Management Console, navigate to **Amazon SageMaker**.
 - b. From the left pane, click **Amazon SageMaker Studio**.
 - c. Click **Attach Image**.
 - d. Select **Existing Image**, select the custom image added, select the image version, and click **Next**.

Attach image to domain

Image source

Choose image source
Register an image with the SageMaker image store and then attach the image to the domain, or attach an existing image from the SageMaker image store.

☐ New image
Attaches 1 version

☒ Existing image
Attaches 1 version

Select an existing image from the SageMaker image store
An attempt to add more versions than are remaining under the version limit will fail. In that case, detach some versions and try again.

Name	DisplayName	Created
<input checked="" type="radio"/> pty-anonsdk	pty-anonsdk	Jun 30, 2021 07:02 UTC
<input type="radio"/> pty-anon	pty-anon	Jun 30, 2021 06:26 UTC
<input type="radio"/> anon-sdk	anonymization_sdk	Jun 29, 2021 08:42 UTC

Available image versions
The selected image version will be attached to the domain.

Name	Version	Created
<input checked="" type="radio"/> pty-anonsdk	8	Jun 30, 2021 14:20 UTC
<input type="radio"/> pty-anonsdk	1	Jun 30, 2021 07:02 UTC

Cancel Next

Figure 6-2: Selecting an Existing Image

- e. Specify the **Studio configuration** properties and click **Submit**.

Note: The image is configured to run as the *root* user, update the **Advance configuration** with the **UID** and **GUID** as 0.

Studio configuration

EFS mount path
The path in the image where the Studio user's EFS home directory will be mounted. [Learn more about EFS mount path](#)

/home/sagemaker-user

Kernel
The kernel included in the image. We recommend one kernel per image. [Learn more about kernels](#)

Kernel name: python3 Kernel display name - optional: pty-anonsdk Delete

1024 character max 1024 character max

Add kernel

Configuration tags - optional
Add new tag
You can add up to 50 tags.

Advanced configuration - optional
Recommended defaults have been selected for you.

By default, SageMaker runs your image as the "sagemaker-user" with a POSIX UID of 1000 and GID of 100. If your image is built with another user, enter the user as users POSIX UID and GID below (5-digit in a 10-digit field). [Learn more about UID and GID](#)

User ID (UID) - optional: 0

Group ID (GID) - optional: 0

10-digit max 10-digit max

Figure 6-3: Updating the GUID and UID

18. Launching the image.
 - a. In the AWS Management Console, navigate to **Amazon SageMaker**.
 - b. From the left pane, click **Amazon SageMaker Studio**.
 - c. From the SageMaker Studio Control panel, click **Open Studio**.
 - d. Select the custom image attached to the domain from the **Select a SageMaker image** list and launch the **Notebook** or **Image Terminal**.

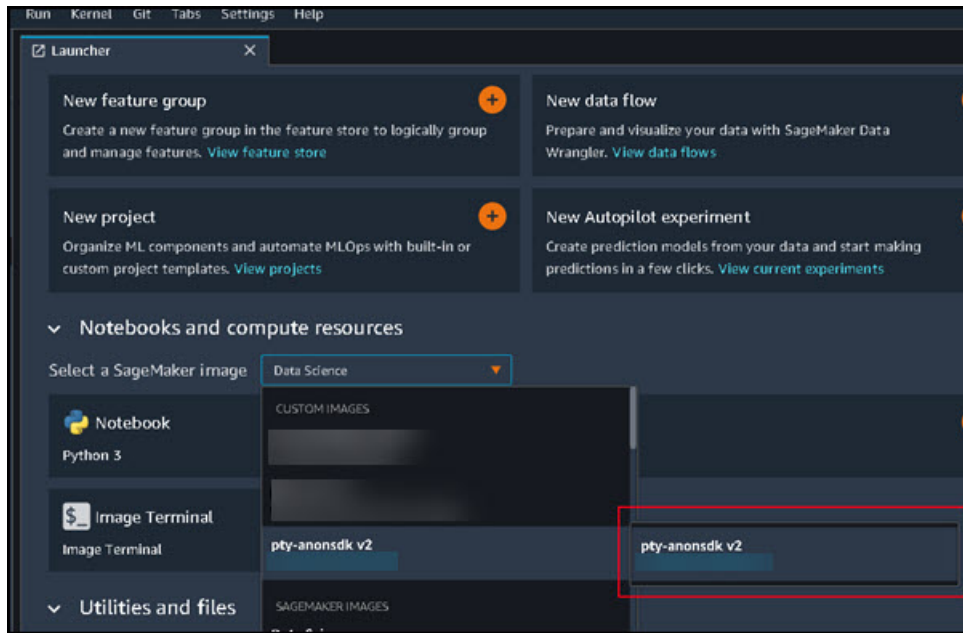


Figure 6-4: Launch Image

6.3 Samples for Cloud-related Source and Destination Files

Use the following code for specifying the source and destination for your cloud provider.

Amazon S3 bucket:

```
"source": {
  "type": "File",
  "file": {
    "name": "s3://<path_to_dataset>",
    "accessOptions": {
      "key": "API Key",
      "secret": "Secret Key"
    }
  }
}
```

Google Cloud Storage:

Note: Ensure that you install and log in to the gcloud CLI, the credentials file is created when you are successfully logged in to the gcloud CLI on the instance setup.

Complete the steps provided here to use Google Cloud Storage.

1. Login to gcloud using the following command:

```
gcloud auth login
```

2. Select your account.
3. Update the `adc.json` file with the following information.

```
{
  "scope": "https://www.googleapis.com/auth/devstorage.full_control",
  "refresh_token": "<Refresh_Token>",
  "client_email": "<User_Email_ID>",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
}
```

```
"client_id": "<Client_ID>.apps.googleusercontent.com",
"client_secret": "<Client_Secret>",
"type": "service_account"
}
```

Note: The *json* file is present in either of the following locations in your instance:

- `~/config/gcloud/application_default_credentials.json`
- `~/config/gcloud/legacy_credentials/<YOUR GOOGLE USERNAME>/adc.json`

4. Create the secret for GCP by using the following command:

```
kubectl create secret generic <secret-name> --namespace <ns-name> --from-file /path/to/adc.json
```

5. Verify if the secret was created successfully by using the command:

```
kubectl get secret -n <ns-name>
```

Update the *values.yaml* file by uncommenting the following snippet to mount that secret file to deploy the pods.

```
#volumes:
#- name: gcs-secret
#  secret:
#    secretName: <secret-name>
```

```
#volumeMounts:
#- name: gcs-secret
#  mountPath: /home/anonuser/gcs
```

After customizing the *values.yaml* file, update the configuration using the following command:

```
helm upgrade <helm name> /path/to/helmchart -n <namespace>
```

For more information about the GCP secret, refer to <https://gcsfs.readthedocs.io/en/latest/>.

You can now use the following code block for specifying the source and destination file.

```
"source": {
  "type": "File",
  "file": {
    "name": "gs://<path_to_source_file>",
    "accessOptions": {
      "token": "<mountpath_to_adc.json>",
      "project": "<Project_id>"
    }
  },
  "format": "CSV"
}
```

Azure Data Lake Storage:

```
"source": {
  "type": "File",
  "file": {
    "name": "adl://<path-to-dataset>",
    "accessOptions": {
      "tenant_id": Tenant_ID,
      "client_id": Client_ID,
      "client_secret": Client_Secret_Key
    }
  }
}
```

```
}
}
```

Azure Blob Storage:

```
"source": {
  "type": "File",
  "file": {
    "name": "abfs://<path_to_source_file>",
    "accessOptions": {
      "account_name": "<account_name>",
      "account_key": "<Account_key>"
    }
  },
  "format": "CSV"
}
```

6.4 YAML Templates

Update and use the following templates for creating the pods and services for using the Protegrity Anonymization API.

6.4.1 cluster-aws.yaml

The `cluster-aws.yaml` file contains the Protegrity Anonymization API configuration for creating the Kubernetes cluster on EKS. Use the template provided with the Protegrity Anonymization API or copy the following code to a `.yaml` file and modify it as per your requirements before running it.

Note: Ensure that you update the placeholder text before running the file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: <cluster_name>    #(provide an appropriate name for your cluster)
  region: <Region where you want to deploy Kubernetes Cluster>    #(specify the region to be
used)
  version: "1.22"
vpc:
  id: "#Update_vpc_here#" #    (enter the vpc id to be used)
  subnets:
    # (In this section specify the subnet region and subnet id accordingly)
    private:
      <Availability zone for the region where you want to deploy your Kubernetes cluster>:
        id: "#Update_id_here#"
      <Availability zone for the region where you want to deploy your Kubernetes cluster>:
        id: "#Update_id_here#"
iam:
  serviceRoleARN: "arn:aws:iam::<#Update AWS Account ID>:role/<#Update EKS SERVICE ROLE
NAME#>"
nodeGroups:
  - name: <Name of your Node Group>
    instanceType: t3a.xlarge
    minSize: 2
    maxSize: 4          # (Set max node size according to load to be processed , for cluster-
autoscaling )
    desiredCapacity: 3
    privateNetworking: true
    iam:
      attachPolicyARNs:
        - "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
        - "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
      withAddonPolicies:
        autoScaler: true
        albIngress: true
```

```

securityGroups:
  withShared: true
  withLocal: true
  attachIDs: ['#Update_security_group_id_linked_to_your_VPC_here#']
tags:
  #Add required tags (Product, name, etc.) here
  k8s.io/cluster-autoscaler/<cluster_name>: "owned"          # (Update your cluster name in
this line) ## These tags are required for
  k8s.io/cluster-autoscaler/enabled: "true"
## cluster-autoscaling
  Product: "Anonymization"
  ssh:
    publicKeyName: '<EC2 Key Pair>'          #Add SSH key to login to Nodes in
the cluster if needed.

```

6.4.1.1 Cluster Metadata

This section specifies the metadata settings required for configuring the cluster.

```

metadata:
  name: <cluster_name>    #(provide an appropriate name for your cluster)
  region: <Region where you want to deploy Kubernetes Cluster>    #(specify the region to be
used)
  version: "1.22"

```

- *name*: Specify a name for the cluster.
- *region*: Specify the AWS region for hosting the cluster.

6.4.1.2 Cloud Provider Details

This section specifies the VPC and IAM account details for the Cloud platform. A minimum of two subnets are required for Protegrity Anonymization API. You can specify additional subnets if required.

