

AMAZON ELASTICACHE – REDIS CACHE

ElastiCache:

ElastiCache is a web service offered by Amazon Web Services (AWS) that acts as a managed in-memory data store and cache. It essentially speeds up your web applications by storing frequently accessed data in-memory, which is much faster than traditional disk-based databases.

Here's a simpler way to think of it: ElastiCache takes the burden of managing your own cache infrastructure away. Instead of setting up and maintaining servers yourself, ElastiCache lets you focus on using the cache to improve your application's performance.

What is Redis?

Redis (REmote DIctionary Server) is an open source, in-memory, NoSQL key/value store that is used primarily as an application cache or quick-response database.

Redis stores data in memory, rather than on a disk or solid-state drive (SSD), which helps deliver unparalleled speed, reliability, and performance.

When an application relies on external data sources, the latency and throughput of those sources can create a performance bottleneck, especially as traffic increases or the application scales. One way to improve performance in these cases is to store and manipulate data in-memory, physically closer to the application. Redis is built to this task: It stores all data in-memory—delivering the fastest possible performance when reading or writing data—and offers built-in replication capabilities that let you place data physically closer to the user for the lowest latency.

Other Redis characteristics worth noting include support for multiple data structures, built-in Lua scripting, multiple levels of on-disk persistence, and high availability.

Here are some key benefits of using Redis cache:

- **Speed:** Since data resides in memory, retrieval times are significantly faster compared to disk-based databases. This can drastically improve the responsiveness of your application.
- **Reduced Database Load:** By caching frequently used data, Redis lessens the burden on your primary database, allowing it to focus on more complex tasks.
- **Scalability:** Redis can be easily scaled horizontally by adding more servers to your cache cluster. This ensures smooth performance as your application grows.

- **Data Structures:** Redis goes beyond simple key-value pairs. It supports various data structures like lists, sets, and sorted sets, enabling you to store and manipulate complex data efficiently.
- **Persistence (optional):** While primarily in-memory, Redis offers optional persistence mechanisms to save data to disk for data recovery in case of server restarts.

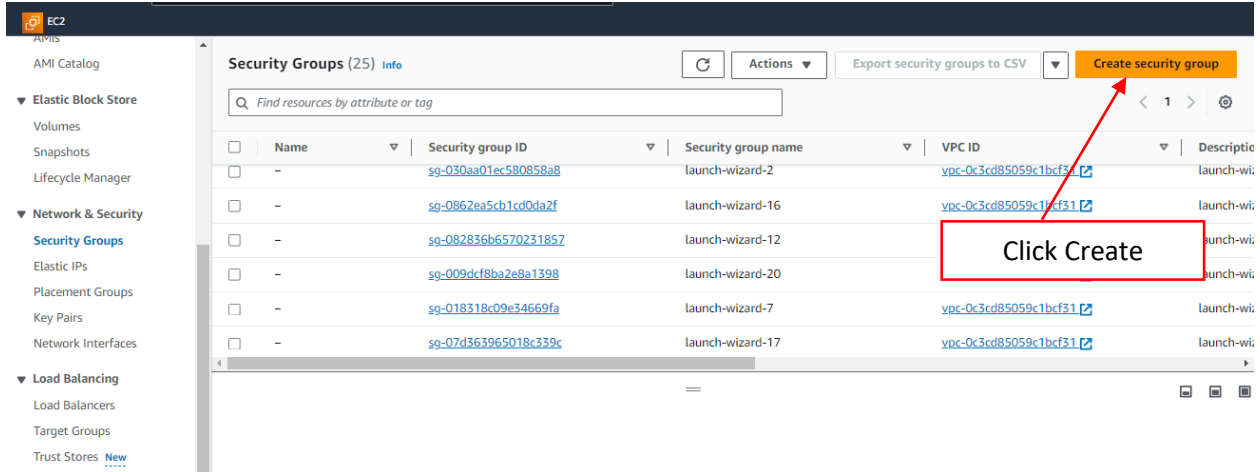
Here are some common use cases for Redis cache:

- **Application Caching:** Store frequently accessed data like product information, user profiles, or search results to minimize database load and improve response times.
- **Session Management:** Store user session data in Redis for faster retrieval and improved user experience.
- **Real-time Applications:** Power features like leaderboards, chat applications, or real-time analytics with low latency by caching frequently changing data.
- **Message Queues:** Utilize Redis lists or pub/sub functionality to implement message queues for communication between different parts of your application.

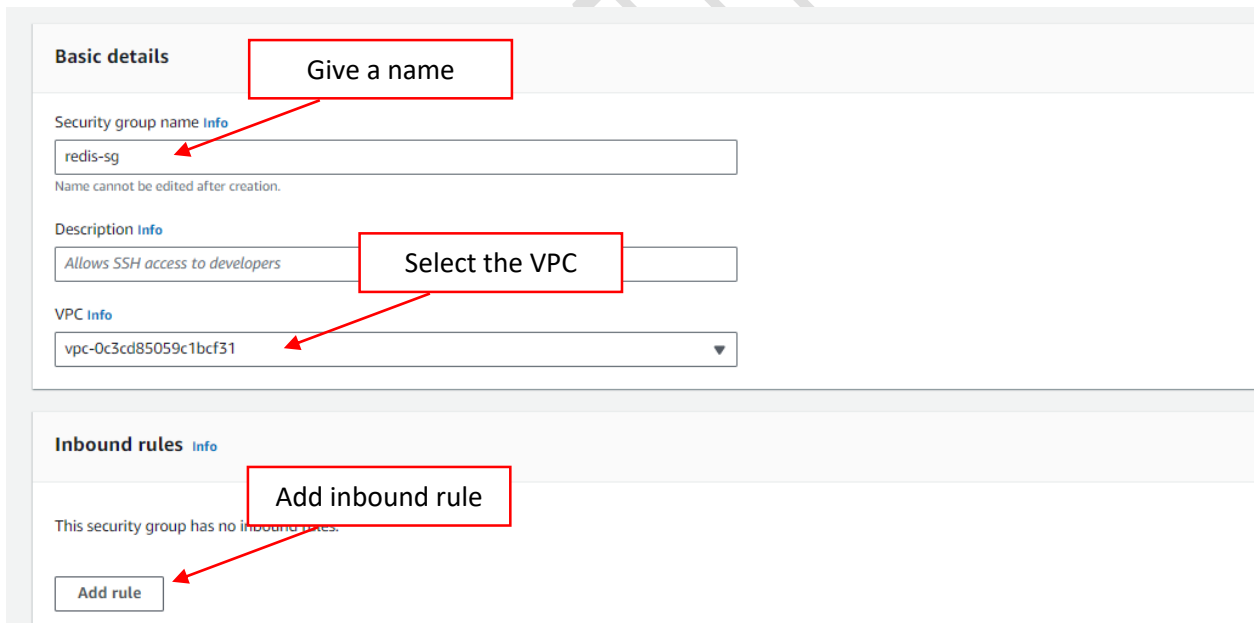
In essence, Redis cache acts as a high-speed layer between your application and the primary database, accelerating data access and enhancing the overall performance of your application.

