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.

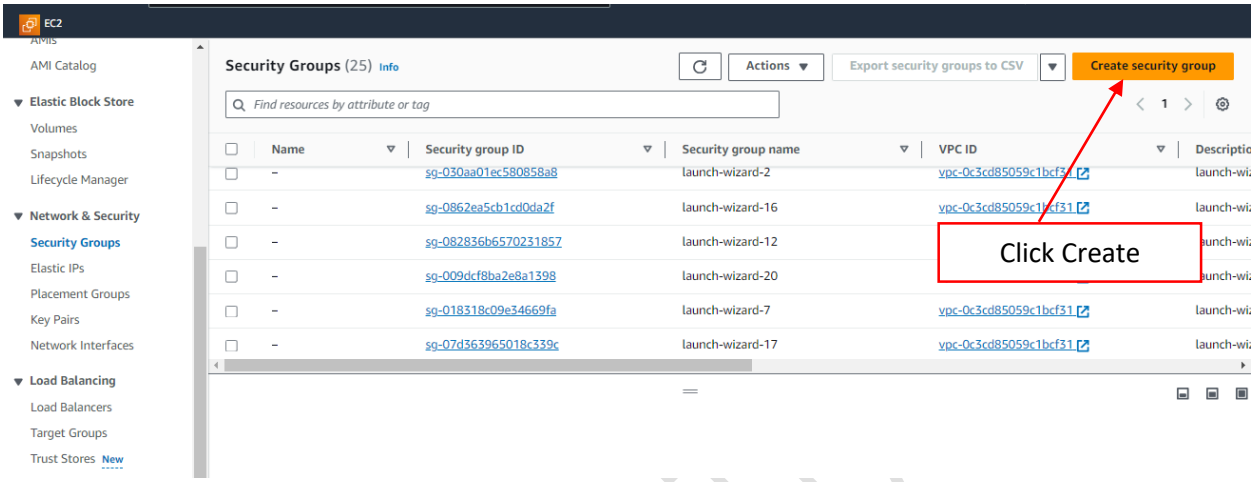
Here are some common use cases for Redis cache:

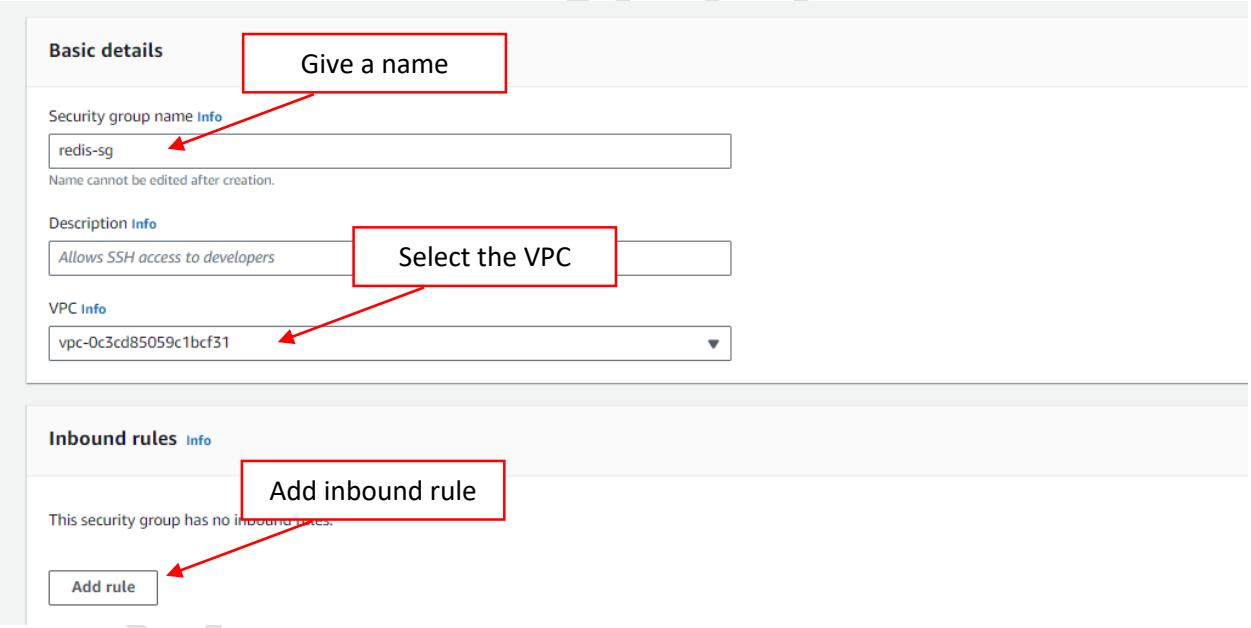
- Application Caching: Store frequently accessed data like product information, user profiles, or search results to minimize database load and improve response times.
- Session Management: Store user session data in Redis for faster retrieval and improved user experience.
- Real-time Applications: Power 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