```

vpc:
  id: "#Update_vpc_here#" # (enter the vpc id to be used)
  subnets:                # (In this section specify the subnet region and subnet id accordingly)
    private:
      <Availability zone for the region where you want to deploy your Kubernetes cluster>:
        id: "#Update_id_here#"
      <Availability zone for the region where you want to deploy your Kubernetes cluster>:
        id: "#Update_id_here#"
iam:
  serviceRoleARN: "arn:aws:iam::<#Update AWS Account ID>:role/<#Update EKS SERVICE ROLE
NAME#>"

```

- *id*: Specify the VPC ID for the Cloud provider.
- *cidr*: Specify the CIDR block value.
- *us-east-1b* and *us-east-1a*: Specify the availability zone for the subnets. Additionally, specify the *id* and the *cidr* values for the subnet zones.
- *serviceRoleARN*: Specify the ARN Role details for the service role.

6.4.1.3 Node Group Details

This sections specifies the settings for configuring the node group.

```

nodeGroups:
- name: <Name of your Node Group>
  instanceType: t3a.xlarge
  minSize: 2
  maxSize: 4          # (Set max node size according to load to be processed , for cluster-

```

```

autoscaling )
  desiredCapacity: 3
  privateNetworking: true
  iam:
    attachPolicyARNs:
      - "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
      - "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
    withAddonPolicies:
      autoScaler: true
      albIngress: true
  securityGroups:
    withShared: true
    withLocal: true
    attachIDs: ['#Update_security_group_id_linked_to_your_VPC_here#']
  tags:
    #Add required tags (Product, name, etc.) here
    k8s.io/cluster-autoscaler/<cluster_name>: "owned"          # (Update your cluster name in
this line) ## These tags are required for
    k8s.io/cluster-autoscaler/enabled: "true"
## cluster-autoscaling
  Product: "Anonymization"
  ssh:
    publicKeyName: '<EC2 Key Pair>'          #Add SSH key to login to Nodes in
the cluster if needed.

```

- *name*: Specify a name for the node group.
- *instanceType*: Specify the instance size for the machine.
- *maxSize*: Specify the maximum nodes to be used in the cluster for autoscaling.
- *attachIDs*: Specify the security group IDs.
- *tags*: Specify the tags for the node group.
- *k8s.io/cluster-autoscaler/<cluster_name>*: Specify the cluster name. This is the value specified in the *metadata*.
- *publicKeyName*: This is the key pair of the EC2 instance for creating the Kubernetes cluster.

6.4.2 values.yaml

The *values.yaml* file contains the configuration for setting up the Protegrity Anonymization API. Use the template provided with the Protegrity Anonymization API or copy the following code to a *.yaml* file and modify it as per your requirements before running it.

```

## PREREQUISITES
## Create separate namespace. Eg: kubectl create ns anon-ns. Update your namespace name in
values.yaml.

## Running all pods in the namespace specific for Protegrity Anonymization API
namespace:
  name: anon-ns          # Update the namespace if required.

## Prerequisite for setting up Database Pod.
## This is to handle any new DB pod getting created that uses the same persistence storage in
case the running Database pod gets disrupted.
dbpersistence:
  ## 1. Get the list of nodes in the cluster. CMD: kubectl get nodes
  ## 2. Get the node name which is running in the same zone where the external-storage is
created. CMD: kubectl describe nodes
  nodename: "<Node_name>"          # Update the Node name

  ## Fetch the zone in which the node is running using the `kubectl describe node/nodename`
command or the following command.
  ## CMD: ` kubectl describe node/<nodename> | grep topology.kubernetes.io/zone | grep -oP
'topology.kubernetes.io/zone=\K[^\s]+' `
  zone: "<Zone in which above Node is running>"

## For EKS cluster, supply the volumeID of the aws-ebs
## For AKS cluster, supply the subscriptionID of the azure-disk
storageId: "<Provide storage ID>"

```

```

fsType: ext4

## This section is required if the image is getting pulled from the Azure Container Registry
## create image pull secrets and specify the name here.
## remove the [] after 'imagePullSecrets:' once you specify the secrets
#imagePullSecrets: []
# - name: regcred

image:
  repository: <Repo_path>                                # Repo path for the Container Registry
in Azure, GCP, AWS

  anonapi_tag: <AnonImage_tag>                            # Tag name of the ANON-API Image.
  anonstorage_tag: <StorageImage_tag>                    # Tag name of the ANON-Storage Image.
  anonworkstation_tag: <WorkstationImage_tag>            # Tag name of the ANON-Workstation Image.

pullPolicy: Always

## Refer to the section in the documentation for setting up and configuring NGINX-INGRESS
before deploying the application.
ingress:
  ## Add the host section with the hostname used as CN while creating server certificates.
  ## While creating the certificates you can use *.protegrity.com as CN and SAN as used in
the below example
  host: anon.protegrity.com                                # Update the host according to your server
certificates.

  ## To terminate TLS on the Ingress Controller Load Balancer.
  ## K8s TLS Secret containing the certificate and key must be provided.
  secret: anon-protegrity-tls                             # Update the secretName according to your
secretName.

  ## To validate the client certificate with the above server certificate
  ## Create the secret of the CA certificate used to sign both the server and client
certificate as shown in the example below
  ca_secret: ca-protegrity                                # Update the ca-secretName according to your
secretName.

  ingress_class: nginx-anon

## Creating a service account for Anonymization
serviceaccount:
  name: anon-service-account

## Setting the pod security context
podSecurityContext:
  runAsNonRoot: true
  runAsUser: 1000
  fsGroup: 1000

anon_service:
  name: anon-dask-svc
  daskMasterPort: 8786
  daskUiPort: 8787
  anonPort: 8090
  labels:
    appname: anon

# Configure the delays for Liveness Probe here
livenessProbe:
  initialDelaySeconds: 50
  periodSeconds: 40

#Configure the delays for Readiness Probe here
readinessProbe:
  initialDelaySeconds: 15
  periodSeconds: 20

##### WORKER CONFIGURATIONS #####
## Increase the number of worker pods as per your requirement
workers:
  labels:

```



```

    app: dask-worker
    replicaCount: 1

## Resources defined for the worker pod
worker_resources:
  requests:
    cpu: 2
    memory: 6Gi
  limits:
    cpu: 2
    memory: 6Gi

## Specs with which worker container should start
containerSpecs:
  memLimit: "6G"
  nthreads: 2

## Worker pod env to read values from configMap manifest.
## A config Map(wrkr-specs) is used to set these values.
workerPodEnv:
  - name: worker_mem_limit
    valueFrom:
      configMapKeyRef:
        name: wrkr-specs
        key: worker-mem-limit
  - name: num_threads
    valueFrom:
      configMapKeyRef:
        name: wrkr-specs
        key: num-threads

  autoscaling:
    minReplicas: 1                                # Min number of worker pods which will be running
    when the cluster starts.
    maxReplicas: 3                                # Max number of worker pods which will autoscale in
    the cluster.
    targetMemoryThreshold: 4Gi                    # Threshold memory-load beyond which worker pods
    will autoscale.

## FOR MORE INFO ABOUT PROCESSING LARGE DATASETS REFER TO THE DOCUMENTATION
#####

## Create the volumes and specify the names here.
## remove the [] after 'volumes:' once you specify volumes
volumes: []
  #- name: gcs-secret                             ##This secret is used when user wants to read and write
  data to a google cloud storage Refer DOC.
  #secret:
  #secretName: adc-gcs-creds

## Create the volumeMounts and specify the names here.
## remove the [] after 'volumeMounts:' once you specify volumeMounts
volumeMounts: []
  #- name: gcs-secret
  #mountPath: /home/anonuser/gcs

## ANON-STORAGE ##
storage:
  namenode_name: pty-namenode
  datanode_name: pty-datanode

  service:
    namenode_svc: pty-namenode-svc
    datanode_svc: pty-datanode-svc
    nn_dn_rpc: 8020
    nn_http: 9870
    dn_http: 9864

  labels:
    namenode_app: pty-namenode
    datanode_app: pty-datanode

storage_resources:

```

```

    requests:
      cpu: 1
      memory: 3Gi
    limits:
      cpu: 1
      memory: 3Gi

  volumeMounts:
    namenode:
      - name: hdfs-name
        mountPath: /hadoop/dfs/name
    datanode:
      - name: hdfs-data
        mountPath: /hadoop/dfs/data

  volumes:
    - name: hdfs-name
      hostPath:
        path: /var/hdfs-name
    - name: hdfs-data
      hostPath:
        path: /var/hdfs-data

## ANON-DATABASE ##
database:
  name: anondb
  labels:
    app: anon-db

  service:
    name: anon-db-svc
    dbport: 5432

  persistence:    ## Persistence Volume size
    pvName: anon-db-pv
    pvcName: anon-db-pvc
    accessMode: ReadWriteOnce
    storageDB:
      size: 20Gi

## ANON-WORKSTATION ##
anonlab:
  name: anonworkstation
  labels:
    app: anon-lab

  service:
    name: anonlab-svc
    labport: 8888

```

6.4.2.1 Basic Configuration

This section specifies the settings for configuring the namespace for anonymization.

```

## Running all pods in the namespace specific for Protegrity Anonymization API
namespace:
  name: anon-ns                                # Update the namespace if required.

## Creating a service account for Anonymization
serviceaccount:
  name: anon-service-account

## Setting pod security context
podSecurityContext:
  runAsNonRoot: true
  runAsUser: 1000
  fsGroup: 1000

labels:
  appname: anon

## This section is if image is getting pulled from Azure Container Registry

```

```
## create image pull secrets and specify the name here.
## remove the [] after 'imagePullSecrets:' once you specify the secrets
#imagePullSecrets: []
#   - name: regcred
```

- *name*: Specifies the namespace name for the pods.
- *name*: Specifies the name for the service account.
- *podSecurityContext*: Specifies the pod security settings.
- *appname*: Specifies the label for the pods.
- *imagePullSecrets*: Specifies the secret information for working with containers. Uncomment the lines, delete the `[]`, and specify the secrets information.

Note: This is only applicable for the Azure cloud platform.

6.4.2.2 Image Configuration

This section specifies the image settings for anonymization.