STEPS TO CREATE A REDIS CACHE

Step 1 : Create a Security Group



The screenshot shows the AWS Management Console interface for the EC2 instance profile. The left sidebar contains navigation links for various services. The main content area displays the 'Security Groups (25)' page. A red box highlights the 'Create security group' button in the top right corner, with an arrow pointing to it and the text 'Click Create'.



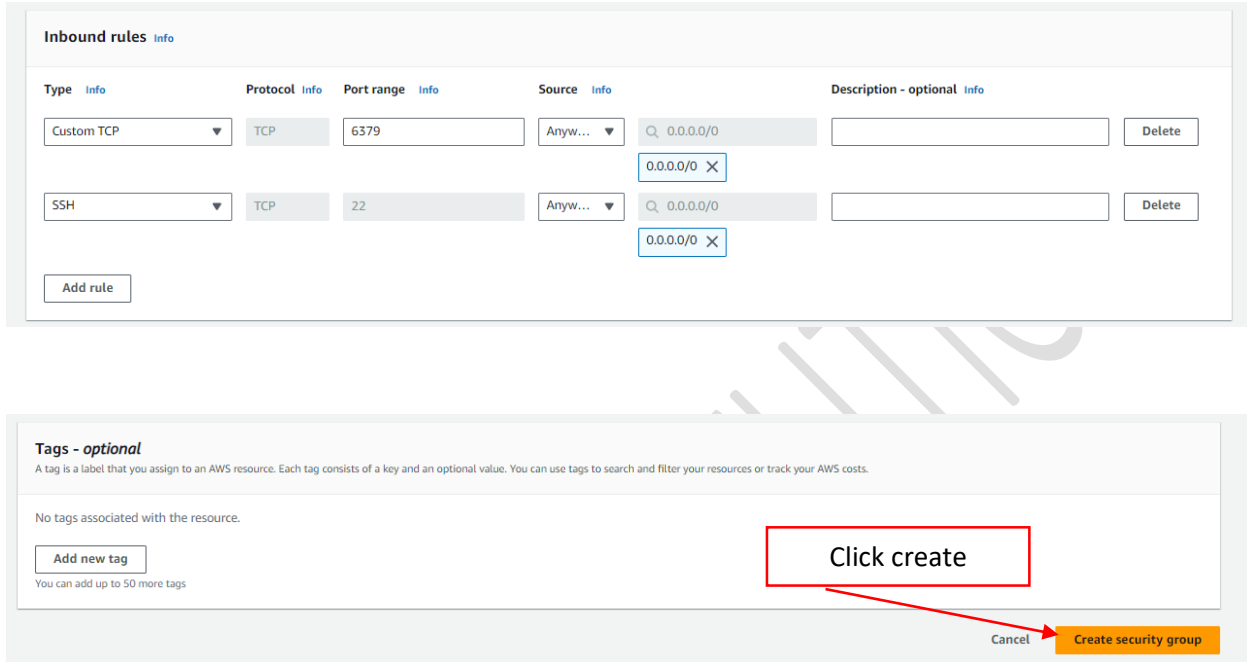
The screenshot shows the 'Create Security Group' wizard in the AWS Management Console. The 'Basic details' section is visible, with the following fields and annotations:

- Security group name:** redis-sg (Annotated with 'Give a name')
- Description:** Allows SSH access to developers
- VPC:** vpc-0c3cd85059c1bcf31 (Annotated with 'Select the VPC')

The 'Inbound rules' section is also visible, showing 'This security group has no inbound rules.' and an 'Add rule' button (Annotated with 'Add inbound rule').

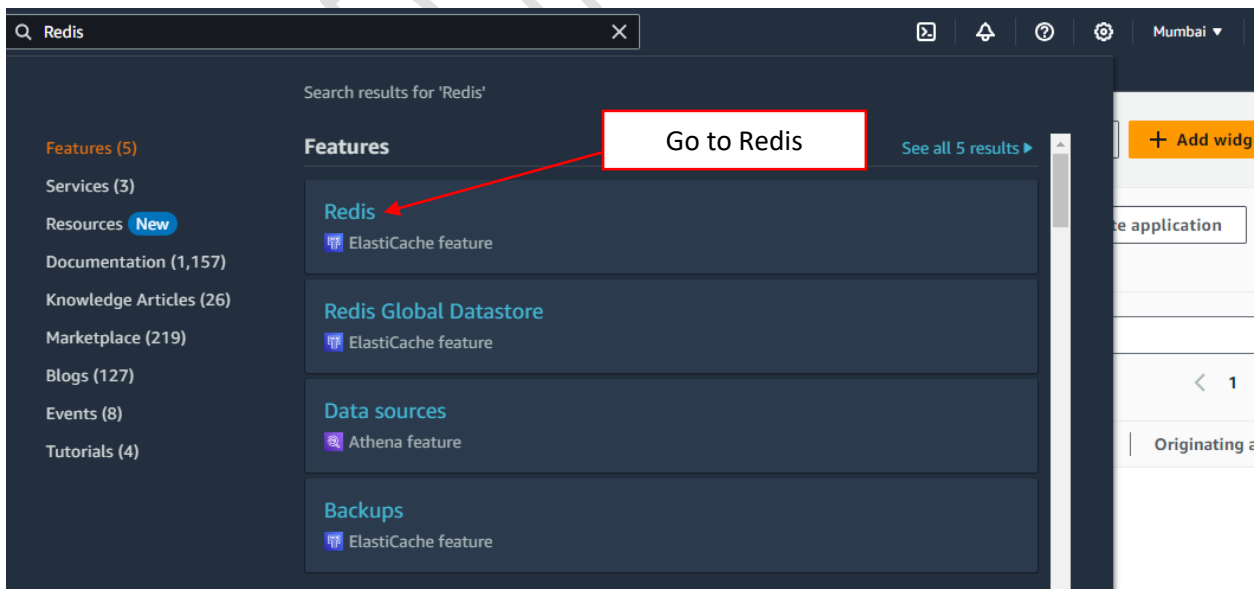
Add SSH, Custom TCP with port range 6379

