**😊 AWS ElastiCache**

Amazon ElastiCache is a managed caching service provided by Amazon Web Services (AWS). It helps improve the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases. ElastiCache supports two popular open-source in-memory caching engines: Redis and Memcached.

With ElastiCache, you can easily deploy, operate, and scale caching environments in the cloud. It simplifies the setup and maintenance of caching clusters, handling tasks such as hardware provisioning, software patching, setup, configuration, and backups.

ElastiCache can be particularly useful for use cases where you need to offload read-heavy workloads from your primary databases, reduce latency, or scale your application's capacity to handle more concurrent users.

Common use cases for ElastiCache include:

1. **Session Store:** Storing session data to improve web application performance.
2. **Database Caching:** Caching frequently accessed database query results.
3. **Content Caching:** Storing frequently accessed content (e.g., images, videos) to reduce latency.
4. **Leaderboards and Counting:** Counting and ranking operations, often seen in gaming and social networking applications.
5. **Real-time Analytics:** Storing and analyzing real-time data for faster processing.

**😁 Benefits of ElastiCache:**

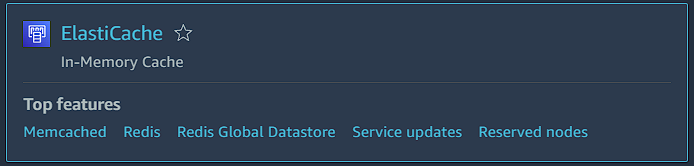
The benefits of using Amazon ElastiCache include:

1. **Improved Performance:** ElastiCache provides fast, in-memory caching solutions that can significantly reduce latency compared to fetching data from disk-based databases. This leads to better overall application performance and a smoother user experience.
2. **Scalability:** ElastiCache makes it easy to scale your caching infrastructure as your application grows. You can easily add or remove cache nodes to accommodate changes in traffic patterns and workload demands without impacting performance.
3. **Managed Service:** AWS manages the infrastructure and operational tasks associated with ElastiCache, such as hardware provisioning, software patching, setup, configuration, and backups. This allows you to focus on building and improving your applications rather than managing infrastructure.
4. **Cost-Effectiveness:** By offloading read-heavy workloads from your primary databases and reducing the need for expensive disk I/O operations, ElastiCache can help lower overall infrastructure costs while improving performance.
5. **High Availability:** ElastiCache supports replication and automatic failover for Redis clusters, ensuring high availability and fault tolerance. This helps minimize downtime and ensures that your applications remain responsive even in the event of node failures.
6. **Compatibility:** ElastiCache supports popular open-source caching engines like Redis and Memcached, making it easy to integrate with existing applications and frameworks.
7. **Security:** ElastiCache provides security features such as encryption at rest and in transit, IAM integration for access control, and VPC isolation to help secure your cache clusters and data.
8. **Monitoring and Insights:** AWS offers tools like Amazon CloudWatch and Amazon CloudTrail for monitoring and logging ElastiCache performance metrics and API activity. This allows you to gain insights into the health and performance of your caching infrastructure.

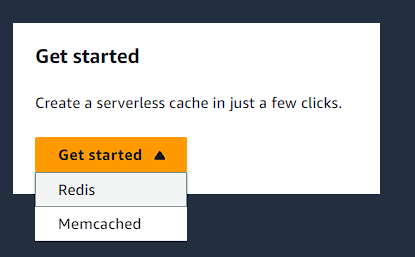
**In this exercise, we're setting up an Amazon ElastiCache Redis cluster and demonstrating how to interact with it using a Python script running on an EC2 instance. The end goal is to understand how to leverage ElastiCache to improve application performance by caching frequently accessed data in memory. The Python script showcases how to store and retrieve data from the cache, illustrating the simplicity and effectiveness of using ElastiCache for performance optimization. This exercise provides hands-on experience with AWS services and demonstrates a practical use case for caching in a cloud environment.**