```
image:
  repository: <Repo_path>           #Repo path for Container Registry in
  Azure, GCP, AWS                   #Tag name of the API Image.
                                     #Tag name of Storage Image.
  anonapi_tag: <AnonImage_tag>      #Tag name of Workstation Image.
  anonstorage_tag: <StorageImage_tag>
  anonworkstation_tag: <WorkstationImage_tag>
  pullPolicy: Always
```

- *repository*: Specifies the registry location to the anonymization container. This is the Cloud location where the container image is uploaded.
- *anonapi_tag*: Specifies the tag assigned to the uploaded API image. This is the tag name you upload the image to the Cloud container registry.
- *anonstorage_tag*: Specifies the tag assigned to the uploaded storage image. This is the tag name you upload the image to the Cloud container registry.
- *anonworkstation_tag*: Specifies the tag assigned to the uploaded Workstation image. This is the tag name you upload the image to the Cloud container registry.
- *pullPolicy*: Specifies whether the image must be pulled from the repository.

6.4.2.3 REST Service Configuration

This section specifies the settings for anonymization REST service.

```
service:
  name: anon-dask-svc
  daskMasterPort: 8786
  daskUiPort: 8787
  anonPort: 8090
```

- *name*: Specifies a name for the REST service.
- *anonPort*: Specifies the port on which the anonymization API runs.

6.4.2.4 Health Check Configuration

This section specifies the the intervals required to run health checks for the API container images.

```
# Configure the delays for Liveness Probe here
livenessProbe:
  initialDelaySeconds: 50
  periodSeconds: 40

#Configure the delays for Readiness Probe here
readinessProbe:
  initialDelaySeconds: 15
  periodSeconds: 20
```

6.4.2.5 Worker Pod Configuration

This section specifies the settings for configuring the worker pods for processing the anonymization requests.

```
    memory: 6Gi
  limits:
    cpu: 2
    memory: 6Gi

## Specs with which worker container should start
workers:
  replicaCount: 1

## Resources defined for worker pod
worker_resources:
  requests:
    cpu: 2
    memory: 6Gi
  limits:
    cpu: 2
    memory: 6Gi

## Specs with which worker container should start
containerSpecs:
  memLimit: "6G"
  nthreads: 2

## Worker pod env to read values from configMap manifest.
## A config Map(wrkr-specs) is used to set these values.
workerPodEnv:
  - name: worker_mem_limit
    valueFrom:
      configMapKeyRef:
        name: wrkr-specs
        key: worker-mem-limit
  - name: num_threads
    valueFrom:
      configMapKeyRef:
        name: wrkr-specs
        key: num-threads

  autoscaling:
    minReplicas: 1
    maxReplicas: 3
    targetMemoryThreshold: 60
the cluster.
will autoscale. # Min number of worker pods which will be Running.
# Max number of worker pods which will autoscale in
# Threshold memory-load beyond which worker pods
```

- *replicaCount*: Specifies the number of worker pods to create for the anonymization request.

Note: If autoscaling is disabled, then increase the *replicaCount* to process large files.

- *worker_resources* and *containerSpecs*: Configure the resources allocated for the worker pods here.

Note: Ensure that the *nthreads* value matches the *cpu* value. Also, the *memLimit* value must match the *memory* value.

- *autoscaling*: Specifies the autoscaling settings, such as, the minimum number of pods, maximum number of pods, and target memory settings for the pods.

Note: If autoscaling is disabled, then the *replicaCount* and *maxReplicas* values must be the same.

- *minReplicas*: This value specifies the minimum number of worker pods that will be running.
- *maxReplicas*: This value specifies the maximum number of worker pods that can be autoscaled.
- *targetMemoryThreshold*: This value specifies the threshold value beyond which the worker pods will be autoscaled.

6.4.2.6 Storage Configuration

This section specifies the settings for configuring the anonymization storage.

```
## ANON-STORAGE ##
storage:
  namenode_name: pty-namenode
  datanode_name: pty-datanode

  service:
    namenode_svc: pty-namenode-svc
    datanode_svc: pty-datanode-svc
    nn_dn_rpc: 8020
    nn_http: 9870
    dn_http: 9864
```

- *namenode_svc*: This service is used by the *Anon-SDK* client and *Anon-API* container to communicate with the namenode pod.
- *datanode_svc*: This service is used by the *Anon-SDK* client and *Anon-API* container to communicate with the datanode pod.

6.4.2.7 Database Configuration

This section specifies the settings for configuring the database for storing the anonymized data.

```
## ANON-DATABASE ##
database:
  name: anondb
  labels:
    app: anon-db

  service:
    name: anon-db-svc
    dbport: 5432

  persistence:    ## Persistence Volume size
    storageDB:
      size: 20Gi
```

- *name*: Specifies the database service name that is used for communication with the Anon API container.
- *size*: Specifies the EBS persistent volume claim size.

6.4.3 docker-compose.yaml

The *docker-compose.yaml* file contains the configuration for setting up the Protegrity Anonymization API. Use the template provided with the Protegrity Anonymization API or copy the following code to a *.yaml* file and modify it as per your requirements before running it.

Note: Ensure that you update the placeholder text before running the file.

```
version: "3.1"

services:
  scheduler:
    image: <Image Name/Id>:latest
    hostname: pty-dask-scheduler
    container_name: pty-scheduler
    ports:
      - "8786:8786"
      - "8787:8787"
    command: [ 'scheduler -c docker' ]
    restart: unless-stopped
    networks:
      - pty-network
  anon:
    image: <Image Name/Id>:latest
    hostname: pty-anon-app
    container_name: pty-anon-app
    ports:
      - "8090:8090"
    command: [ 'anon -c docker' ]
    restart: unless-stopped
    networks:
      - pty-network
  pty-worker:
    image: <Image Name/Id>:latest
    environment: ##Provide worker mem limit and number of threads according to resources
    available
      - worker_mem_limit=6G
      - num_threads=2
    depends_on:
      - scheduler
    command: [ 'worker -c docker' ]
    deploy:
      mode: replicated
      replicas: 2
    restart: unless-stopped
    networks:
      - pty-network
  anondb:
    image: <Image Name/Id>:latest
    hostname: pty-anondb
    user: postgres
    container_name: pty-anondb
    ports:
      - "5432:5432"
    command: [ 'database -c docker' ]
    restart: unless-stopped
    networks:
      - pty-network
  namenode:
    image: <Image Name/Id>:latest
    container_name: pty-namenode-svc
    hostname: pty-namenode-svc
    ports:
      - 9870:9870
      - 8020:8020
    volumes:
      - hadoop_namenode:/hadoop/dfs/name
    environment:
      - CLUSTER_NAME=anon-storage
```

```

    env_file:
      - ptystorage.env
    command: [ "/bin/bash", "/home/anonuser/start.sh", "namenode" ]
    restart: unless-stopped
    networks:
      - pty-network
  datanode:
    image: <Image Name/Id>:latest
    container_name: pty-datanode
    hostname: pty-datanode
    ports:
      - 9864:9864
      - 9866:9866
    volumes:
      - hadoop_datanode:/hadoop/dfs/data
    environment:
      SERVICE_PRECONDITION: "pty-namenode-svc:8020"
    env_file:
      - ptystorage.env
    command: [ "/bin/bash", "/home/anonuser/start.sh", "datanode" ]
    restart: unless-stopped
    networks:
      - pty-network
  nginx-proxy:
    image: nginx:1.20.1
    restart: unless-stopped
    ports:
      - 443:443
      - 80:80
    volumes:
      - ./cert:/cert/:Z
      - ./nginx.conf:/etc/nginx/nginx.conf:Z
    networks:
      - pty-network
    command: [ nginx, '-g', 'daemon off;' ]
  workstation:
    image: <Image Name/Id>:latest
    restart: unless-stopped
    hostname: workstation
    container_name: pty-workstation
    # extra_hosts: ##### Uncomment and edit this section for using jupyter-workstation
    # to send request to Protegrity Anonymization-API
    #   - "anon.protegrity.com: <IP_of_host_machine>"
    ports:
      - "8888:8888"
    networks:
      - pty-network
    environment:
      JUPYTER_ENABLE_LAB: "yes"
      RESTARTABLE: "yes"
    command: "jupyter-lab --NotebookApp.token='' --NotebookApp.password='' --
NotebookApp.ip='0.0.0.0' --no-browser"

  volumes:
    hadoop_namenode:
    hadoop_datanode:
  networks:
    pty-network:

```

6.5 Configuration Checklist

Use these checklists to obtain information from your Cloud provider. This information is used to update the configuration information in the *yaml* files.

6.5.1 AWS Checklist

Update the table using from your AWS account to configure the Protegrity Anonymization API.

CLI Installation:

Variable	Value	Obtain from...
AWS Access Key ID		AWS > IAM > Users > <user_name> > Security credentials > Access key ID.
AWS Secret Access Key		https://aws.amazon.com/blogs/security/how-to-find-update-access-keys-password-mfa-aws-management-console/
Default region name		AWS > EC2 > Region name from the upper-right corner
Default output format	json	

cluster-aws.yaml:

Variable	Value	Obtain from...
metadata:		
name		Specify a name.
region		AWS > EC2 > Region name from the upper-right corner
vpc:		
id		AWS > EC2 > Instance_Id > Networking > VPC ID
cidr		AWS > EC2 > Instance_Id > VPC_Id > IPv4 CIDR
subnets:		
private:		
us-east-1a:		AWS > VPC > Subnets > Subnet > Availability Zone
id		AWS > VPC > Subnets > Subnet > Subnet ID
cidr		AWS > VPC > Subnets > Subnet > IPv4 CIDR
us-east-1b:		AWS > VPC > Subnets > Subnet > Availability Zone
id		AWS > VPC > Subnets > Subnet > Subnet ID
cidr		AWS > VPC > Subnets > Subnet > IPv4 CIDR
nodeGroups:		
securityGroups		
attachIDs		AWS > VPC > Security Groups > security_group > Security group ID

6.6 Setting Up Logging for the Protegrity Anonymization API

Logging is helpful to know the tasks being performed on the system. It is especially helpful to trace and resolve errors in the configuration and to see that a software is processing a request and is not stalled. You need to set up logging for the Protegrity Anonymization API if you require it. In logging, Protegrity Anonymization API captures the internal processing and saves it in a log file that you can view for further analysis. Update and use the script files provided here for logging as per your requirements.

Note: This is an alternative way for obtaining logs.

1. Navigate to the machine where the Protegrity Anonymization API is set up.

2. Use the `Anon_logs.sh` script to pull the logs for the task being performed in the Protegrity Anonymization API pod.
3. Assign the appropriate permissions and run the `Anon_logs.sh` script.