TCP port 6379 is commonly used by Redis, an in-memory data structure store that is often used as a database, cache, and message broker. Redis uses this port for its default client-server communication protocol.

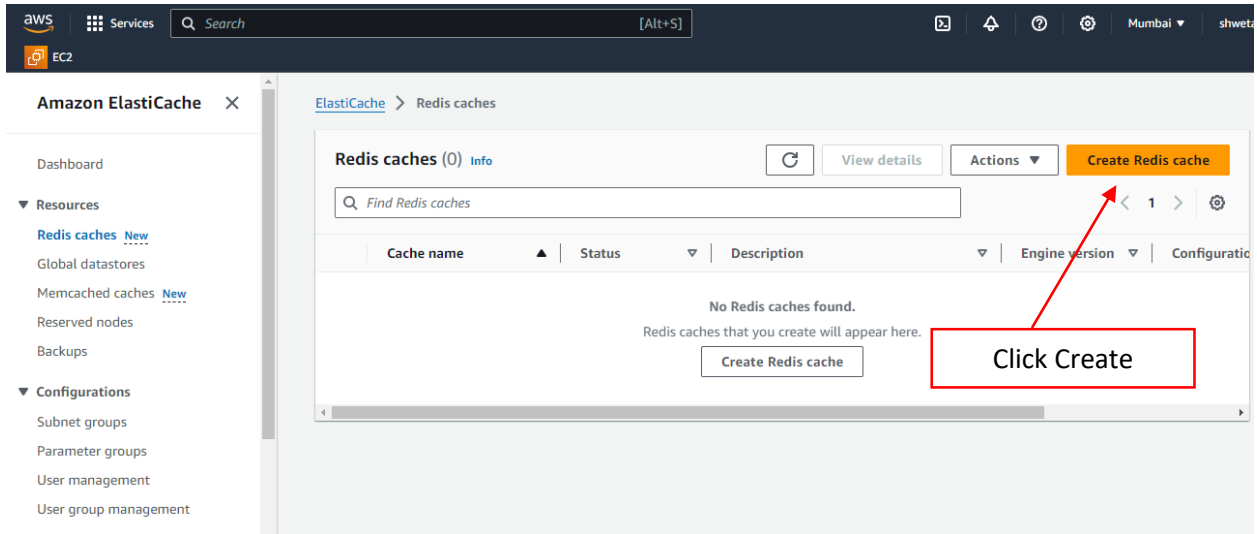


The screenshot shows the AWS IAM console interface for configuring a security group. The 'Inbound rules' section is visible, showing two rules: 'Custom TCP' with port range 6379 and 'SSH' with port range 22. The 'Tags' section is also visible, showing 'No tags associated with the resource.' and a 'Click create' button. A red arrow points from the 'Click create' button to the 'Create security group' button at the bottom right.

Step 2 : Create a Cache



The screenshot shows the AWS IAM console search results for 'Redis'. The search results are displayed in a list format. The first result is 'Redis', which is highlighted with a red box and a red arrow pointing to it. The 'Go to Redis' button is also visible. The search results are categorized by 'Features' and 'Resources'.



Amazon ElastiCache

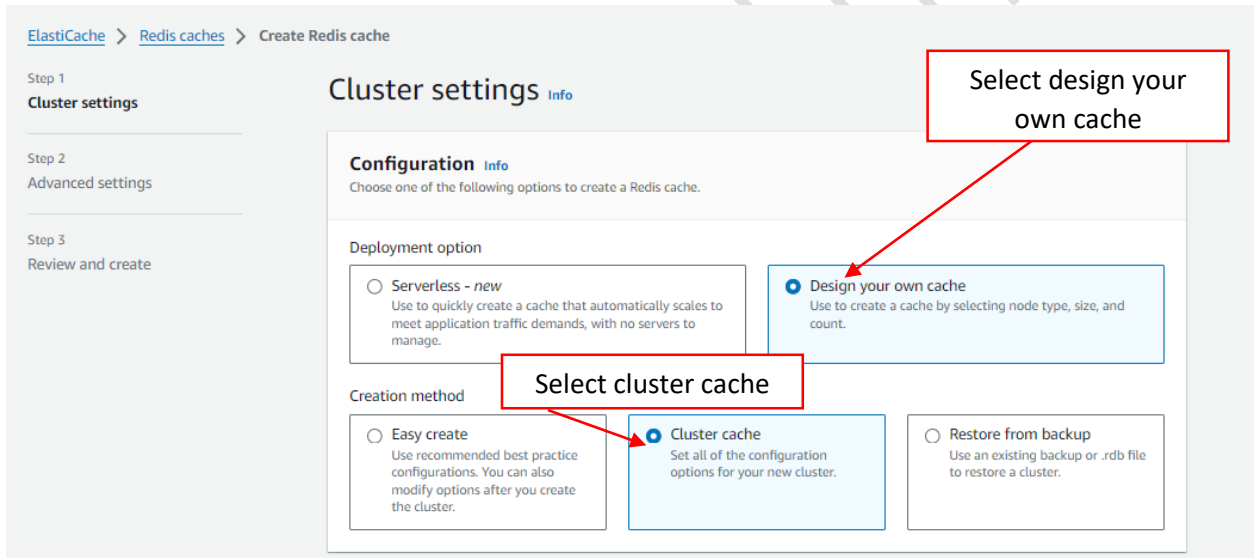
Redis caches (0) Info

Find Redis caches

No Redis caches found.
Redis caches that you create will appear here.