**😁 To begin with the Lab:**

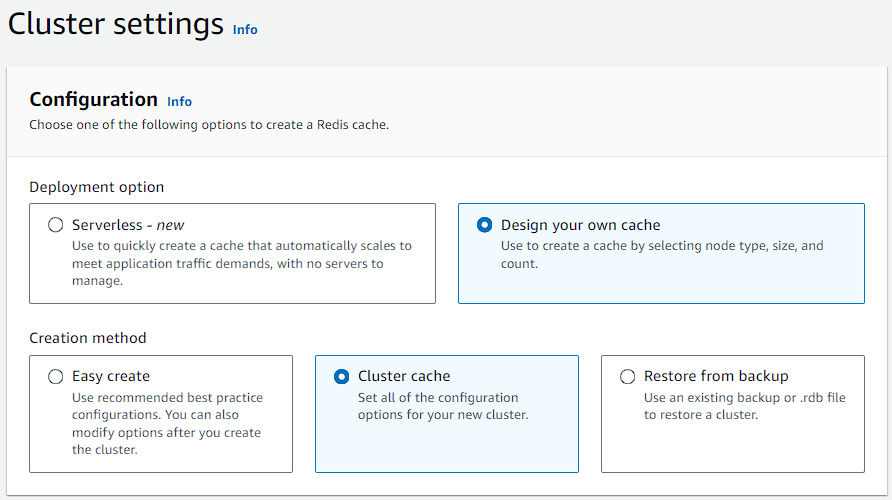
1. Log in to AWS Console and navigate to ElastiCache. Choose this service accordingly.



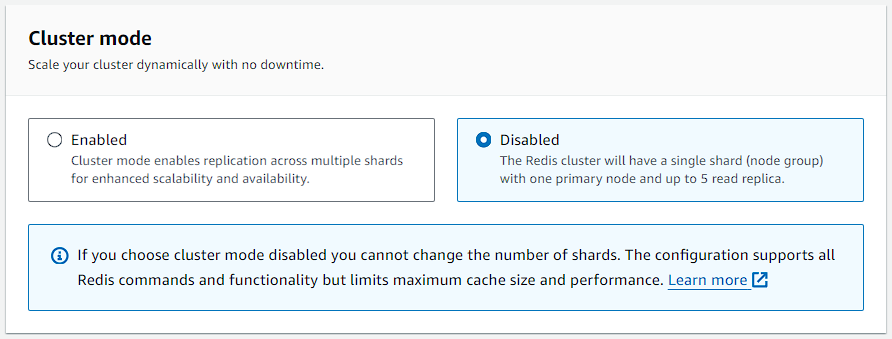
1. From its dashboard page you will see a get started option. And this has two more options in it. Here you have to choose Redis.



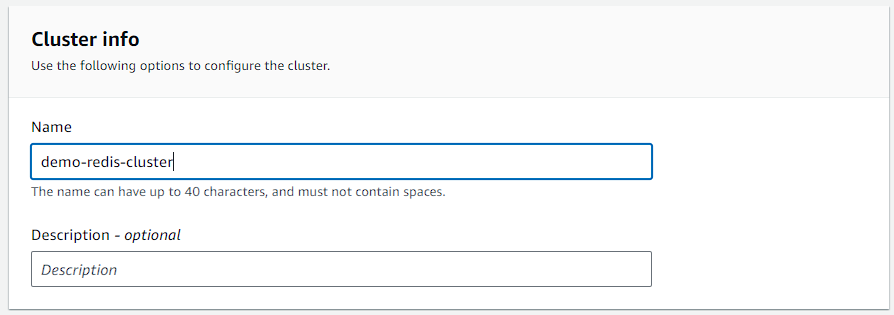
1. Then you have two options in the configuration, so you have to design your own cache for now.
2. After that you have to select cluster cache.