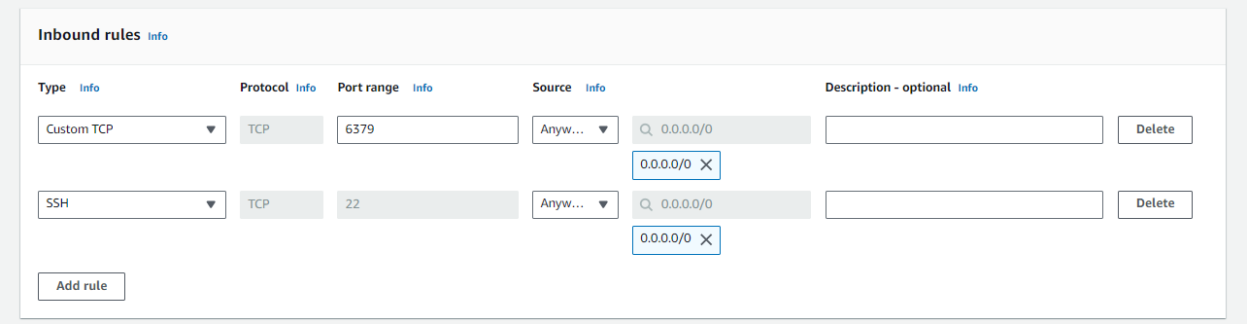
Step 1 : Create a Security Group

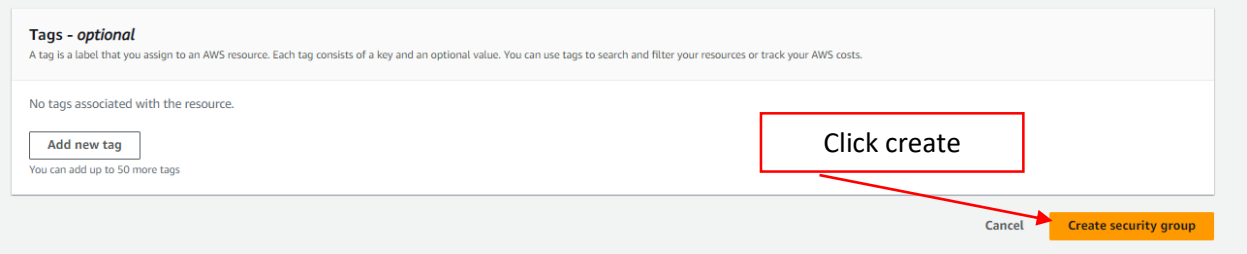




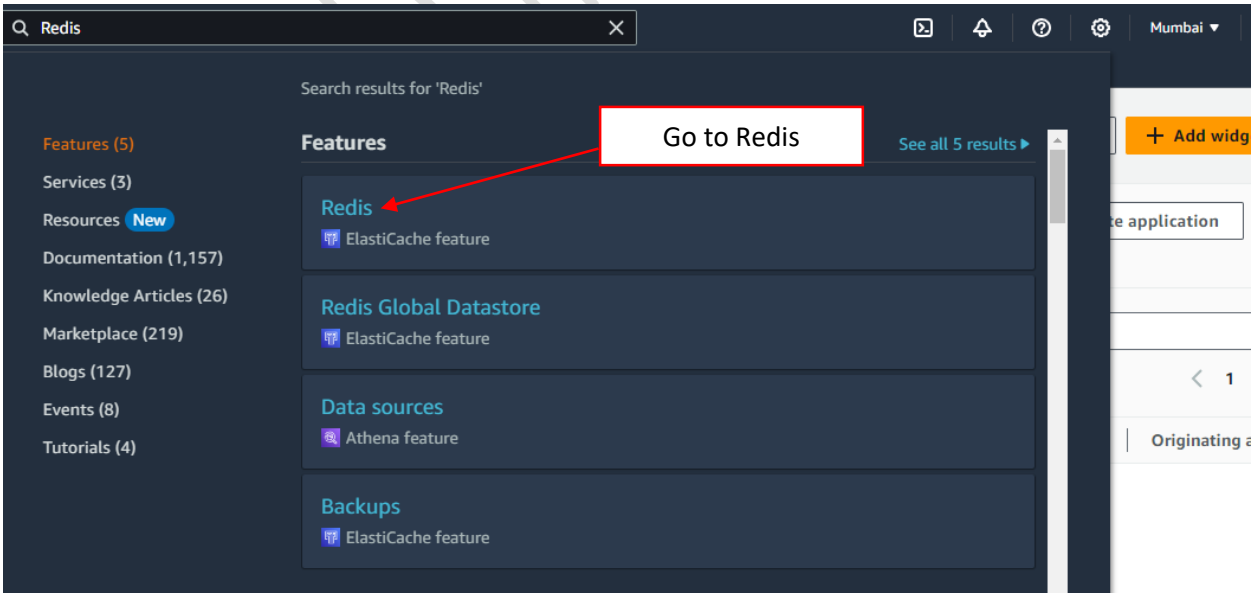
Add SSH, Custom TCP with port range 6379

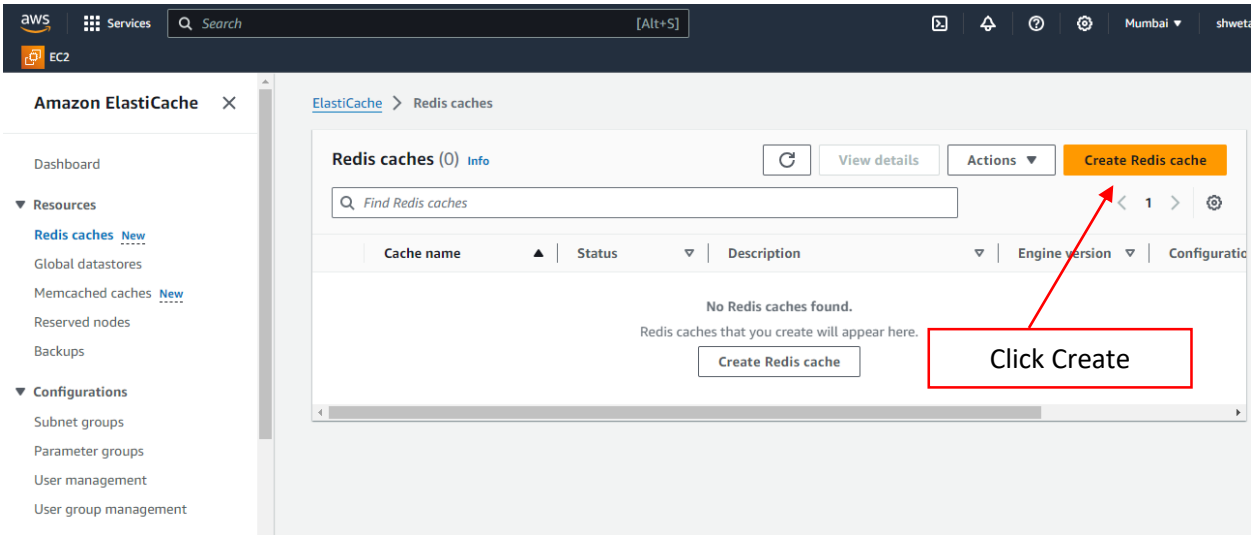
TCP port 6379 is commonly used by Redis, an in-memory data structure store that is often used as a database, cache, and message broker. Redis uses this port for its default client-server communication protocol.