```
chmod +x Anon_logs.sh
./<path_to_script>/Anon_logs.sh
```

6.7 Creating a DNS Entry for the ELB Hostname in Route53

This section describes the steps to configure hostnames specified in the `values.yaml` file of the Helm chart for resolving the hostname of the Elastic Load Balancer (ELB) that is created by the NGINX Ingress Controller.

1. Configure Route53 for DNS resolution.
 - Create a private hosted zone in the Route53 service.
 - In our case, the domain name for the hosted zone is *protegrity.com*.
 - Select the VPC where the Kubernetes cluster is created.
2. Create a hostname for the ELB in the private hosted zone created in step 1.
 - Create a Record Set with type A - Ipv4 address
 - Select Alias as *yes*
 - Specify the Alias Target to the ELB created by the Nginx Ingress Controller
3. Save the record Create Inbound endpoint for DNS queries from a network to the hosted VPC used in Kubernetes.
 - Select Configure endpoints in the Route53 Resolver service.
 - Select *Inbound Only* endpoint.
 - Give a name to the endpoint.
 - Select the VPC used in the Kubernetes cluster and Route53 private hosted zone.
 - Select the availability zone as per the subnet.
 - Review and create the endpoint.
 - Note the IP addresses from the Inbound endpoint page.
 - Send CURL request to the hostname created using the Route 53 service

For more information about Amazon Route53, refer to the [Amazon Route53](#) documentation.

For more information about Azure DNS, refer to the [Azure DNS](#) documentation.

6.8 Protegrity Anonymization API Risk Metrics

This section describes how the risk metrics are derived. It details the descriptions and the equations used to calculate the risk.

6.8.1 Definitions

The following definitions are used for risk calculations:

- **Data Provider or Custodian:** The custodian of the data, responsible for controlling the process of sharing by anonymizing the data as well as putting in place other controls which prevents data from being misused and or re-identified.
- **Data Recipient:** Person or institution who receives the data from the data provider.
- **Data Set:** The collection of all records containing the data on subjects.
- **Adversary:** Data recipient who has the motives to attempt and means to succeed the re-identification of the data and intends to use the data in ways which may be harmful to individuals contained in the data set.

- **Target:** Person whose details are in the data set who has been selected by the adversary to focus the re-identification attempt on.

6.8.2 Types of Risks

The Protegrity Anonymization API uses the Prosecutor, Journalist and Marketer risk models to assess probability of re-identification attacks. A description of these risks are provided here.

- **Prosecutor Risk:** If the adversary can know that the target is in the data set, then it is called Prosecutor Risk. The fact that target is part of data set increases the risk of successful re-identification.
- **Journalist Risk:** When the adversary doesn't know for certain that the target is in the data set then it is called Journalist Risk.
- **Marketer Risk:** Under Marketer Risk, the adversary attempts to re-identify as many subjects in the data set as possible. If the risk of re-identifying an individual subject is possible, then the risk of multiple subjects being re-identified is also possible.

Relationship between the three risks

Prosecutor Risk \geq Journalist Risk \geq Marketer Risk

If the data set is protected against the prosecutor and the journalist risk, depending on the adversary's knowledge of target's participation, then by default it is also protected against the marketer risk.

6.8.3 Measuring Risks

This section details the strategy used by Protegrity Anonymization API to calculate risks.

For calculating risks, the population is the entire pool from which the sample data set is drawn. In the current calculation of risk metrics, the population considered is the same as the sample. In case of journalist calculation, it is good to consider the population from a larger data set from which the sample is drawn.

The following annotations are used in the calculations:

- R_a is the proportion of records with risk above the threshold which is at highest risk.
- R_b is the maximum probability of re-identification which is at maximum risk.
- R_c is the proportion of records that can be re-identified on an average which is the success rate of re-identification.

As part of the risk calculations, anonymization API calculates the following metrics:

- $_pR_a$ is the highest prosecutor risk.
- $_pR_b$ is the maximum prosecutor risk.
- $_pR_c$ is the success rate of prosecutor risk.
- $_jR_a$ is the highest journalist risk.
- $_jR_b$ is the maximum journalist risk.
- $_jR_c$ is the success rate of journalist risk.
- $_mR_c$ is the success rate of marketer risk.

Table 6-1: Risk Equations

Risk Type	Equation	Notes
Prosecutor	$_pR_a = 1/n \sum_j f_j \times 1(f_j > T)$ $_pR_b = 1 / \min(f_j)$ $_pR_c = J / n$	<ul style="list-style-type: none"> • f_j size of equivalence class in the sample. • F_j size of equivalence class in the population. • $f_j = F_j$ if sample is same as population. • n is number of records in the sample.

Risk Type	Equation	Notes
		<ul style="list-style-type: none"> T is the risk threshold which is the highest allowable probability of correctly re-identifying single record. Value of T in the calculation is 0.1.
Journalist	${}_jR_a = 1/n - {}_j f_j \times 1(1 / F_j > T)$ ${}_jR_b = 1 / \min(F_j)$ ${}_jR_c = \max (J / F_j) , 1 / n - f_j / F_j$	<ul style="list-style-type: none"> f_j size of equivalence class in the sample. F_j size of equivalence class in the population. $f_j = F_j$ if sample is same as population. n is number of records in the sample. T is the risk threshold. Value of T in the calculation is 0.1.
Marketer	${}_mR_c = 1/n - {}_j f_j / F_j$	<ul style="list-style-type: none"> n is number of records in the sample. f_j size of equivalence class in the sample F_j size of equivalence class in the population.

6.8.3.1 Measuring Prosecutor Risk

You can understand how the prosecutor risk is measured with the following use case. In this use case, an adversary is attempting to re-identify a specific target, Alice. The adversary knows that Alice is in the published data set. Thus, the Prosecutor Risk criteria is met.

In the following example, the original dataset contains patient information and some prescription information, such as, the DIN (drug identification number). In the published data set, the patient name which is a direct identifier is suppressed. The Year Of Birth is anonymized by generalizing.

In this scenario, the adversary has some background information about Alice, such as, the year of birth and gender. The adversary also knows that Alice is in the published data set.

Table 6-2: Original Dataset

Name	Gender	Year Of Birth	DIN
John Smith	Male	1979	2046059
Alan Patel	Male	1982	716839
Hercules Green	Male	1979	2241497
Gill Stringer	Female	1995	2046059
Alice Smith	Female	1987	392537
Bill Nash	Male	1995	363766
Albert Blackwell	Male	1998	544981
Beverly McCulsky	Female	1984	293512
Douglas Henry	Male	1979	544981
Freda Shields	Female	1995	596612
Fred Thompson	Male	1987	72565

Table 6-3: De-Identification

Gender	Decade Of Birth	DIN
Male	1970-1979	2046059
Male	1980-1989	716839
Male	1970-1979	2241497

Gender	Decade Of Birth	DIN
Female	190-1999	2046059
Female	1980-1989	392537
Male	1990-1999	363766
Male	1990-1999	544981
Female	1980-1989	293512
Male	1970-1979	544981
Female	1990-1999	596612
Male	1980-1989	72565

Published Data Set: **Alice Smith Gender: Female DOB:1987**

The adversary knows that Alice is in the published data set and can match on the year of birth and gender. There are two records that match the required criteria. Therefore, the matching equivalence class size is 2. Since the adversary does not know which one of those records pertains to Alice, the adversary will select one as random.

Here, the probability of re-identification is 0.5.

Hence, the probability of correct re-identification is:

$$p_j = 1 / f_j$$

Where, f_j is the size of the matching equivalence class in the published data set.

Thus, you can calculate the prosecutor risk of each equivalence class in the published data set.

Using prosecutor risk of each equivalence classes you can calculate the derived metrics using prosecutor risk equations assuming value as 0.33.

The following table shows the Prosecutor Risk of re-identification for every equivalence class in the example data set.

Table 6-4: Equivalence Class

Gender	Year Of Birth	p_j
Male	1970-1979	0.33
Male	1980-1989	0.5
Male	1990-1999	0.5
Female	1990-1999	0.5
Female	1980-1989	0.5

Table 6-5: Computations of All Three Types of Derived Prosecutor Risk Metrics

Derived Risk Metrics	Equation	Risk Value
pR_a	$1/n \sum_j f_j \times I(1 / f_j > T)$	0.73
pR_b	$1 / \min(f_j)$	0.5
pR_c	$ J / n$	0.45

Calculation of pR_a :

1. Consider T value as 0.33. $I(.)$, identify function returns 0, 1, 1, 1, 1.
2. Equivalence class size are 3, 2, 2, 2, 2 in published data set.
3. Identity function value multiply by size of equivalence size returns 0, 2, 2, 2, 2.
4. Total 11 records in sample. Value of step 3 / total records returns 0, 0.1818, 0.1818, 0.1818, 0.1818.

- Total of Step 4 returns 0.73, which value of pRa

Calculation of pRb:

- Minimum size of equivalence class is 2
- Value of $\frac{1}{2}$ is 0.5. Value of pRb is 0.5

Calculation of pRc:

- Total equivalence classes are 5
- Total equivalence classes / total records = $5 / 11 = 0.45$.
- Value of pRc is 0.45

6.8.3.2 Measuring Journalist Risk

For Journalist Risk to be applied, the published data set should be a specific sample.

There are two general types of re-identification attacks under journalist risk:

- The adversary is targeting a specific individual.
- The adversary is targeting any individual.

In case of journalist attack, the adversary will match the published data set with another identification data set, such as, voter registry, all patient data in hospital, and so on.

Identification of the data set represents the population of which the published data set is a sample.

For example, the sample published data set is drawn from the identification data set.

Published Dataset		Identification Dataset	
f1 = 4	[50, Male]	[50, Male]	F1 = 10
f2 = 3	[50, Female]	[50, Female]	F2 = 8
f3 = 2	[35, Male]	[35, Male]	F3 = 14
f4 = 1	[35, Female]	[35, Female]	F4 = 4
		[65, Female]	F5 = 2

Figure 6-5: Datasets

Table 6-6: Derived Risk metrics of Journalist Risk:

Derived Risk Metrics	Equation	Risk Value
jRa	$1/n \sum_j f_j \times I(1/F_j > T)$	0
jRb	$1 / \min(F_j)$	0.25
jRc	$\max(J / F_j, 1/n \sum_j f_j / F_j)$	0.13

Calculation of jRa:

- T value is 0.33. Size of equivalence classes in the identity dataset are 10, 8, 14, 4, 2.
- Identity function returns 0 when value $1/F$ is less than value else 1.
- Identify function returns 0, 0, 0, 0, 1.
- Equivalence sizes in samples are 4, 3, 2, 1.
- Values of equivalence size / number of records are 0.4, 0.3, 0.2, 0.1.

- Product of above value with identity function values are 0, 0, 0, 0.
- Value of jRa is **0**.

Calculation of jRb:

- Minimum of equivalence size of identification data set is 4
- Value of jRb is **0.25**.

Calculation of jRc:

- Number of equivalence classes in 5 in identification data set.
- Total records in identification data set 38.
- Number of equivalence classes / total records = $5/38 = 0.131$.
- Equivalence classes in sample / equivalence classes in identification data set are 0.4, 0.375, 0/142857, 0/25.
- Total of above values 1.16.
- Above value / total records in sample = $1/16 / 10 = 0.116$.
- Max (0.131, 0.116) = **0.131**.

6.8.3.3 Measuring Marketer Risk

The use case for deriving the marketer risk is shown here.

Table 6-7: Derived Risk Metrics of Marketer Risk

Derived Risk Metrics	Equation	Risk Value
mRc	$1/n \sum f_j / F_j$	0.116

Calculation of mRc:

- Equivalence classes in sample / equivalence classes in identification data set are 0.4, 0.375, 0/142857, 0/25.
- Total of above values 1.16.
- Above value / total records in sample = $1/16 / 10 = 0.116$.
- Value of marketer risk is **0.116**.

6.9 Sample Data Sets

Use the following data set to test the Protegrity Anonymization API. This data set is comprehensive and can give you thorough insights about working with the Protegrity Anonymization API.

Adult Dataset: Here is an extract of the data set, the complete data set can be found in the *adult.csv* file in the *samples* directory.