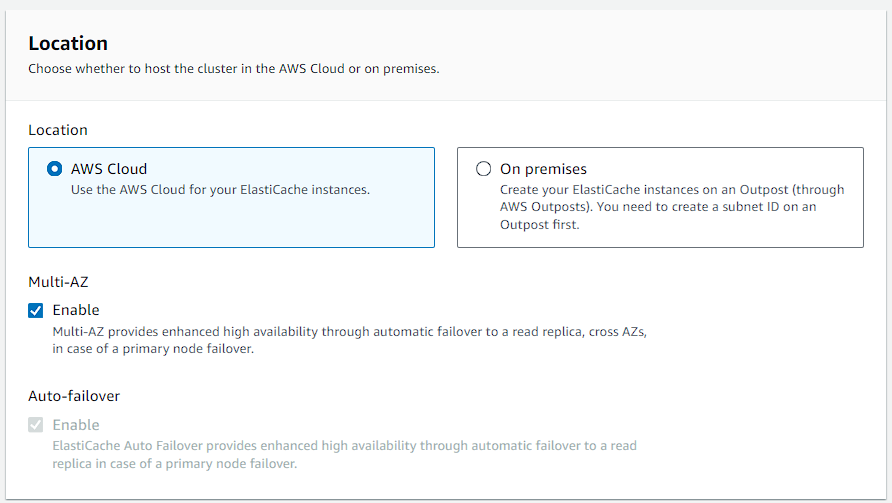
1. For the cluster mode keep it to default that is disabled.



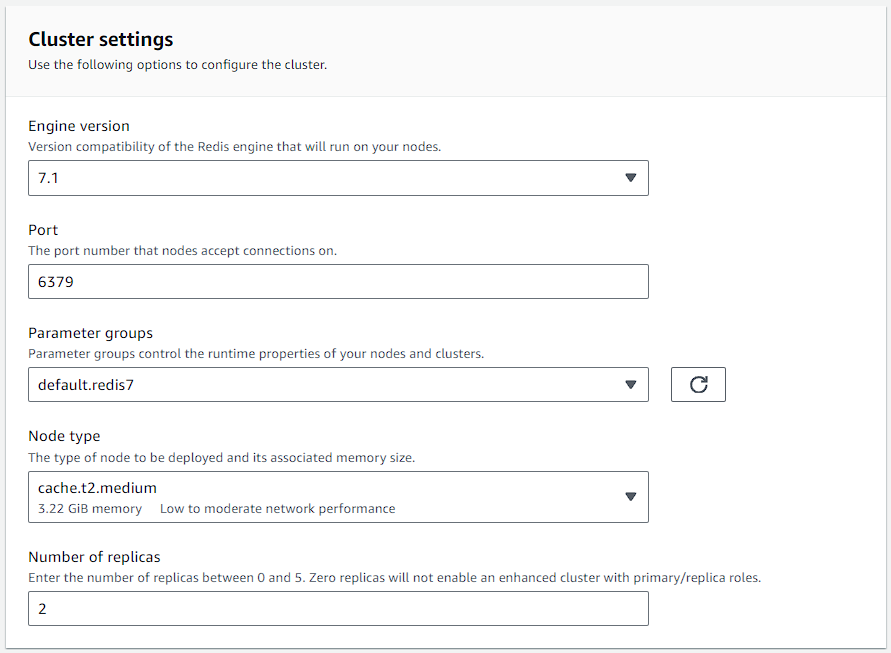
1. After that you are going to give it a name.



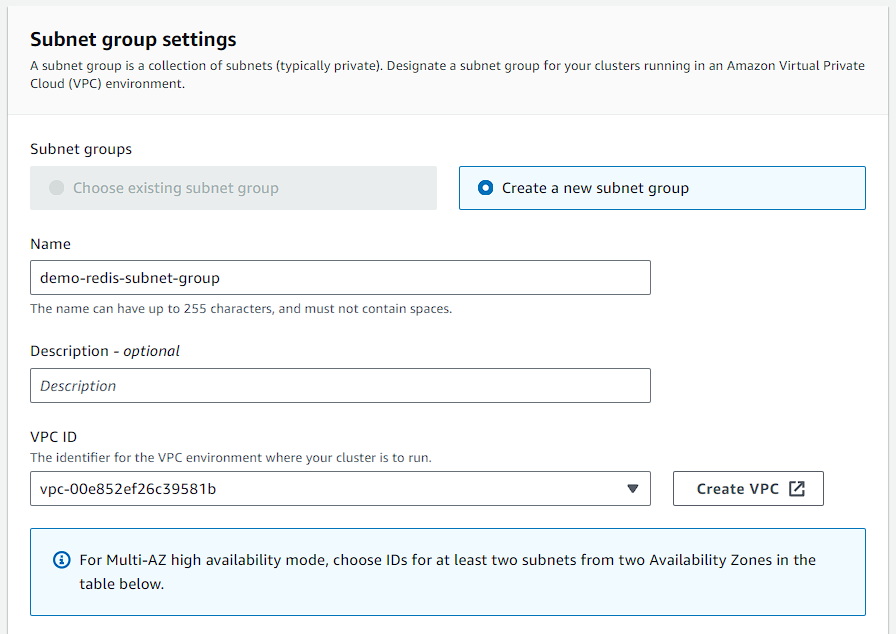
1. Now for the location choose AWS Cloud and turn on Muti-AZ.