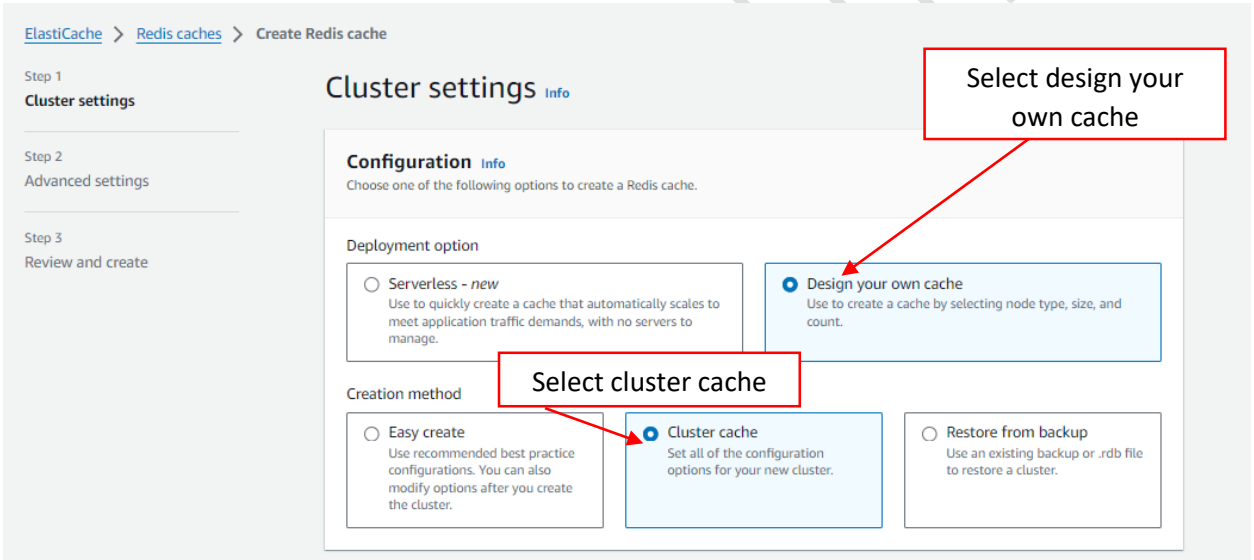




Step 2 : Create a Cache




aws
Services
Search
[Alt+S]
Mumbai
EC2
Amazon ElastiCache
Dashboard
Resources
Redis caches New
Global datastores
Memcached caches New
Reserved nodes
Backups
Configurations
Subnet groups
Parameter groups
User management
User group management
ElastiCache > Redis caches
Redis caches (0) Info
View details
Actions
Create Redis cache
Find Redis caches
Cache name
Status
Description
Engine version
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
Select design your own cache
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.
Select cluster cache
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.

Cluster mode

Scale your cluster dynamically with no downtime.

Let the cluster mode be disabled

- ○ 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.

ⓘ 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.

Give name and description

Name

cloudinstitution-redis-cluster

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.

Select the Location where you want to host the cluster

Location

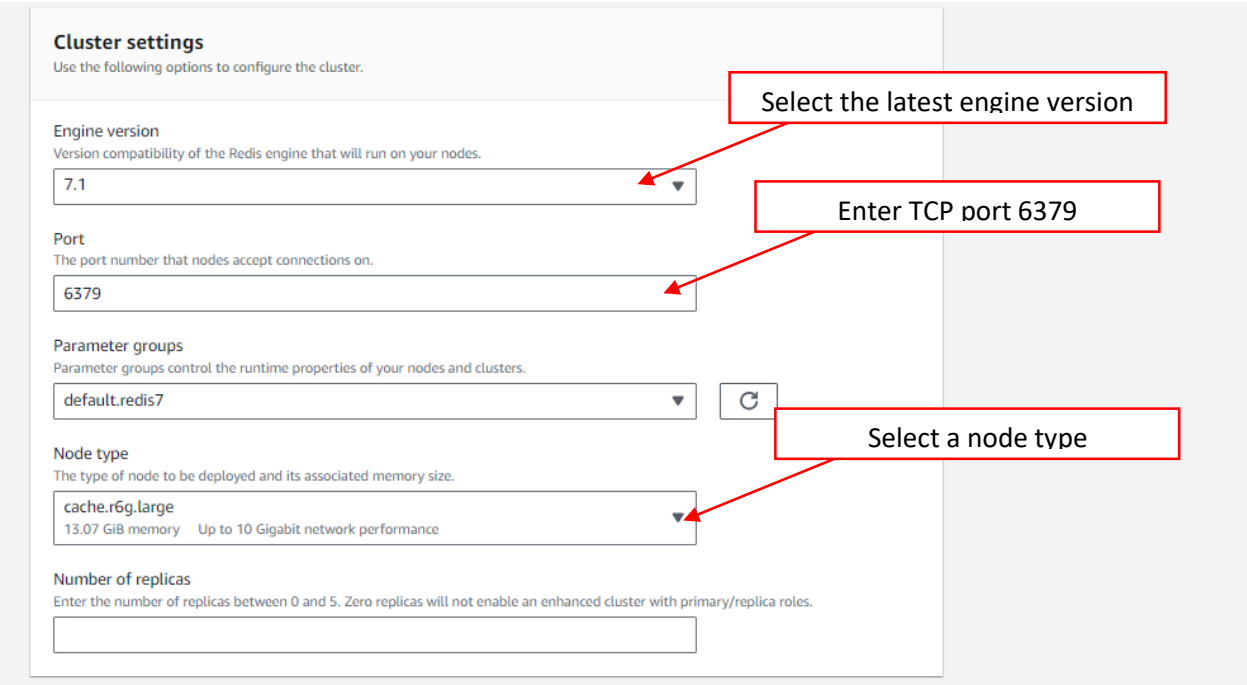
- ◉ AWS Cloud
  Use the AWS Cloud for your ElastiCache instances.
- ○ On premises
  Create your ElastiCache instances on an Outpost (through AWS Outposts). You need to create a subnet ID on an Outpost first.