```
sex;age;race;marital-status;education;native-
country;citizenSince;weight;workclass;occupation;salary-class
Male;39;White;Never-married;Bachelors;United-States;08-01-1971;185.38;State-gov;Adm-
clerical;<=50K
Male;50;White;Married-civ-spouse;Bachelors;United-States;19-04-1960;176.32;Self-emp-not-
inc;Exec-managerial;<=50K
Male;38;White;Divorced;HS-grad;United-States;07-12-1971;159.13;Private;Handlers-cleaners;<=50K
Male;53;Black;Married-civ-spouse;11th;United-States;22-05-1957;170.45;Private;Handlers-
cleaners;<=50K
Female;28;Black;Married-civ-spouse;Bachelors;Cuba;03-02-1982;178.79;Private;Prof-
specialty;<=50K
Female;37;White;Married-civ-spouse;Masters;United-States;06-12-1972;161.65;Private;Exec-
managerial;<=50K
Female;49;Black;Married-spouse-absent;9th;Jamaica;18-04-1961;162.73;Private;Other-
service;<=50K
Male;52;White;Married-civ-spouse;HS-grad;United-States;21-05-1958;171.75;Self-emp-not-
```

```

inc;Exec-managerial;>50K
Female;31;White;Never-married;Masters;United-States;31-12-1978;164.03;Private;Prof-
specialty;>50K
Male;42;White;Married-civ-spouse;Bachelors;United-States;11-02-1968;186.33;Private;Exec-
managerial;>50K
Male;37;Black;Married-civ-spouse;Some-college;United-States;06-12-1972;189.49;Private;Exec-
managerial;>50K
Male;30;Asian-Pac-Islander;Married-civ-spouse;Bachelors;India;01-02-1980;178.70;State-
gov;Prof-specialty;>50K
Female;23;White;Never-married;Bachelors;United-States;08-04-1987;183.22;Private;Adm-
clerical;<=50K
Male;32;Black;Never-married;Assoc-acdm;United-States;01-01-1978;156.63;Private;Sales;<=50K
Male;34;Amer-Indian-Eskimo;Married-civ-
spouse;7th-8th;Mexico;03-12-1975;173.41;Private;Transport-moving;<=50K
Male;25;White;Never-married;HS-grad;United-States;06-03-1985;170.72;Self-emp-not-inc;Farming-
fishing;<=50K
Male;32;White;Never-married;HS-grad;United-States;01-01-1978;174.91;Private;Machine-op-
inspct;<=50K
Male;38;White;Married-civ-spouse;11th;United-States;07-12-1971;176.47;Private;Sales;<=50K
Female;43;White;Divorced;Masters;United-States;12-02-1967;179.88;Self-emp-not-inc;Exec-
managerial;>50K
Male;40;White;Married-civ-spouse;Doctorate;United-States;09-01-1970;170.80;Private;Prof-
specialty;>50K
Female;54;Black;Separated;HS-grad;United-States;23-06-1956;171.61;Private;Other-service;<=50K
Male;35;Black;Married-civ-spouse;9th;United-States;04-12-1974;183.71;Federal-gov;Farming-
fishing;<=50K
Male;43;White;Married-civ-spouse;11th;United-States;12-02-1967;158.63;Private;Transport-
moving;<=50K
Female;59;White;Divorced;HS-grad;United-States;28-07-1951;181.64;Private;Tech-support;<=50K
Male;56;White;Married-civ-spouse;Bachelors;United-States;25-06-1954;171.80;Local-gov;Tech-
support;>50K
Male;19;White;Never-married;HS-grad;United-States;12-05-1991;172.74;Private;Craft-repair;<=50K
Male;39;White;Divorced;HS-grad;United-States;08-01-1971;159.41;Private;Exec-managerial;<=50K
Male;49;White;Married-civ-spouse;HS-grad;United-States;18-04-1961;176.76;Private;Craft-
repair;<=50K
Male;23;White;Never-married;Assoc-acdm;United-States;08-04-1987;164.43;Local-gov;Protective-
serv;<=50K
Male;20;Black;Never-married;Some-college;United-States;11-05-1990;157.60;Private;Sales;<=50K
Male;45;White;Divorced;Bachelors;United-States;14-03-1965;176.38;Private;Exec-managerial;<=50K
Male;30;White;Married-civ-spouse;Some-college;United-States;01-02-1980;160.60;Federal-gov;Adm-
clerical;<=50K
Male;22;Black;Married-civ-spouse;Some-college;United-States;09-04-1988;173.41;State-gov;Other-
service;<=50K
Male;48;White;Never-married;11th;Puerto-Rico;17-04-1962;189.50;Private;Machine-op-inspct;<=50K
Male;21;White;Never-married;Some-college;United-States;10-05-1989;162.76;Private;Machine-op-
inspct;<=50K
Female;19;White;Married-AF-spouse;HS-grad;United-States;12-05-1991;158.42;Private;Adm-
clerical;<=50K
Male;48;White;Married-civ-spouse;Assoc-acdm;United-States;17-04-1962;160.75;Self-emp-not-
inc;Prof-specialty;<=50K
Male;31;White;Married-civ-spouse;9th;United-States;31-12-1978;172.10;Private;Machine-op-
inspct;<=50K
Male;53;White;Married-civ-spouse;Bachelors;United-States;22-05-1957;189.74;Self-emp-not-
inc;Prof-specialty;<=50K
Male;24;White;Married-civ-spouse;Bachelors;United-States;07-04-1986;170.08;Private;Tech-
support;<=50K
Female;49;White;Separated;HS-grad;United-States;18-04-1961;173.71;Private;Adm-clerical;<=50K
Male;25;White;Never-married;HS-grad;United-States;06-03-1985;160.52;Private;Handlers-
cleaners;<=50K
Male;57;Black;Married-civ-spouse;Bachelors;United-States;26-07-1953;178.12;Federal-gov;Prof-
specialty;>50K
Male;53;White;Married-civ-spouse;HS-grad;United-States;22-05-1957;186.11;Private;Machine-op-
inspct;<=50K
Female;44;White;Divorced;Masters;United-States;13-02-1966;162.80;Private;Exec-managerial;<=50K
Male;41;White;Married-civ-spouse;Assoc-voc;United-States;10-01-1969;172.39;State-gov;Craft-
repair;<=50K
Male;29;White;Never-married;Assoc-voc;United-States;02-02-1981;168.83;Private;Prof-
specialty;<=50K
Female;25;Other;Married-civ-spouse;Some-college;United-States;06-03-1985;179.12;Private;Exec-
managerial;<=50K
Female;47;White;Married-civ-spouse;Prof-school;Honduras;16-03-1963;163.02;Private;Prof-
specialty;>50K

```

```
Male;50;White;Divorced;Bachelors;United-States;19-04-1960;172.18;Federal-gov;Exec-
managerial;>50K
```

6.10 Sample Requests for Protegrity Anonymization REST API

Modify and use the sample requests provided here for anonymizing your data set. Use these requests as a template or as a guideline for building the required request.

6.10.1 Tree-based Aggregation for Attributes with k-Anonymity

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 1 Quasi Identifier
- **Suppression:** 0.01
- **Privacy Model:** K-Anonymity with k value as 50

Note: In this example, the data has custom delimiters.