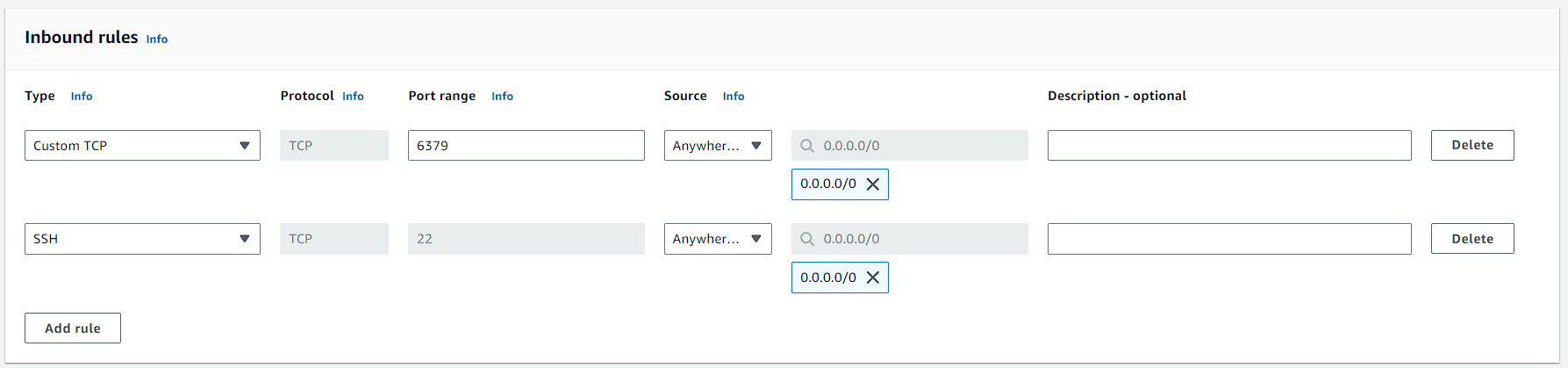
1. In the cluster settings select the latest engine version and the port number is default 6379 with the parameter groups.
2. Then in the node type select it to t2.medium or you can choose a smaller node too.
3. Now the number of replicas can be between 0 to 5. So, we have kept it to 2 which is default too.



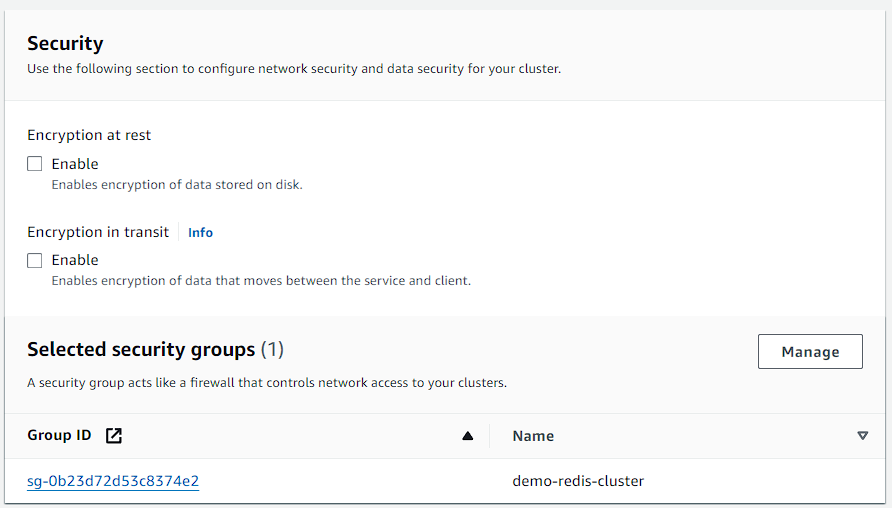
1. For the subnet group you have to create a new subnet group and choose your default VPC if you want. Then move to next page.