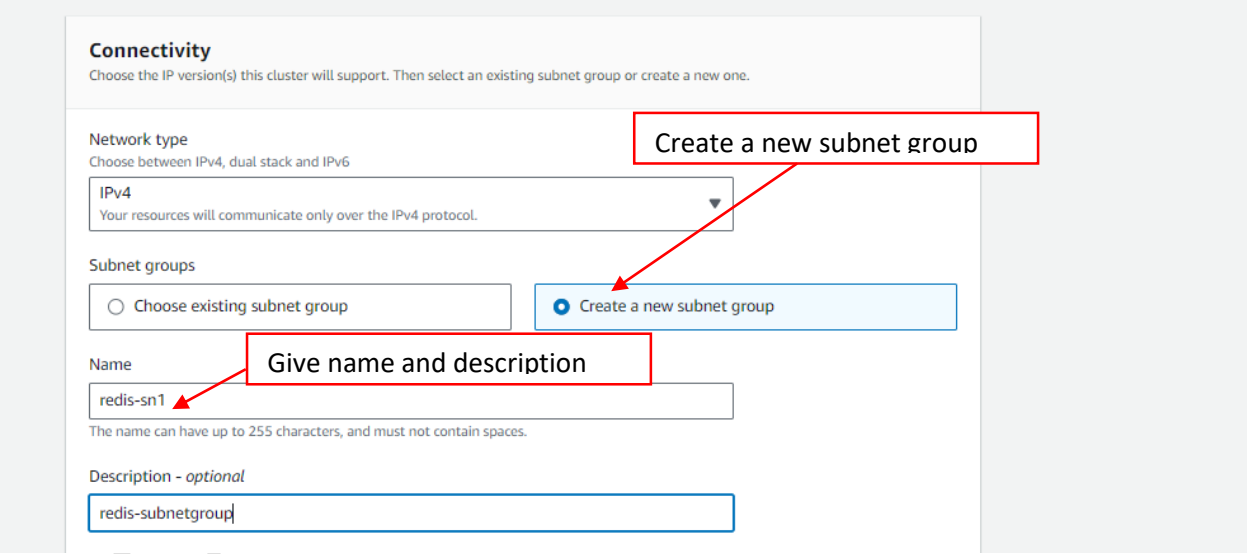
Multi-AZ

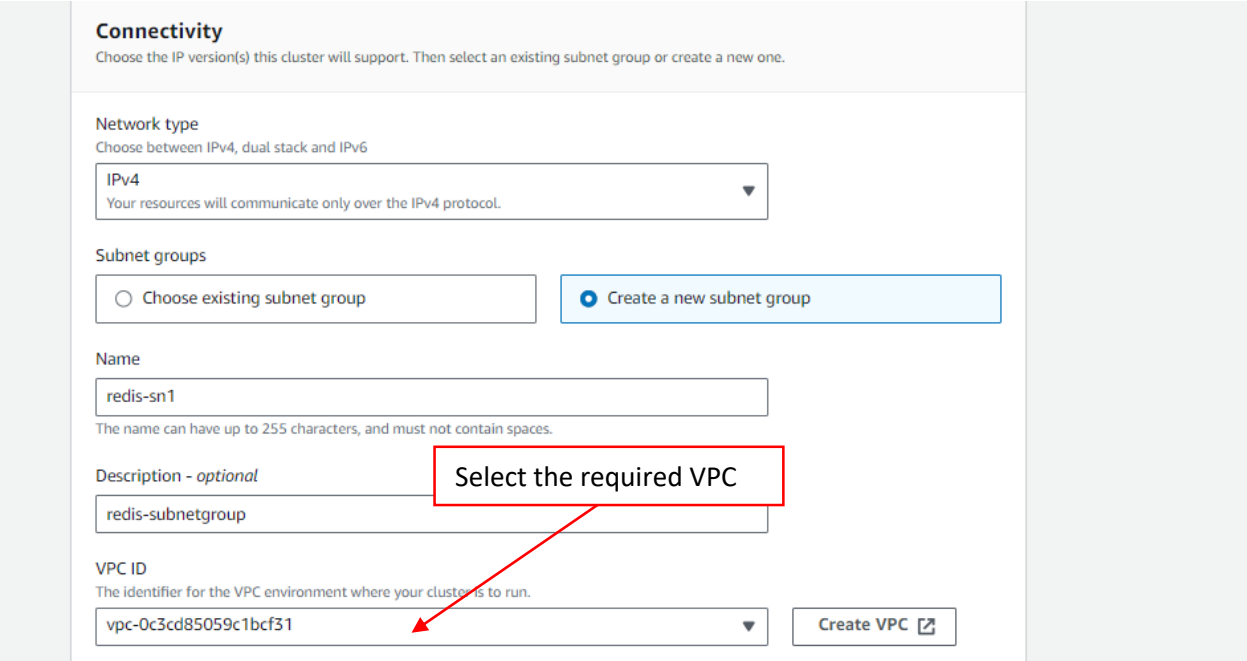
☐ Enable
Multi-AZ provides enhanced high availability through automatic failover to a read replica, cross AZs, in case of a primary node failover.