```
{
  "source": {
    "type": "File",
    "file": {
      "name": "samples/adult.csv",
      "props": {
        "sep": ";"
      }
    }
  },
  "attributes": [
    {
      "name": "age",
      "dataType": "String",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Generalization",
      "generalization": {
        "type": "Masking Based",
        "hierarchyType": "Rule",
        "rule": {
          "masking": {
            "maskOrder": "Right To Left",
            "maskChar": "*",
            "maxDomainSize": 2
          }
        }
      }
    }
  ],
  "privacyModel": {
    "k": {
      "kValue": 50
    }
  },
  "config": {
    "maxSuppression": 0.01
  },
  "target": {
    "type": "File",
    "file": {
      "name": "s3://<Your-S3-BucketName>/anon-adult-e1.csv",
      "props": {
```



```

        "line_terminator": "\n"
      },
      "accessOptions": {
        "key": "<Your-S3-API Key>",
        "secret": "<Your-S3-API Secret>"
      }
    }
  }
}

```

6.10.2 Tree-based Aggregation for Attributes with k-Anonymity, l-Diversity, and t-Closeness

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 4 Quasi Identifiers, 2 Sensitive Attributes
- **Suppression:** 0.10
- **Privacy Model:** K with value 3, T-closeness with value 0.2, and L-diversity with value 2

Note: In this example, for an attribute, the generalization hierarchy is a part of the request.

```

{
  "source": {
    "type": "File",
    "file": {
      "name": "samples/adult.csv",
      "props": {
        "sep": ";",
        "decimal": ".",
        "quotechar": "\"",
        "escapechar": "\\",
        "encoding": "utf-8"
      }
    }
  },
  "attributes": [
    {
      "name": "marital-status",
      "dataType": "String",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Generalization",
      "generalization": {
        "type": "Tree Based",
        "hierarchyType": "Data Store",
        "dataStore": {
          "type": "File",
          "format": "CSV",
          "file": {
            "name": "samples/hierarchy/adult_hierarchy_marital-status.csv",
            "props": {
              "delimiter": ";",
              "quotechar": "\"",
              "header": null
            }
          }
        }
      }
    },
    {
      "name": "native-country",
      "dataType": "String",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Generalization",
      "generalization": {

```

```

        "type": "Tree Based",
        "hierarchyType": "Data Store",
        "dataStore": {
            "type": "File",
            "format": "CSV",
            "file": {
                "name": "samples/hierarchy/adult_hierarchy_native-country.csv",
                "props": {
                    "delimiter": ";",
                    "quotechar": "\"",
                    "header": null
                }
            }
        }
    },
    {
        "name": "occupation",
        "dataType": "String",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
            "type": "Tree Based",
            "hierarchyType": "Data Store",
            "dataStore": {
                "type": "File",
                "format": "CSV",
                "file": {
                    "name": "samples/hierarchy/adult_hierarchy_occupation.csv",
                    "props": {
                        "delimiter": ";",
                        "quotechar": "\"",
                        "header": null
                    }
                }
            }
        }
    },
    {
        "name": "race",
        "dataType": "String",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
            "type": "Tree Based",
            "hierarchyType": "Data",
            "data": {
                "hierarchy": [
                    [
                        "White",
                        "*"
                    ],
                    [
                        "Asian-Pac-Islander",
                        "*"
                    ],
                    [
                        "Amer-Indian-Eskimo",
                        "*"
                    ],
                    [
                        "Black",
                        "*"
                    ]
                ],
                "defaultHierarchy": [
                    "Other",
                    "*"
                ]
            }
        }
    },
    {

```

```

        "name": "sex",
        "dataType": "String",
        "classificationType": "Sensitive Attribute"
      },
      {
        "name": "salary-class",
        "dataType": "String",
        "classificationType": "Sensitive Attribute"
      }
    ],
    "config": {
      "maxSuppression": 0.10
    },
    "privacyModel": {
      "k": {
        "kValue": 3
      },
      "tcloseness": [
        {
          "name": "salary-class",
          "emdType": "EMD with equal ground distance",
          "tFactor": 0.2
        }
      ],
      "ldiversity": [
        {
          "name": "sex",
          "lFactor": 2,
          "lType": "Distinct-l-diversity"
        }
      ]
    },
    "target": {
      "type": "File",
      "file": {
        "name": "s3://<Your-S3-BucketName>/anon-adult_klt.csv",
        "props": {
          "line_terminator": "\n"
        },
        "accessOptions": {
          "key": "<Your-S3-API Key>",
          "secret": "<Your-S3-API Secret>"
        }
      }
    }
  }
}

```

6.10.3 Micro-Aggregation and Generalization with Aggregates

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 2 Quasi Identifiers, 1 Aggregation-based Quasi Identifier, 2 Micro Aggregations, and 2 Sensitive Attributes
- **Suppression:** 0.50
- **Privacy Model:** K with value 5, T-closeness with value 0.2, and L-diversity with value 2

```

{
  "source": {
    "type": "File",
    "file": {
      "name": "samples/adult.csv",
      "props": {
        "sep": ";"
      }
    }
  },
  "attributes": [

```

```

{
  "name": "age",
  "dataType": "Integer",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Micro Aggregation",
  "aggregateFn": "GMean"
},
{
  "name": "marital-status",
  "dataType": "String",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Micro Aggregation",
  "aggregateFn": "Mode"
},
{
  "name": "native-country",
  "dataType": "String",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "generalization": {
    "type": "Tree Based",
    "hierarchyType": "Data Store",
    "dataStore": {
      "type": "File",
      "format": "CSV",
      "file": {
        "name": "samples/hierarchy/adult_hierarchy_native-country.csv",
        "props": {
          "delimiter": ";",
          "quotechar": "\"",
          "header": null
        }
      }
    }
  }
},
{
  "name": "occupation",
  "dataType": "String",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "generalization": {
    "type": "Tree Based",
    "hierarchyType": "Data Store",
    "dataStore": {
      "type": "File",
      "format": "CSV",
      "file": {
        "name": "samples/hierarchy/adult_hierarchy_occupation.csv",
        "props": {
          "delimiter": ";",
          "quotechar": "\"",
          "header": null
        }
      }
    }
  }
},
{
  "name": "race",
  "dataType": "String",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "generalization": {
    "type": "Aggregation Based",
    "hierarchyType": "Aggregate",
    "aggregateFn": "Mode"
  }
},
{
  "name": "sex",
  "classificationType": "Sensitive Attribute",
  "dataType": "String"
}

```

```

    },
    {
      "name": "salary-class",
      "classificationType": "Sensitive Attribute",
      "dataType": "String"
    }
  ],
  "config": {
    "maxSuppression": 0.50
  },
  "privacyModel": {
    "k": {
      "kValue": 5
    },
    "tcloseness": [
      {
        "name": "salary-class",
        "emdType": "EMD with equal ground distance",
        "tFactor": 0.2
      }
    ],
    "ldiversity": [
      {
        "name": "sex",
        "lType": "Distinct-l-diversity",
        "lFactor": 2
      }
    ]
  },
  "target": {
    "type": "File",
    "file": {
      "name": "s3://<Your-S3-BucketName>/anon-adult_micro.csv",
      "props": {
        "line_terminator": "\n"
      },
      "accessOptions": {
        "key": "<Your-S3-API Key>",
        "secret": "<Your-S3-API Secret>"
      }
    }
  }
}

```

6.10.4 Parquet File Format

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket in the Parquet format
- **Data set:** 4 Quasi Identifiers, 1 Aggregation-based Quasi Identifier, 1 Micro Aggregation, and 1 Sensitive Attribute
- **Suppression:** 0.4
- **Privacy Model:** K with value 350 and L-diversity with value 2

Note: In this example, for an attribute, the generalization hierarchy is part of the request.

```

{
  "source": {
    "type": "File",
    "file": {
      "name": "samples/adult.csv",
      "props": {
        "sep": ";",
        "decimal": ",",
        "quotechar": "\"",
        "escapechar": "\\\"",
        "encoding": "utf-8"
      }
    }
  }
}

```

```

    },
    "attributes": [
      {
        "name": "age",
        "dataType": "Integer",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
          "hierarchyType": "Rule",
          "type": "Rounding",
          "rule": {
            "interval": {
              "levels": [
                "5",
                "10",
                "50",
                "100"
              ],
              "lowerBound": "5",
              "upperBound": "100"
            }
          }
        }
      },
      {
        "name": "marital-status",
        "dataType": "String",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Micro Aggregation",
        "aggregateFn": "Mode"
      },
      {
        "name": "citizenSince",
        "dataType": "Date",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
          "type": "Rounding",
          "hierarchyType": "Rule",
          "rule": {
            "daterange": {
              "levels": [
                "WD.M.Y",
                "FD.M.Y",
                "QTR.Y",
                "Y"
              ]
            }
          }
        },
        "props": {
          "dateformat": "dd-mm-yyyy"
        }
      },
      {
        "name": "occupation",
        "dataType": "String",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
          "type": "Tree Based",
          "hierarchyType": "Data Store",
          "dataStore": {
            "type": "File",
            "format": "CSV",
            "file": {
              "name": "samples/hierarchy/adult_hierarchy_occupation.csv",
              "props": {
                "delimiter": ";",
                "quotechar": "\"",
                "header": null
              }
            }
          }
        }
      }
    ]
  }

```

```

    }
  }
},
{
  "name": "race",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "dataType": "String",
  "generalization": {
    "type": "Aggregation Based",
    "hierarchyType": "Aggregate",
    "aggregateFn": "Mode"
  }
},
{
  "name": "salary-class",
  "dataType": "String",
  "classificationType": "Quasi Identifier",
  "dataTransformationType": "Generalization",
  "generalization": {
    "type": "Masking Based",
    "hierarchyType": "Rule",
    "rule": {
      "masking": {
        "maskOrder": "Left To Right",
        "maskChar": "*",
        "maxDomainSize": 3
      }
    }
  }
},
{
  "name": "sex",
  "dataType": "String",
  "classificationType": "Sensitive Attribute"
}
],
"config": {
  "maxSuppression": 0.4,
  "redactOutliers": true,
  "suppressionData": "Any"
},
"privacyModel": {
  "k": {
    "kValue": 350
  },
  "ldiversity": [
    {
      "name": "sex",
      "lType": "Distinct-l-diversity",
      "lFactor": 2
    }
  ]
},
"target": {
  "type": "File",
  "file": {
    "name": "s3://<Your-S3-BucketName>/anon-adult-rules",
    "format": "Parquet",
    "accessOptions": {
      "key": "<Your-S3-API Key>",
      "secret": "<Your-S3-API Secret>"
    }
  }
}
}

```

6.10.5 Retaining and Redacting

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket in the Parquet format
- **Data set:** 2 Quasi Identifiers, 1 Aggregation-based Quasi Identifier, 1 Micro Aggregation, 1 Non-Sensitive Attribute, 1 Identifying Attribute, and 2 Sensitive Attributes
- **Suppression:** 0.10
- **Privacy Model:** K with value 200 and L-diversity with value 2

Note: In this example, for an attribute, the generalization hierarchy is part of the request.