Create Redis cache

Click Create



ElastiCache > Redis caches > Create Redis cache

Step 1
Cluster settings

Step 2
Advanced settings

Step 3
Review and create

Cluster settings Info

Configuration Info
Choose one of the following options to create a Redis cache.

Deployment option

☐ Serverless - new
Use to quickly create a cache that automatically scales to meet application traffic demands, with no servers to manage.

☒ Design your own cache
Use to create a cache by selecting node type, size, and count.

Creation method

☐ Easy create
Use recommended best practice configurations. You can also modify options after you create the cluster.

☒ Cluster cache
Set all of the configuration options for your new cluster.

☐ Restore from backup
Use an existing backup or .rdb file to restore a cluster.

Select design your own cache

Select cluster cache

Cluster mode

Scale your cluster dynamically with no downtime.

☐ Enabled

Cluster mode enables replication across multiple shards for enhanced scalability and availability.

☒ Disabled

The Redis cluster will have a single shard (node group) with one primary node and up to 5 read replica.

Let the cluster mode be disabled

i If you choose cluster mode disabled you cannot change the number of shards. The configuration supports all Redis commands and functionality but limits maximum cache size and performance. [Learn more](#)

Cluster info

Use the following options to configure the cluster.

Name

cloudinstitution-redis-cluster

Give name and description

The name can have up to 40 characters, and must not contain spaces.

Description - optional

Example: cloudinstitution-redis-cluster

Location

Choose whether to host the cluster in the AWS Cloud or on premises.

Location

☒ AWS Cloud

Use the AWS Cloud for your ElastiCache instances.

☐ On premises

Create your ElastiCache instances on an Outpost (through AWS Outposts). You need to create a subnet ID on an Outpost first.

Select the Location where you want to host the cluster

Multi-AZ

☐ Enable

Multi-AZ provides enhanced high availability through automatic failover to a read replica, cross AZs, in case of a primary node failover.

Auto-failover

☐ Enable

ElastiCache Auto Failover provides enhanced high availability through automatic failover to a read replica in case of a primary node failover.

Cluster settings

Use the following options to configure the cluster.

Engine version
Version compatibility of the Redis engine that will run on your nodes.

7.1

Port
The port number that nodes accept connections on.

6379

Parameter groups
Parameter groups control the runtime properties of your nodes and clusters.

default.redis7

Node type
The type of node to be deployed and its associated memory size.

cache.r6g.large
13.07 GiB memory Up to 10 Gigabit network performance

Number of replicas
Enter the number of replicas between 0 and 5. Zero replicas will not enable an enhanced cluster with primary/replica roles.

Annotations:

- Select the latest engine version (points to Engine version dropdown)
- Enter TCP port 6379 (points to Port input)
- Select a node type (points to Node type dropdown)

Connectivity

Choose the IP version(s) this cluster will support. Then select an existing subnet group or create a new one.

Network type
Choose between IPv4, dual stack and IPv6

IPv4
Your resources will communicate only over the IPv4 protocol.

Subnet groups

☐ Choose existing subnet group

☒ Create a new subnet group

Name

redis-sn1

The name can have up to 255 characters, and must not contain spaces.

Description - optional

redis-subnetgroup

Annotations:

- Create a new subnet group (points to Create a new subnet group radio button)
- Give name and description (points to Name input)

Connectivity

Choose the IP version(s) this cluster will support. Then select an existing subnet group or create a new one.

Network type

Choose between IPv4, dual stack and IPv6

IPv4

Your resources will communicate only over the IPv4 protocol.

Subnet groups

☐ Choose existing subnet group

☒ Create a new subnet group

Name

redis-sn1

The name can have up to 255 characters, and must not contain spaces.

Description - optional

redis-subnetgroup

Select the required VPC

VPC ID

The identifier for the VPC environment where your cluster is to run.

vpc-0c3cd85059c1bcf31

Create VPC [↗](#)

VPC ID

The identifier for the VPC environment where your cluster is to run.

vpc-0c3cd85059c1bcf31

Create VPC [↗](#)

i For Multi-AZ high availability mode, choose IDs for at least two subnets from two Availability Zones in the table below.

Click on Manage

Selected subnets (3)

Manage

Availability Zone ▲	Subnet ID ▼	Subnet name ▼	CIDR block (IPv4) ▼
ap-south-1a	subnet-028d7974e8e42f6d0	-	172.31.32.0/20
ap-south-1b	subnet-0f5a609862be8c27e	-	172.31.0.0/20
ap-south-1c	subnet-0c8542891f0108f78	-	172.31.16.0/20

The name can have up to 255 characters, and must not contain spaces.

Manage subnets

Add or remove subnets from the table

Subnets (3/3)

Find subnets

<input checked="" type="checkbox"/>	Availability Zone ▲	Subnet ID ▼	Subnet name ▼	CIDR block (IPv4) ▼
<input checked="" type="checkbox"/>	ap-south-1a	subnet-028d7974e8e42f6d0	-	172.31.32.0/20
<input checked="" type="checkbox"/>	ap-south-1b	subnet-0f5a609862be8c27e	-	172.31.0.0/20
<input checked="" type="checkbox"/>	ap-south-1c	subnet-0c8542891f0108f78	-	172.31.16.0/20

Cancel Choose

ap-south-1c subnet-0c8542891f0108f78 172.31.16.0/20

Availability Zone placements

Use the following fields to configure placements for Availability Zones.

Availability Zone placements

By locating nodes in different Availability Zones, you reduce the chance that a failure in one Availability Zone, such as a power outage, will cause your entire system to fail. Choose **Specify Availability Zones** if you want to specify Availability Zones for cluster nodes.

No preference Create next

Cancel Next

Step 1
[Cluster settings](#)

Step 2
Advanced settings

Step 3
Review and create

Advanced settings [info](#)

Security

Use the following section to configure network security and data security for your cluster.

Encryption at rest

☐ **Enable**
Enables encryption of data stored on disk.