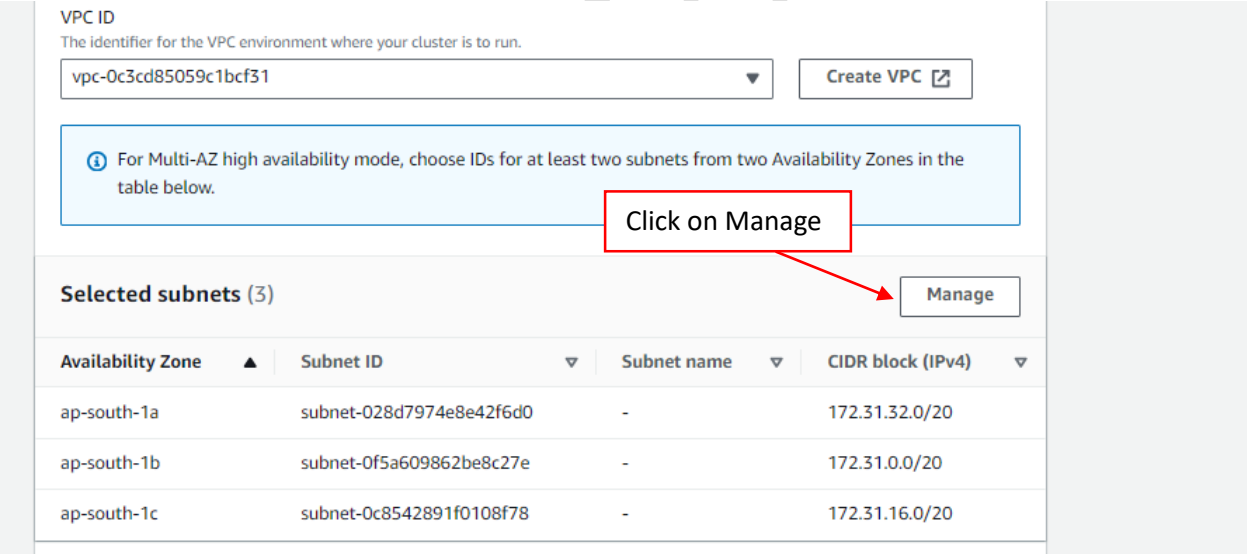
Auto-failover

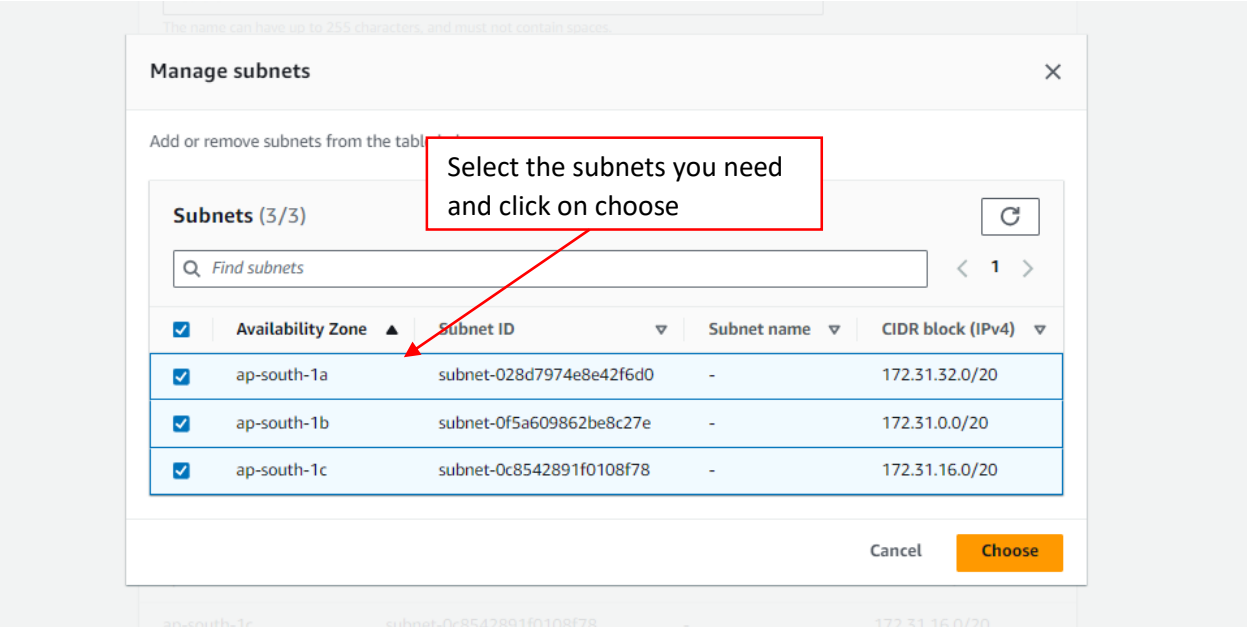
☐ Enable
ElastiCache Auto Failover provides enhanced high availability through automatic failover to a read replica in case of a primary node failover.

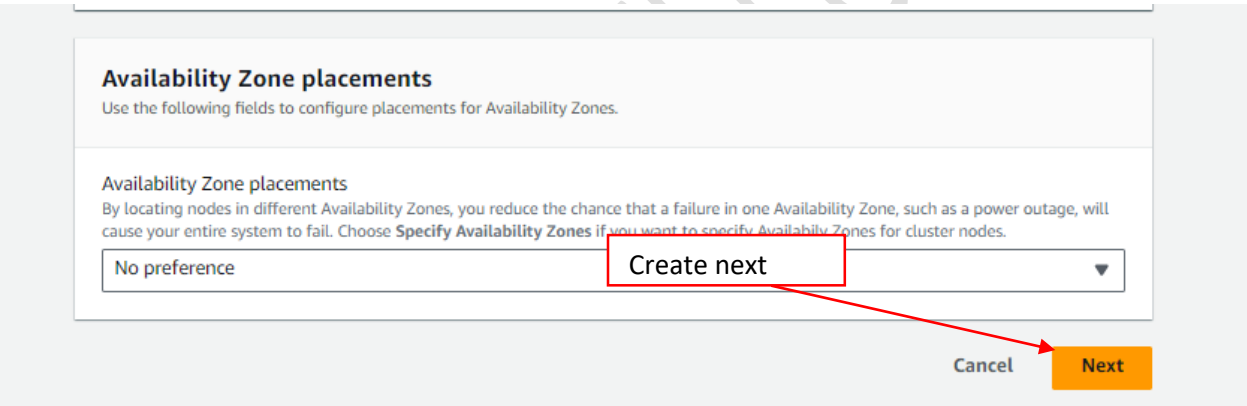

Cluster settings
Use the following options to configure the cluster.
Select the latest engine version
Engine version
Version compatibility of the Redis engine that will run on your nodes.
7.1
Enter TCP port 6379
Port
The port number that nodes accept connections on.
6379
Parameter groups
Parameter groups control the runtime properties of your nodes and clusters.
default.redis7
Select a node type
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.

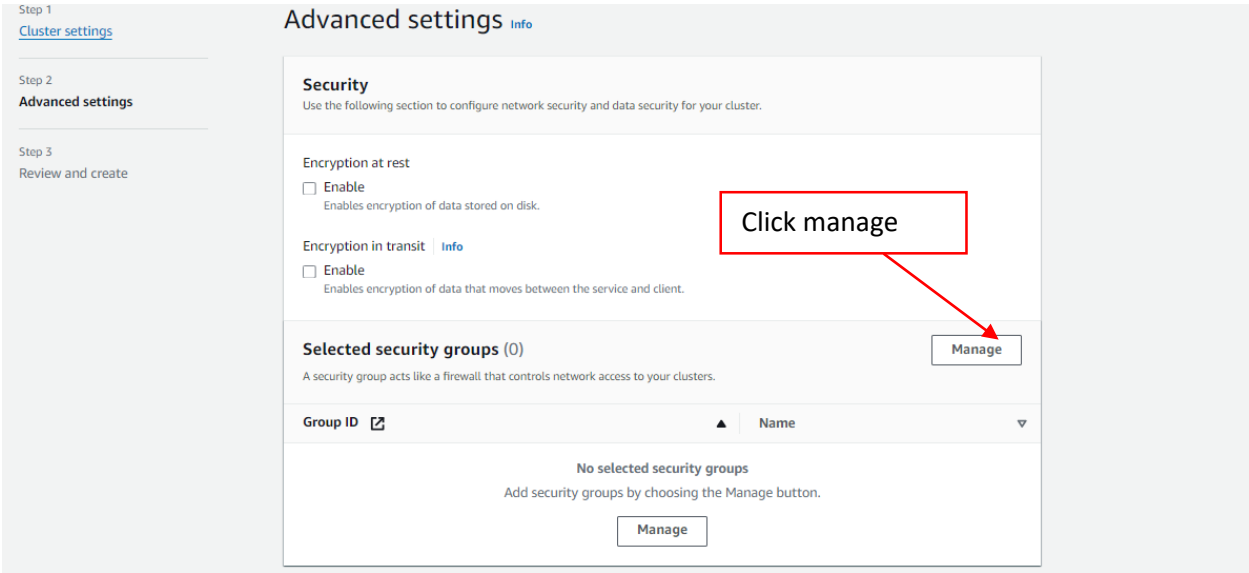

Connectivity
Choose the IP version(s) this cluster will support. Then select an existing subnet group or create a new one.
Create a new subnet group
Network type
Choose between IPv4, dual stack and IPv6
IPv4
Your resources will communicate only over the IPv4 protocol.
Subnet groups
Choose existing subnet group
Create a new subnet group
Give name and description
Name
redis-sn1
The name can have up to 255 characters, and must not contain spaces.
Description - optional
redis-subnetgroup