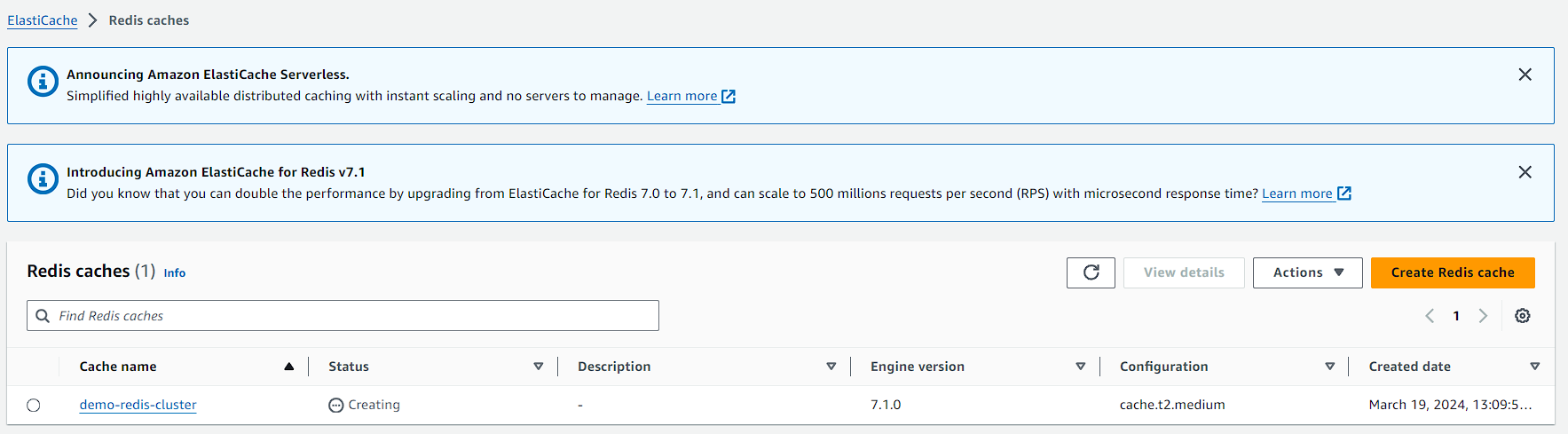
1. On the next you are going to add your security group but first, you have to create one. For that go to EC2 and create a new security group. Add these two rules in it, with that you are also going to use the same Security group in your EC2 instance too.
2. Once thing to remember make your security group in the same VPC as your cluster.



1. Now turn off both the encryptions and add your security group.



1. Now keep the rest of the settings as they are and create your cluster.
2. Below you can see that currently your cluster is creating.