Encryption in transit [info](#)

☐ **Enable**
Enables encryption of data that moves between the service and client.

Selected security groups (0)

A security group acts like a firewall that controls network access to your clusters.

Group ID [🔗](#) **Name**

No selected security groups

Add security groups by choosing the Manage button.

[Manage](#)

[Click manage](#)

Services | Search

Step 1
[Cluster settings](#)

Step 2
Advanced settings

Step 3
Review and create

Manage security groups

Security groups (1/25)

< 1 2 3 >

<input type="checkbox"/>	Security group ID 🔗	Security group name	
<input type="checkbox"/>	sg-07d363965018c339c	launch-wizard-17	
<input type="checkbox"/>	sg-082836b6570231857	launch-wizard-12	launch-wizard-12 created 2024-04-17T10:56:39.2
<input type="checkbox"/>	sg-082ea443ec2f2f59b	launch-wizard-1	launch-wizard-1 created 2024-02-12T06:44:20.8
<input type="checkbox"/>	sg-0862ea5cb1cd0da2f	launch-wizard-16	launch-wizard-16 created 2024-04-23T06:15:43.3
<input checked="" type="checkbox"/>	sg-08b551e0feee66604	redis-sg	launch-wizard-23 created 2024-06-05T07:53:56.4
<input type="checkbox"/>	sg-08ce2fe1ae5de0860	launch-wizard-22	launch-wizard-22 created 2024-06-05T07:24:26.8
<input type="checkbox"/>	sg-0afcb799dcd68c59f	launch-wizard-4	launch-wizard-4 created 2024-03-26T04:24:56.7
<input type="checkbox"/>	sg-0b951d855cdd71f6c	launch-wizard-13	launch-wizard-13 created 2024-04-22T06:41:37.5
<input type="checkbox"/>	sg-0c935e0187be98f59	launch-wizard-9	launch-wizard-9 created 2024-04-15T07:14:11.3
<input type="checkbox"/>	sg-0c9bdfdbf979f8c5d	launch-wizard-3	launch-wizard-3 created 2024-02-12T08:20:05.7

[Cancel](#) [Choose](#)

[Select the security group that contains Custom TCP with port 6379](#)

Backup

You can use backups to restore a cluster or seed a new cluster. The backup consists of the cluster's metadata, along with all of the data in the cluster.

- ☐ **Enable automatic backups**
ElastiCache will automatically create a daily backup of a set of replicas.

Maintenance

Configure maintenance settings for the cluster.

Maintenance window

Specify the time range (UTC) for updates such as patching an operating system, updating drivers, and installing software or patches.

- ☒ **No preference**
☐ **Specify maintenance window**

Auto upgrade minor versions

- ☒ **Enable**
Automatically schedule cluster upgrade to the latest minor version, once it becomes available.
Cluster upgrade will only be scheduled during the maintenance window.

Topic for Amazon SNS notification

Choose an SNS topic from the list, or enter the Amazon Resource Name (ARN) for an existing topic. If no topic is chosen, no notifications are sent.

Logs

Specify whether to provide the Redis slow logs or engine logs.

Slow logs

- ☐ **Enable**
Provide the Redis slow log for queries that exceed a specified runtime.

Engine logs

- ☐ **Enable**
Provide the engine log for queries that exceed a specified runtime.

Tags

You can use tags to search and filter your clusters, or track your AWS costs.

No tags associated with the cluster.

Add new tag

You can add 50 more tags.

Click next

Cancel

Previous

Next

Logs

Specify whether to provide the Redis slow logs or engine logs.

Slow logs	Engine logs	
Disabled	Disabled	

Tags

You can use tags to search and filter your clusters, or track your AWS costs.

Key	Value
No tags found. Tags that you create will appear here.	

Cancel Previous Create

Review and click create

Redis Cache created successfully

The cluster was created successfully.

ElastiCache > Redis caches

Redis caches (1) Info

Find Redis caches

< 1 >

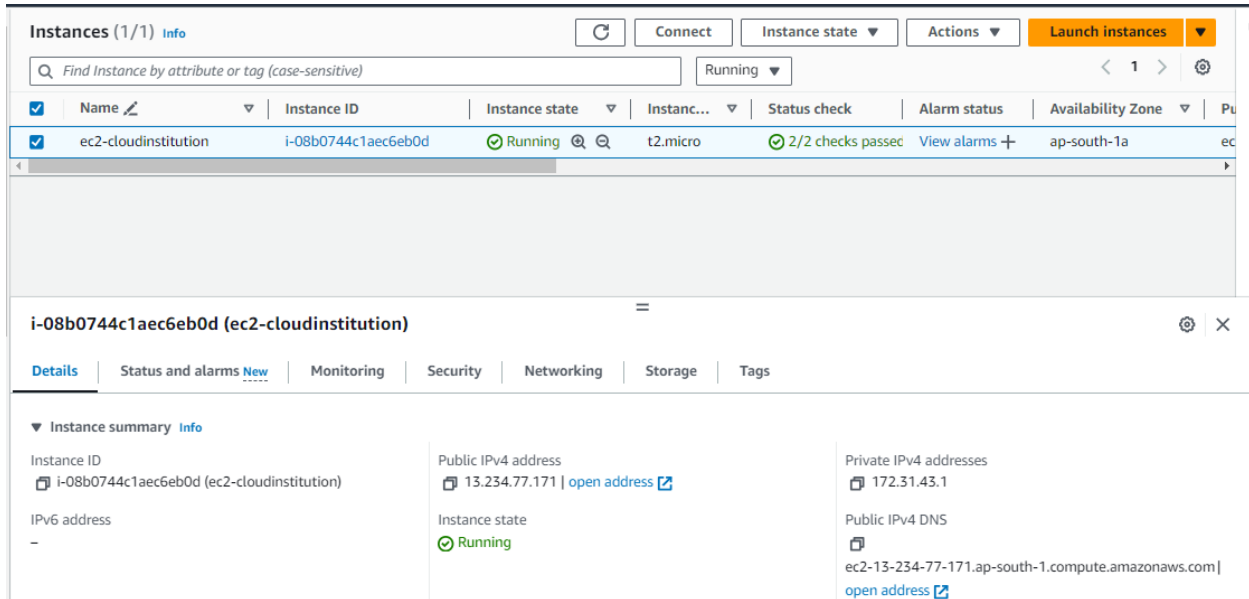
View details

Actions

Create Redis cache

	Cache name	Status	Description	Engine version	Configuration	Created date
	cloudinstitution-redis-cluster	Creating	Example-cloudinstitution-redis-cluster	7.1.0	cache.r6g.large	June 5, 202