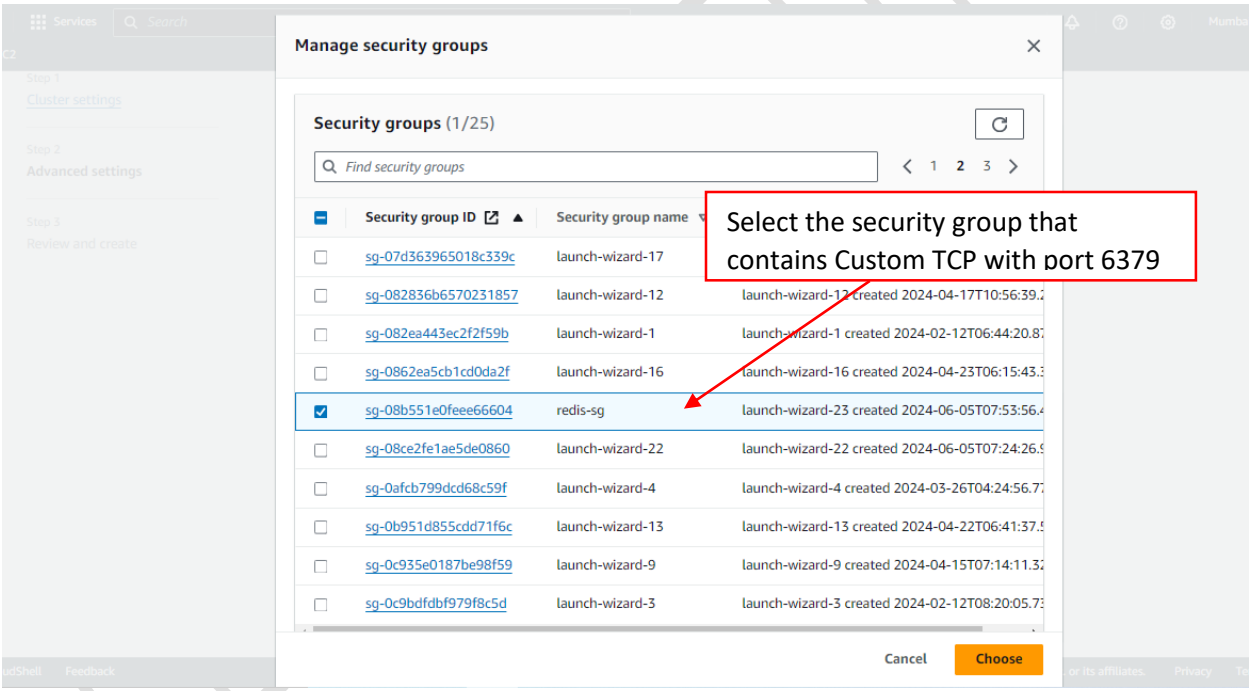

Connectivity
Choose the IP version(s) this cluster will support. Then select an existing subnet group or create a new one.
Network type
Choose between IPv4, dual stack and IPv6
IPv4
Your resources will communicate only over the IPv4 protocol.
Subnet groups
Choose existing subnet group
Create a new subnet 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
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


Manage subnets
Add or remove subnets from the tabl
Select the subnets you need and click on choose
Subnets (3/3)
Find subnets
1
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
Cancel
Choose


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.
Click manage
Selected security groups (0)
Manage
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


Manage security groups
Security groups (1/25)
Find security groups
1 2 3
Security group ID
Security group name
Select the security group that contains Custom TCP with port 6379
sg-07d363965018c339c launch-wizard-17
sg-082836b6570231857 launch-wizard-12
sg-082ea443ec2f2f59b launch-wizard-1
sg-0862ea5cb1cd0da2f launch-wizard-16
sg-08b551e0feee66604 redis-sg
sg-08ce2fe1ae5de0860 launch-wizard-22
sg-0afcb799dcd68c59f launch-wizard-4
sg-0b951d855cdd71f6c launch-wizard-13
sg-0c935e0187be98f59 launch-wizard-9
sg-0c9bdfdbf979f8c5d launch-wizard-3
Cancel
Choose

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.

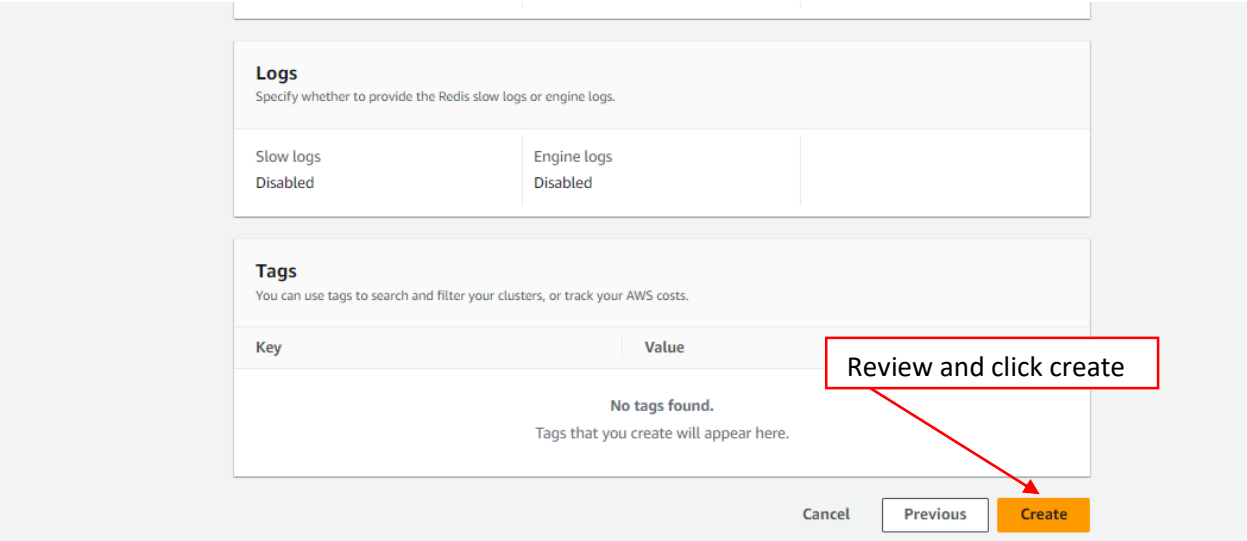
No tags associated with the cluster.