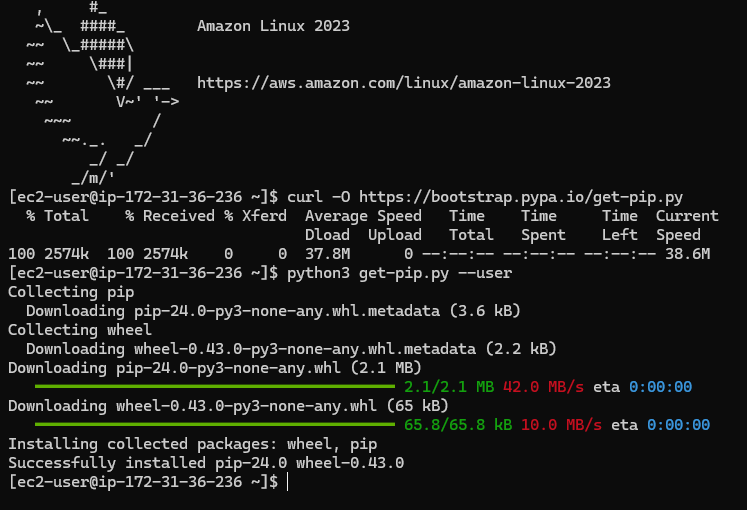


1. Now you should navigate to EC2 and launch an instance with the same security group.
2. Then you have to connect to it via SSH. After that, you have to do some commands.
3. These commands will download and install PIP, and then use PIP to install the Redis library. The --user flag is used to install packages locally for the current user, which is often a good practice to avoid conflicts with system packages. Once installed, you can use the Redis library in your Python scripts on the EC2 instance. Make sure to replace 'your-endpoint' with the actual endpoint of your ElastiCache cluster in your Python script.

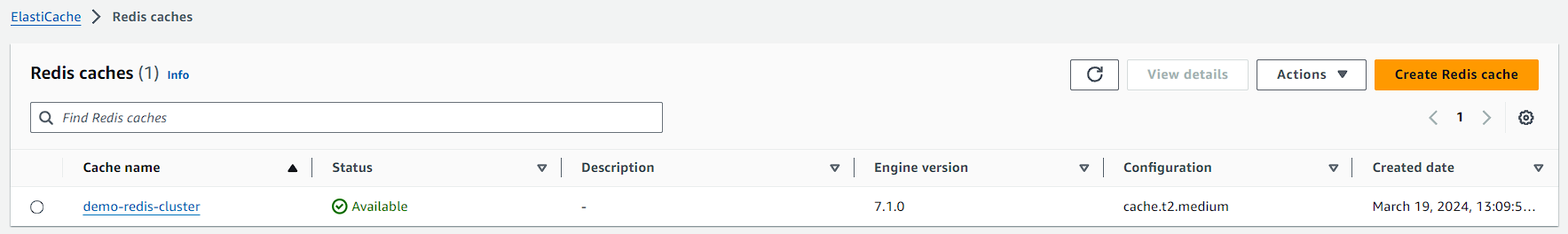
**curl -O https://bootstrap.pypa.io/get-pip.py**