Step 3 : Create a EC2 instance (select the created Security Group [example : redis-sg] while creating instance)



Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) Running

Name	Instance ID	Instance state	Instanc...	Status check	Alarm status	Availability Zone	Public IP
ec2-cloudinstitution	i-08b0744c1aec6eb0d	Running	t2.micro	2/2 checks passed	View alarms	ap-south-1a	13.234.77.171

i-08b0744c1aec6eb0d (ec2-cloudinstitution)

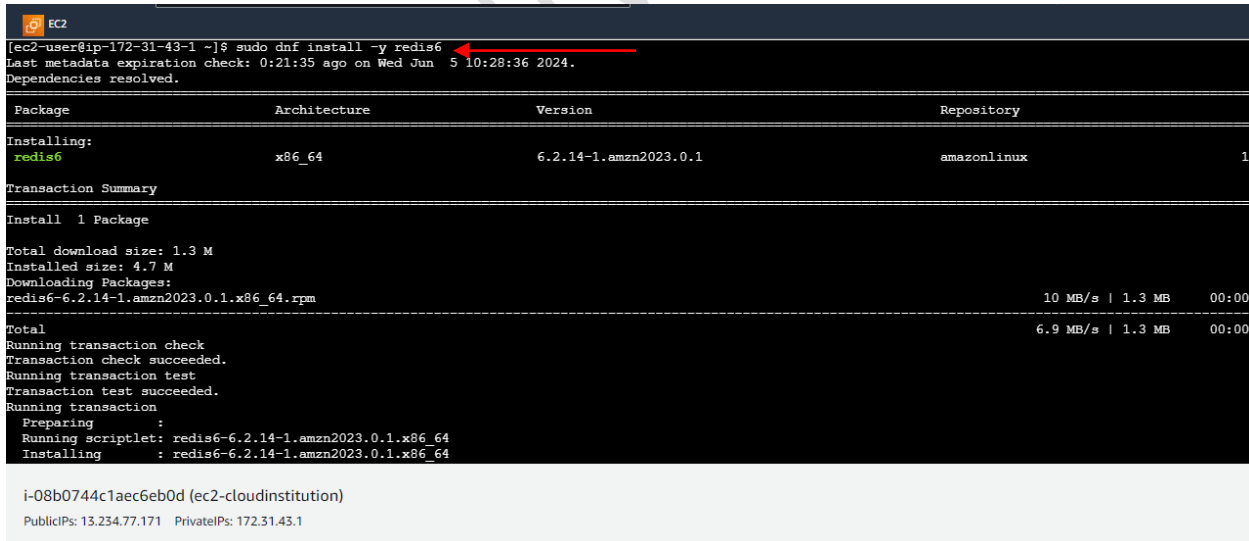
Details | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID i-08b0744c1aec6eb0d (ec2-cloudinstitution)	Public IPv4 address 13.234.77.171 open address	Private IPv4 addresses 172.31.43.1
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-13-234-77-171.ap-south-1.compute.amazonaws.com open address

Step 4 : Install Redis in the EC2 instance

“**sudo dnf install -y redis6**” - By running this command, you'll initiate the installation process. DNF will automatically locate the necessary Redis packages and their dependencies, download them, and install them on your system. (redis6 is the version here)



```

[ec2-user@ip-172-31-43-1 ~]$ sudo dnf install -y redis6
Last metadata expiration check: 0:21:35 ago on Wed Jun  5 10:28:36 2024.
Dependencies resolved.

Package                Architecture    Version           Repository
-----                -
Installing:
redis6                 x86_64         6.2.14-1.amzn2023.0.1  amazonlinux

Transaction Summary
-----
Install 1 Package

Total download size: 1.3 M
Installed size: 4.7 M
Downloading Packages:
redis6-6.2.14-1.amzn2023.0.1.x86_64.rpm                                10 MB/s | 1.3 MB  00:00
-----
Total                                                                6.9 MB/s | 1.3 MB  00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                :
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing               : redis6-6.2.14-1.amzn2023.0.1.x86_64

```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

“ **sudo systemctl start redis6** ” - is for starting the Redis server on most Linux distributions that use systemd for service management.

```
aws Services Search [Alt+S]
EC2
Transaction Summary
-----
Install 1 Package

Total download size: 1.3 M
Installed size: 4.7 M
Downloading Packages:
redis6-6.2.14-1.amzn2023.0.1.x86_64.rpm
-----
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : 
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing      : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Verifying       : redis6-6.2.14-1.amzn2023.0.1.x86_64

Installed:
  redis6-6.2.14-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl start redis6
[ec2-user@ip-172-31-43-1 ~]$
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

“ **sudo systemctl enable redis6** ” – is for enabling the Redis service on most Linux distributions that use systemd for service management.