Add new tag

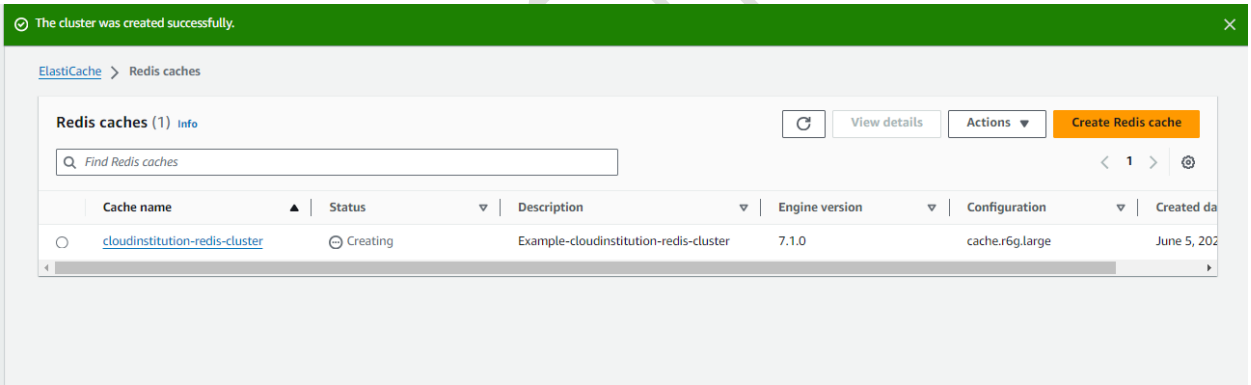
You can add 50 more tags.

Click next

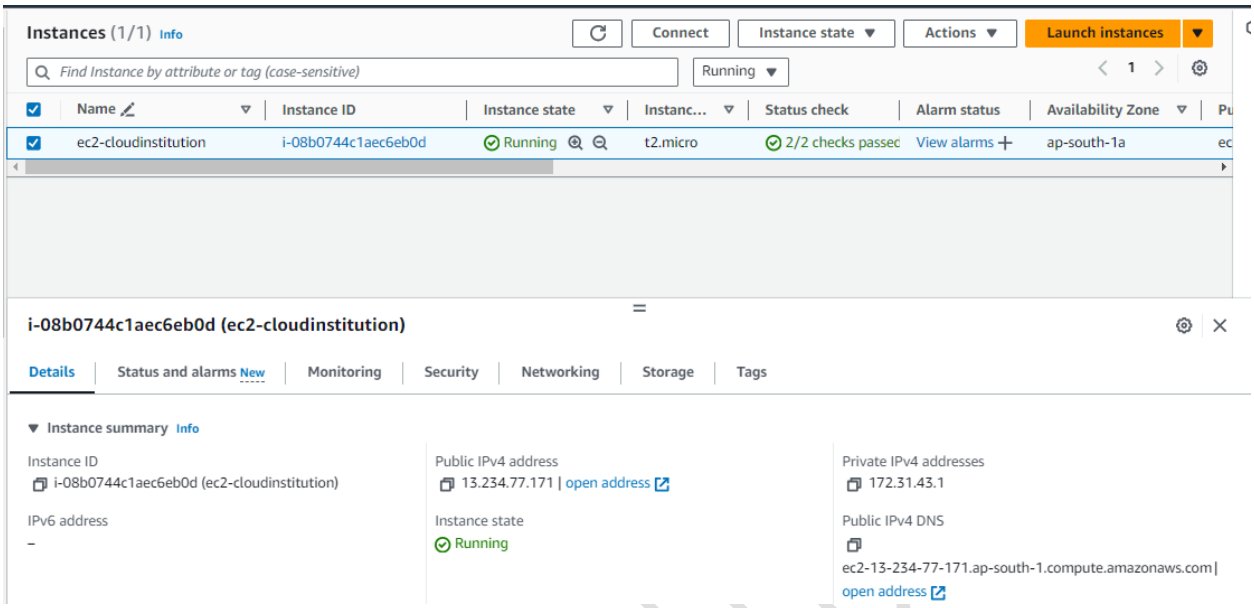
Cancel | Previous | Next



Redis Cache created successfully

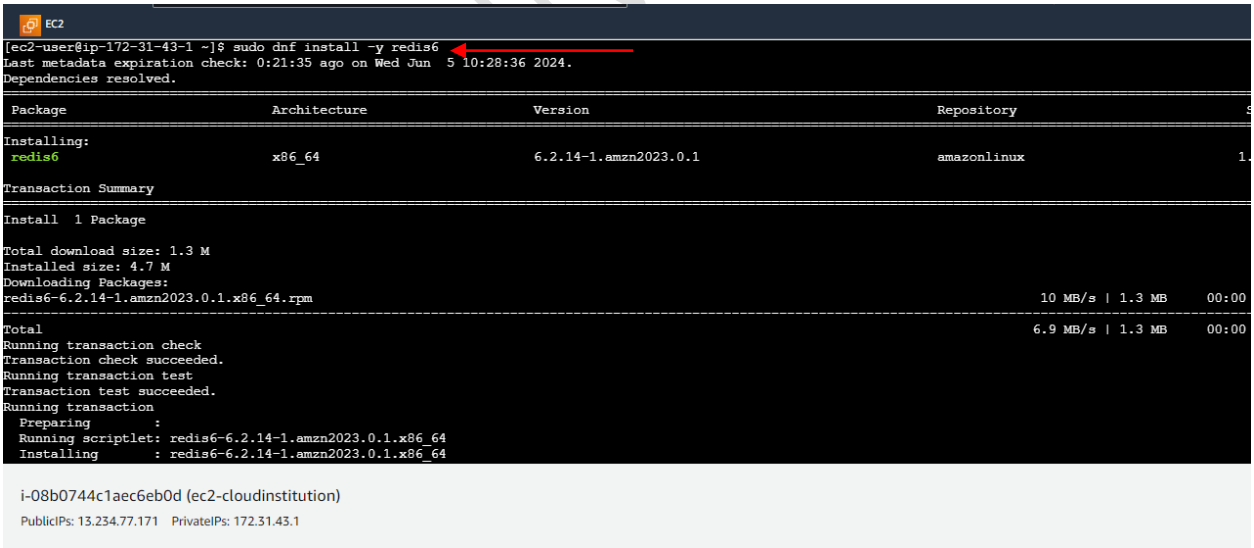


Step 3 : Create a EC2 instance (select the created Security Group [example : redis-sg] while creating instance)



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)



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

```
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.