```
{
  "source": {
    "type": "File",
    "file": {
      "name": "samples/adult.csv",
      "props": {
        "sep": ";",
        "decimal": ",",
        "quotechar": "\"",
        "escapechar": "\\\"",
        "encoding": "utf-8"
      }
    }
  },
  "attributes": [
    {
      "name": "age",
      "dataType": "Integer",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Generalization",
      "generalization": {
        "type": "Rounding",
        "hierarchyType": "Rule",
        "rule": {
          "interval": {
            "levels": [
              "5",
              "10",
              "50",
              "100"
            ]
          }
        }
      }
    },
    {
      "name": "marital-status",
      "dataType": "String",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Micro Aggregation",
      "aggregateFn": "Mode"
    },
    {
      "name": "occupation",
      "dataType": "String",
      "classificationType": "Quasi Identifier",
      "dataTransformationType": "Generalization",
      "generalization": {
        "type": "Tree Based",
        "hierarchyType": "Data Store",
        "dataStore": {
          "type": "File",
          "format": "CSV",
          "file": {
            "name": "samples/hierarchy/adult_hierarchy_occupation.csv",

```



```

        "props": {
            "delimiter": ";",
            "quotechar": "\"",
            "header": null
        }
    },
    {
        "name": "race",
        "dataType": "String",
        "classificationType": "Quasi Identifier",
        "dataTransformationType": "Generalization",
        "generalization": {
            "type": "Aggregation Based",
            "hierarchyType": "Aggregate",
            "aggregateFn": "Mode"
        }
    },
    {
        "name": "citizenSince",
        "dataType": "Date",
        "classificationType": "Identifying Attribute"
    },
    {
        "name": "education",
        "dataType": "String",
        "classificationType": "Non-Sensitive Attribute"
    },
    {
        "name": "salary-class",
        "dataType": "String",
        "classificationType": "Sensitive Attribute"
    },
    {
        "name": "sex",
        "dataType": "String",
        "classificationType": "Sensitive Attribute"
    }
],
"config": {
    "maxSuppression": 0.10,
    "suppressionData": "Any"
},
"privacyModel": {
    "k": {
        "kValue": 200
    },
    "ldiversity": [
        {
            "name": "sex",
            "lType": "Distinct-l-diversity",
            "lFactor": 2
        },
        {
            "name": "salary-class",
            "lType": "Distinct-l-diversity",
            "lFactor": 2
        }
    ]
},
"target": {
    "type": "File",
    "file": {
        "name": "s3://<Your-S3-BucketName>/anon-adult_retd",
        "format": "Parquet",
        "accessOptions": {
            "key": "<Your-S3-API Key>",
            "secret": "<Your-S3-API Secret>"
        }
    }
}

```

```
}
}
```

6.11 Sample Requests for Protegrity Anonymization SDK

Modify and use the sample requests provided here for anonymizing your data set. Use these requests as a template or as a guideline for building the required request.

6.11.1 Tree-based Aggregation for Attributes with k-Anonymity

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 1 Quasi Identifier
- **Suppression:** 0.01
- **Privacy Model:** K-Anonymity with k value as 50

Note: In this example, the data has custom delimiters.

```
#import the anonsdk library
import anonsdk as asdk
import pandas as pd
import dask as df

# s3 bucket credentials
s3_key = <AWS_Key>
s3_secret = <AWS_Secret>

#set the source path for anonymization
# dataset path
source_csv_path = "D://adult.csv"
# create Store Object source_datastore
source_datastore = asdk.FileDataStore(source_csv_path)

#Set the target path for anonymized result
# anonymized file path
target_csv_path = "s3://target/anon-adult-e1.csv"
# create Store Object target_datastore
target_datastore = asdk.FileDataStore(target_csv_path, access_options={"key": s3_key,"secret":
s3_secret})

# Create connection Object with Rest API server
conn = asdk.Connection("https://anon.protegrity.com/")
df = dask.dataframe.read_csv(source_csv_path,sep=";")
df.head()

# create AnonObject with connection, dataframe metadata and source path
anon_object = asdk.AnonElement(conn, df.compute(), source_datastore)
# configure masking of string datatype
anon_object["age"] = asdk.Gen_Mask(maskchar="*",maskOrder="R",maxLength=2)

#Configure K-anonymity , suppression in the dataset allowed
anon_object.config.k = asdk.K(50)
anon_object.config['maxSuppression'] = 0.01

# Send Anonymization request with Transformation Configuration with the target store
job = asdk.anonymize(anon_object,target_datastore ,force=True)

# check the status of the job <check the status iteratively until 'status': 'Completed' >
job.status()
```

```
# check the comparative risk statistics from the source and result dataset
job.riskStat()

# check the comparative utility statistics from the source and result dataset
job.utilityStat()
```

6.11.2 Tree-based Aggregation for Attributes with k-Anonymity, l-Diversity, and t-Closeness

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 4 Quasi Identifiers, 2 Sensitive Attributes
- **Suppression:** 0.10
- **Privacy Model:** K with value 3, T-closeness with value 0.2, and L-diversity with value 2

Note: In this example, for an attribute, the generalization hierarchy is a part of the request.

```
#import the anonsdk library
import anonsdk as asdk
import pandas as pd
import dask as df

# s3 bucket credentials
s3_key = <AWS_Key>
s3_secret = <AWS_Secret>

#set the source path for anonymization
# dataset path
source_csv_path = "D://adult.csv"
# create Store Object source_datastore
source_datastore = asdk.FileDataStore(source_csv_path)

#Set the target path for anonymized result
# anonymized file path
target_csv_path = "s3://target/anon-adult_klt.csv"

# create Store Object target_datastore
target_datastore = asdk.FileDataStore(target_csv_path, access_options={"key": s3_key,"secret":
s3_secret})

# Create connection Object with Rest API server
conn = asdk.Connection("https://anon.protegrity.com/")

# create AnonObject with connection, dataframe metadata and source path
df = dask.dataframe.read_csv(source_csv_path,sep=";")
df.head()
anon_object = asdk.AnonElement(conn, df.compute(), source_datastore)

# configuration
hierarchy_marital_status_path = "samples\\hierarchy\\adult_hierarchy_marital-status.csv"
df_ms = dask.dataframe.read_csv(hierarchy_marital_status_path,sep=";").compute()
print(df_ms)
anon_object['marital-status']=asdk.Gen_Tree(df_ms)

hierarchy_native_country_path = "samples\\hierarchy\\adult_hierarchy_native-country.csv"
df_nc = dask.dataframe.read_csv(hierarchy_native_country_path,sep=";").compute()
print(df_nc)
anon_object['nativecountry']=asdk.Gen_Tree(df_nc)

hierarchy_occupation_path = "hierarchy\\adult_hierarchy_occupation.csv"
df_occ = dask.dataframe.read_csv(hierarchy_occupation_path).compute()
print(df_occ)
anon_object['occupation']=asdk.Gen_Tree(df_occ)

df_race = pd.DataFrame(data={"lvl0":["White","Asian-Pac-Islander","Amer-
```

```

Indian","Black","Other"], "lvl1":["*","*","*","*","*"]})
anon_object['race']=asdk.Gen_Tree(df_race)

#Configure K-anonymity , suppression allowed in the dataset
anon_object.config.k = asdk.K(3)
anon_object.config['maxSuppression'] = 0.10

#Configure L-diversity and T-closeness
anon_object["sex"]=asdk.LDiv(lfactor=2)
anon_object["salary-class"]=asdk.TCclose(tfactor=0.2)

# Send Anonymization request with Transformation Configuration with the target store
job = asdk.anonymize(anon_object,target_datastore ,force=True)

# check the status of the job
job.status()

# check the comparative risk statistics from the source and result dataset
job.riskStat()

# check the comparative utility statistics from the source and result dataset
job.utilityStat()

```

6.11.3 Micro-Aggregation and Generalization with Aggregates

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 2 Quasi Identifiers, 1 Aggregation-based Quasi Identifier, 2 Micro Aggregations, and 2 Sensitive Attributes
- **Suppression:** 0.50
- **Privacy Model:** K with value 5, T-closeness with value 0.2, and L-diversity with value 2

```

#import the anonsdk library
import anonsdk as asdk
import pandas as pd
import dask as df

# s3 bucket credentials
s3_key = <AWS_Key>
s3_secret = <AWS_Secret>

#set the source path for anonymization
# dataset path
source_csv_path = "D://adult.csv"
# create Store Object source_datastore
source_datastore = asdk.FileDataStore(source_csv_path)

#Set the target path for anonymized result
# anonymized file path
target_csv_path = "s3://target/anon-adult_micro.csv"
# create Store Object target_datastore
target_datastore = asdk.FileDataStore(target_csv_path, access_options={"key": s3_key,"secret":
s3_secret})

# Create connection Object with Rest API server
conn = asdk.Connection("https://anon.protegrity.com/")
df = dask.dataframe.read_csv(source_csv_path,sep=";")
df.head()

# create AnonObject with connection, dataframe metadata and source path
anon_object = asdk.AnonElement(conn, df.compute(), source_datastore)

# configuration
hierarchy_native_country_path = "hierarchy\\adult_hierarchy_native-country.csv"
df_nc = dask.dataframe.read_csv(hierarchy_native_country_path,sep=";").compute()
print(df_nc)

```

```
anon_object['nativecountry']=asdk.Gen_Tree(df_nc)

hierarchy_occupation_path = "samples\\hierarchy\\adult_hierarchy_occupation.csv"
df_occ = dask.dataframe.read_csv(hierarchy_occupation_path).compute()
print(df_occ)
anon_object['marital-status']=asdk.Gen_Tree(df_occ)

# applying aggregation rules
anon_object['age']=asdk.MicroAgg(asdk.AggregateFunction.GMean)
anon_object['race']=asdk.Gen_Agg(asdk.AggregateFunction.Mode)

# applying micro-aggregation rule
anon_object['marital-status']=asdk.MicroAgg(asdk.AggregateFunction.Mode)

#Configure K-anonymity , suppression in the dataset allowed
anon_object.config.k = asdk.K(5)
anon_object.config['maxSuppression'] = 0.50

#Configure L-diversity and T-closeness
anon_object["sex"]=asdk.LDiv(lfactor=2)
anon_object["salary-class"]=asdk.TCclose(tfactor=0.2)

# Send Anonymization request with Transformation Configuration with the target store
job = asdk.anonymize(anon_object,target_datastore ,force=True)

# check the status of the job
job.status()

# check the comparative risk statistics from the source and result dataset
job.riskStat()

# check the comparative utility statistics from the source and result dataset
job.utilityStat()
```

6.11.4 Retaining and Redacting

This sample uses the following attributes:

- **Source:** Local file system
- **Target:** Amazon S3 bucket
- **Data set:** 2 Quasi Identifiers, 1 Aggregation-based Quasi Identifier, 1 Micro Aggregation, 1 Non-Sensitive Attribute, 1 Identifying Attribute, and 2 Sensitive Attributes
- **Suppression:** 0.10
- **Privacy Model:** K with value 200 and L-diversity with value 2

Note: In this example, for an attribute, the generalization hierarchy is part of the request.