```
aws Services Search [Alt+S]
EC2
Transaction Summary
-----
Install 1 Package

Total download size: 1.3 M
Installed size: 4.7 M
Downloading Packages:
redis6-6.2.14-1.amzn2023.0.1.x86_64.rpm
-----
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : 
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing      : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Verifying       : redis6-6.2.14-1.amzn2023.0.1.x86_64

Installed:
  redis6-6.2.14-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl start redis6
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl enable redis6
Created symlink /etc/systemd/system/multi-user.target.wants/redis6.service → /usr/lib/systemd/system/redis6.service.
[ec2-user@ip-172-31-43-1 ~]$
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

“ **sudo systemctl is-enabled redis6** ” - is used to check if the Redis service is configured to start automatically on system boot on Linux distributions that use systemd for service management.

```
aws Services [Alt+S]
EC2
Total download size: 1.3 M
Installed size: 4.7 M
Downloading Packages:
redis6-6.2.14-1.amzn2023.0.1.x86_64.rpm
-----
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing     : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Verifying      : redis6-6.2.14-1.amzn2023.0.1.x86_64

Installed:
  redis6-6.2.14-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl start redis6
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl enable redis6
Created symlink /etc/systemd/system/multi-user.target.wants/redis6.service → /usr/lib/systemd/system/redis6.service.
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl is-enabled redis6 ←
enabled
[ec2-user@ip-172-31-43-1 ~]$

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1
```

“ **redis6-server --version** ” - is to check the version of the Redis server installed on your system.

```
EC2
Downloading Packages:
redis6-6.2.14-1.amzn2023.0.1.x86_64.rpm
-----
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing     : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Verifying      : redis6-6.2.14-1.amzn2023.0.1.x86_64

Installed:
  redis6-6.2.14-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl start redis6
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl enable redis6
Created symlink /etc/systemd/system/multi-user.target.wants/redis6.service → /usr/lib/systemd/system/redis6.service.
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl is-enabled redis6
enabled
[ec2-user@ip-172-31-43-1 ~]$ redis6-server --version ←
Redis server v=6.2.14 sha=00000000:0 malloc=jemalloc-5.1.0 bits=64 build=42ac571a5183f322
[ec2-user@ip-172-31-43-1 ~]$

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1
```

“ redis6-cli ping “ - is used to check if a Redis server is up and running.

```
EC2

Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : 
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing      : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Verifying       : redis6-6.2.14-1.amzn2023.0.1.x86_64

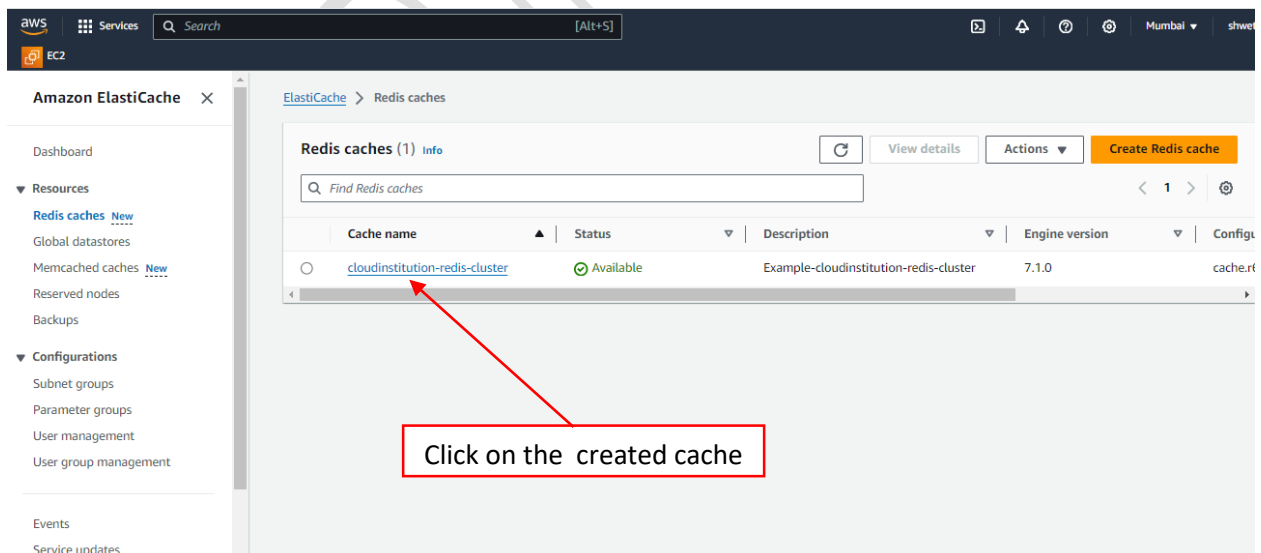
Installed:
  redis6-6.2.14-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl start redis6
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl enable redis6
Created symlink /etc/systemd/system/multi-user.target.wants/redis6.service → /usr/lib/systemd/system/redis6.service.
[ec2-user@ip-172-31-43-1 ~]$ sudo systemctl is-enabled redis6
enabled
[ec2-user@ip-172-31-43-1 ~]$ redis6-server --version
Redis server v=6.2.14 sha=000000000:0 malloc=jemalloc-5.1.0 bits=64 build=42ac571a5183f322
[ec2-user@ip-172-31-43-1 ~]$ redis6-cli ping ←
PONG
[ec2-user@ip-172-31-43-1 ~]$
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)

PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

Now go to the cache console



The screenshot shows the Amazon ElastiCache console. On the left is a navigation menu with options like Dashboard, Resources, Configurations, and Events. The main area displays 'Redis caches (1)' with a table containing one entry: 'cloudinstitution-redis-cluster' with status 'Available'. A red arrow points to this entry, and a red box with the text 'Click on the created cache' is overlaid on the image.

Cache name	Status	Description	Engine version	Config
cloudinstitution-redis-cluster	Available	Example-cloudinstitution-redis-cluster	7.1.0	cache.r

cloudinstitution-redis-cluster [Info](#)




Start migration

Modify

Backup

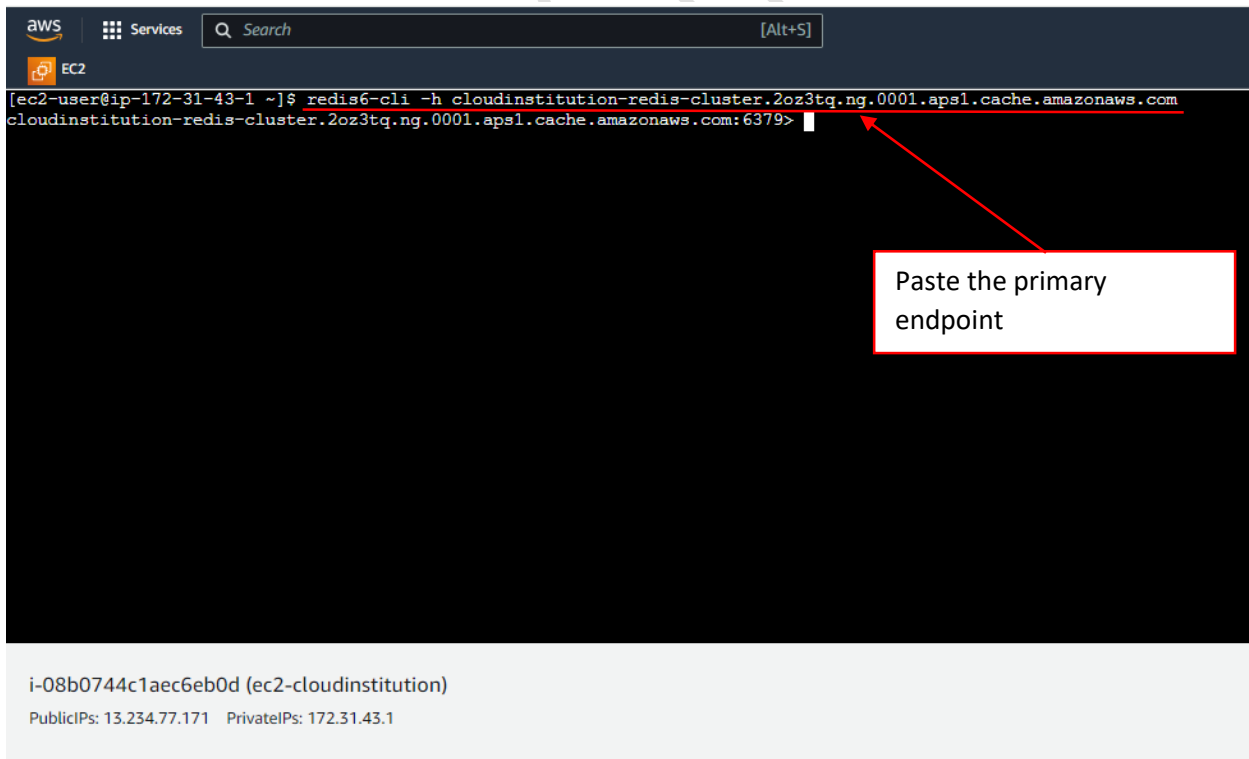
Delete

▼ Cluster details

Cluster name cloudinstitution-redis-cluster	Description Example-cloudinstitution-redis-cluster	Node type cache.r6g.large	Status ✔ Available
Engine Redis	Engine version 7.1.0	Global datastore -	Global datastore role -
Update status Up to date	Cluster mode Disabled	Shards 1	Number of nodes 3
Data tiering Disabled		Auto-failover Disabled	Encryption in transit Disabled
Encryption at rest Disabled	default.redis7	Outpost ARN -	Configuration endpoint -
Primary endpoint  cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379	Reader endpoint  cloudinstitution-redis-cluster-ro.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379	ARN  arn:aws:elasticache:ap-south-1:473869189128:replicationgroup:cloudinstitution-redis-cluster	Data migration No active migrations

Copy the primary endpoint

“ redis6-cli -h <paste the primary endpoint> ” - is used to connect to a remote Redis server using the Redis command-line interface (CLI) tool