**python3 get-pip.py --user**

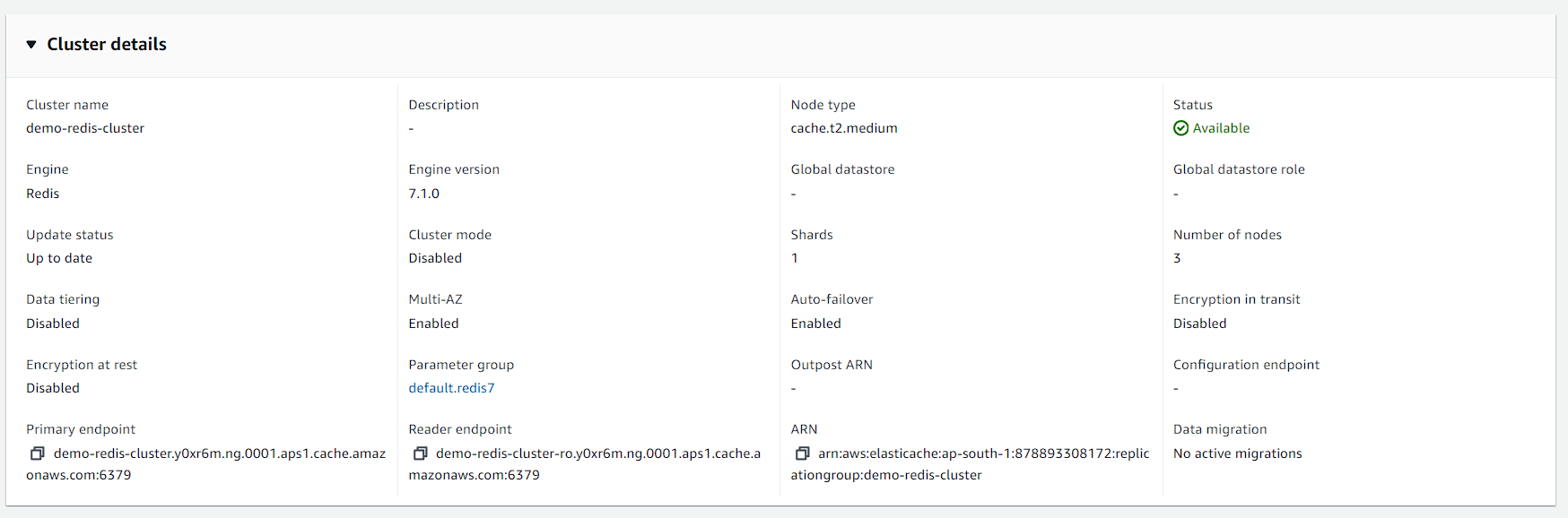
**pip install redis --user**



1. Now go back to your Redis cluster in ElastiCache and check for your cluster whether it is ready or not.



1. Now can open it and check its properties and every other thing. Then copy the primary endpoint.



1. Now you have to paste this endpoint into your code. Below is your code.
2. Run this command to create a file and save your code in it.

**nano python-script.py**



**import redis**

**# Replace 'your-endpoint' with the endpoint of your ElastiCache cluster, which you can find in the AWS console**

**cache = redis.StrictRedis(host='demo-redis-cluster.y0xr6m.ng.0001.aps1.cache.amazonaws.com', port=6379, db=0)**

**# Storing a value in the cache**

**cache.set('my-demo-redis-key', 'my-demo-redis-value')**

**# Retrieving the stored value from the cache**

**value = cache.get('my-demo-redis-key')**

**# Decoding the bytes literal to a string**

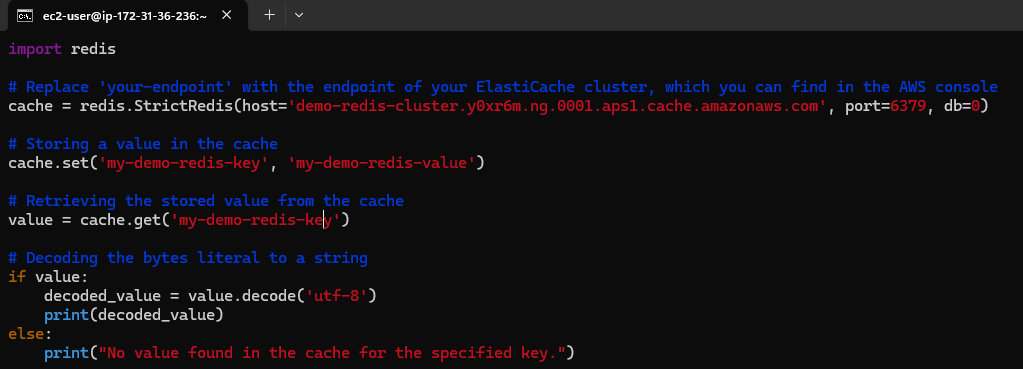
**if value:**

**decoded\_value = value.decode('utf-8')**

**print(decoded\_value)**

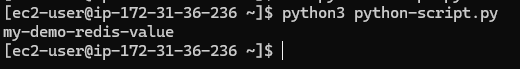
**else:**

**print("No value found in the cache for the specified key.")**



1. Now you have to run your python script.
2. So, you are getting the result accordingly.

**python3 python-script.py**



1. So, what it's done is in the script, it stores the value in the cache, and that is a key-value pair. "my-demo-redis-key” corresponds to the value "my-demo-redis-value". And then we retrieve the value from the cache and simply display it on the screen.
2. So, a simple exercise is just to show that we can store and retrieve values in our ElastiCache cluster.
3. Once you are done terminate your instance and delete your cluster.