```
#import the anonsdk library
import anonsdk as asdk
import pandas as pd
import dask as df

# s3 bucket credentials
s3_key = <AWS_Key>
s3_secret = <AWS_Secret>

#set the source path for anonymization
# dataset path
source_csv_path = "D://adult.csv"
# create Store Object source_datastore
source_datastore = asdk.FileDataStore(source_csv_path)

#Set the target path for anonymized result
# anonymized file path
target_csv_path = "s3://target/anon-adult_retd"
```

```
# create Store Object target_datastore
target_datastore = asdk.FileDataStore(target_csv_path, access_options={"key": s3_key,"secret":
s3_secret})

# Create connection Object with Rest API server
conn = asdk.Connection("https://anon.protegrity.com/")
df = dask.dataframe.read_csv(source_csv_path,sep=";")
df.head()

# create AnonObject with connection, dataframe metadata and source path
anon_object = asdk.AnonElement(conn, df.compute(), source_datastore)

# configuration
hierarchy_occupation_path = "samples\\hierarchy\\adult_hierarchy_occupation.csv"
df_occ = dask.dataframe.read_csv(hierarchy_occupation_path,sep=";").compute()
print(df_occ)
anon_object['marital-status']=asdk.Gen_Tree(df_occ)
anon_object['marital-status']=asdk.MicroAgg(asdk.AggregateFunction.Mode)
anon_object['race']=asdk.Gen_Agg(asdk.AggregateFunction.Mode)
anon_object['age']=asdk.Gen_Interval([5,10,50,100])
anon_object['citizenSince']=asdk.Preserve()
anon_object['education']=asdk.Preserve()
anon_object['salary-class']=asdk.Redact()
anon_object['sex']=asdk.Redact()

#Configure K-anonymity , suppression in the dataset allowed
anon_object.config.k = asdk.K(200)
anon_object.config['maxSuppression'] = 0.10

#Configure L-diversity
anon_object["sex"]=asdk.LDiv(lfactor=2)
anon_object["salary-class"]=asdk.LDiv(lfactor=2)

# Send Anonymization request with Transformation Configuration with the target store
job = asdk.anonymize(anon_object,target_datastore ,force=True)

# check the status of the job
job.status()

# check the comparative risk statistics from the source and result dataset
job.riskStat()

# check the comparative utility statistics from the source and result dataset
job.utilityStat()
```

6.12 Troubleshooting

This section provides problems that you might face while working with the Protegrity Anonymization API and the solutions or workarounds to resolve those problems.

6.12.1 Known Issues

A list of known issues with their solution or workaround, if applicable, are provided here. The steps provided to resolve the known issues ensure that your product does not throw errors or crash.

- **Error:** The order of the columns in the output file and input file may be different.

Observation:

The order of the columns in the output does not match the column order as per the source data set.

Cause:

The Protegrity Anonymization API processes the source data set and saves the data to the output file as and when the processing for a column is complete. Hence, to conserve memory and complete the processing faster, the column order in the output file might not match the column sequence of the source file.

- **Error:** The *RuntimeError: An attempt has been made to start a new process before the current process has finished its bootstrapping phase.* error is displayed when you run an anonymization job from the SDK.

Observation:

The error similar to the one displayed above appears when an anonymization request is raised using a scripting language, such as Python.

Cause:

The Protegrity Anonymization API SDK code can be run using an SDK tool, such as, a Jupyter Notebook. However, if the anonymization code is run using a scripting language, such as, Python, then the code must be enclosed and executed in the *main()* block.

Resolution:

In a Python script, calls to the Protegrity Anonymization API SDK code must be enclosed and executed in the *main()* block.

- **Error:** Ingress cannot process the request when the data frame attached to the anonymization job request is more than 150 MB.

Observation:

If large files over 150 MB are submitted for anonymization, then the anonymization job fails.

Resolution:

When the anonymization task fails to process large files, use a Cloud storage service, such as, Amazon S3 bucket, Google Cloud Storage, Azure Data Lake, or Azure Blob storage, for storing the source and destination file.

For more information about using external data sources, refer the to section [Identifying the Source and Target](#).

- **Error:** Plain text response to anonymization job contains HTTP URLs.

Observation:

When you submit an anonymization job and the response has URLs, then the URLs are returned in plain text even if the request was sent using the HTTPS protocol.

Resolution:

Manually Change the *http* links to use *https*.

- **Error:** In the Docker service, the API response URL appears as *http://app/*.

Observation:

In a Docker-based deployment, when you submit an anonymization job and the response has URLs, then the URLs are returned as *http://app/* instead of *http://anon.protegrity.com*.

Resolution:

Update the *nginx.conf* file with the following code so that the response URLs are correct.

```
upstream anon.protegrity.com
{
    server anon:8090;
}

location /
{
    proxy_set_header X-Forwarded-Host $host:$server_port;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://anon.protegrity.com;
}
```

- **Error:** *Mean_importance* and importance values return the value *NaN* in the logs.

Observation:

When the anonymization job is invoked it returns *NaN* for the mean importance and the feature importance in the logs.

Resolution:

Try running the anonymization job again.

- **Error:** The data types set detected by anonymization does not match the data that is present in the column.

Observation:

The data type specified in the column and the data type that the Auto Anonymizer detects is different. For example, the Auto Anonymizer detects date as phone number (phoneno) and credit card number as integer.

Resolution:

The Auto Anonymizer uses the internal intelligence to detect the data type for performing internal operations efficiently and hence the data type detected might not match the data in the column. However, the quality of the data is retained even though the data type is different.

- **Error:** Dataregex detection with ipv6 data failed

Observation:

When the Protegrity Anonymization API is configured on a system that uses an IPv6 address, then the IP address is not detected as the type *IP* and the request fails with the *INTERNAL SERVER ERROR* error.

Resolution:

Try updating the IP address of the Protegrity Anonymization API to use a non-IPv6 address.

- **Error:** All the hierarchy listed in the country and state hierarchy files are taking into consideration irrespective of the dataframe country and state fields

Observation:

The values present in the data is not used for generating the hierarchy values. However, the data values from the default list that is set for the hierarchy files references is used for detecting the hierarchy with the detectHierarchy API.

- **Error:** 502 Bad Gateway error when the message body is a string, empty, or an array

Observation:

When you send a post request for detecting the classification type API and the message body of the request is of the type string, empty, or array, then the response received is *502 Bad Gateway* instead of *400 Bad Request*.

6.12.2 Limitations

A list of limitations with their solution or workaround, if applicable, are provided here. The steps provided to resolve the limitations ensure that your product does not throw errors or crash.

- **Error:** The "info": "[('('shuffle-split-efb361971622d13bfecb17dd75515411', 1, 5, (15, 14))|', <Worker 'tls://<IP_Address>:42613', name: tls://<IP_Address>:42613, memory: 0, processing: 182>)]", "status": "Failed" error is displayed.

Observation:

When the anonymization job is run with the T-closeness privacy model set and the source file size is more than the total memory available for all workers in the cluster, then the execution fails.

Resolution:

The T-closeness privacy model requires adequate memory for processing files. To prevent the task from failing, ensure that the total memory available for all workers in the cluster is three times the size of the source file.

- **Error:** The Abort API does not abort the running job immediately.

Observation:

After aborting the task, it might take time before all the running processes are stopped.

- **Error:** Failed to edit the *max_Supression* parameter after an exception.

Observation:

When you run an anonymization job with the *max_Supression* value outside the range 0.1 to 0.99, such as the value 1.99, then an *Invalid Config* exception is raised. After this exception is raised, if you update the *max_Supression* to a valid value, then the exception still appears.

- **Error:** Memory utilization of the anon dask scheduler pod keeps on increasing

Observation:

When you create a cluster for running anonymization using Dask, then the memory usage in the Dask Scheduler keeps increases steadily over time even when no anonymization request is being processed. This appears like a memory leak issue in Dask, however, in a longevity test performed over the course of 5 days, the memory usage stabilizes after 3 days.

Resolution:

Restart the system when no anonymization request is being processed and the memory utilization is very high.

6.12.3 Working with Certificates

Use the commands provided in this section to work with and troubleshoot any certificate-related issues.

- Verify the certificate and view the certificate information.

```
openssl verify -verbose -CAfile cacert.pem server.crt
```

- Check a certificate and view information about the certificate, such as, signing authority, expiration date, and other certificate-related information.

```
openssl x509 -in server.crt -text -noout
```

- Check the SSL key and verify the key for consistency.

```
openssl rsa -in server.key -check
```

- Verify the CSR and view the CSR data that was entered when generating the certificate.

```
openssl req -text -noout -verify -in server.csr
```

- Verify that the certificate and corresponding key matches by displaying the md5 checksums of the certificate and key. The checksums can then be compared to verify that the certificate and key match.

```
openssl x509 -noout -modulus -in server.crt | openssl md5
openssl rsa -noout -modulus -in server.key | openssl md5
```