```
aws
Services
Search [Alt+S]
EC2
[ec2-user@ip-172-31-43-1 ~]$ redis6-cli -h cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379>
```

Paste the primary endpoint

i-08b0744c1aec6eb0d (ec2-cloudinstitution)
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

To create a key named "cloudinstitution" we use the command **set name <key_name>**

And the command "**get name**" retrieves a value associated with the key name in the Redis cache

```
EC2
[ec2-user@ip-172-31-43-1 ~]$ redis-cli -h cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> set name cloudinstitution
OK
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> get name
"cloudinstitution"
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> 
```

Similarly create some keys like course and duration

```
EC2
[ec2-user@ip-172-31-43-1 ~]$ redis-cli -h cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> set name cloudinstitution
OK
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> get name
"cloudinstitution"

cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> set course AWS
OK
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> get course
"AWS"
cloudinstitution-redis-cluster.2oz3tg.ng.0001.ap.s1.cache.amazonaws.com:6379> 
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)

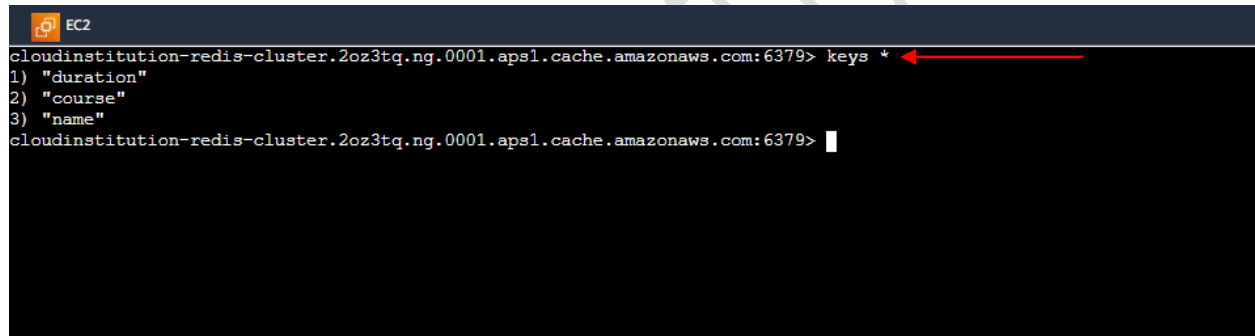
PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

```
cloudinstitution-redis-cluster.2oz3tq.ng.0001.ap.s1.cache.amazonaws.com:6379> set duration 3months
OK
cloudinstitution-redis-cluster.2oz3tq.ng.0001.ap.s1.cache.amazonaws.com:6379> get duration
"3months"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.ap.s1.cache.amazonaws.com:6379> █
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)

PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

The command **"keys *"** in Redis is used to retrieve all the keys present in the current database of your Redis server.



```
cloudinstitution-redis-cluster.2oz3tq.ng.0001.ap.s1.cache.amazonaws.com:6379> keys *
1) "duration"
2) "course"
3) "name"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.ap.s1.cache.amazonaws.com:6379> █
```