```
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.

```
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.

```
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

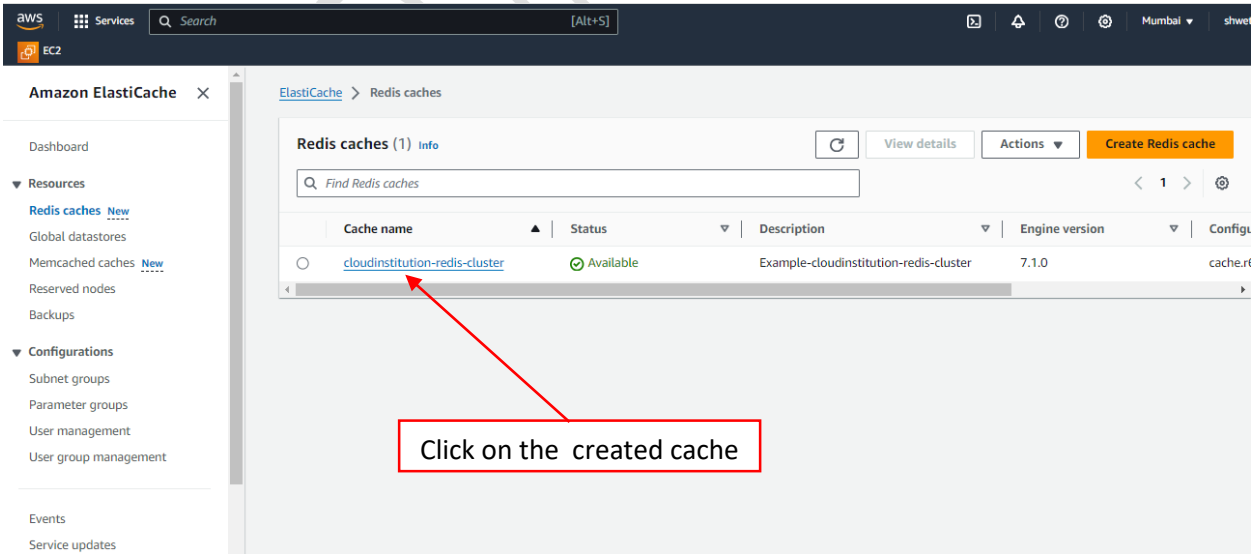
“ **redis6-cli ping** “ - is used to check if a Redis server is up and running.

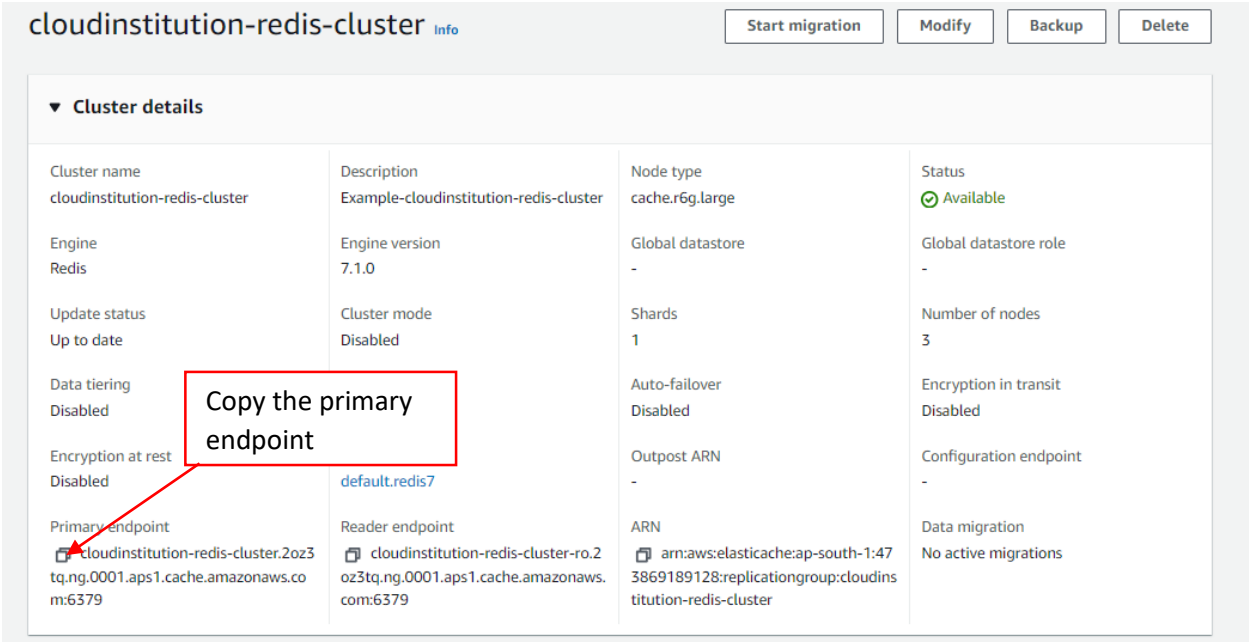
```
EC2
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  Installing       : redis6-6.2.14-1.amzn2023.0.1.x86_64
  Running scriptlet: redis6-6.2.14-1.amzn2023.0.1.x86_64
  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=00000000:0 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





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

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 ~]$ redis6-cli -h cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> set name cloudinstitution
OK
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> get name
"cloudinstitution"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379>
```

Similarly create some keys like course and duration

```
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> set name cloudinstitution
OK
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> get name
"cloudinstitution"

cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> set course AWS
OK
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> get course
"AWS"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379>
```

i-08b0744c1aec6eb0d (ec2-cloudinstitution)

PublicIPs: 13.234.77.171 PrivateIPs: 172.31.43.1

```
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> set duration 3months
OK
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> get duration
"3months"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.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.

EC2

```
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379> keys *
1) "duration"
2) "course"
3) "name"
cloudinstitution-redis-cluster.2oz3tq.ng.0001.aps1.cache.amazonaws.com:6379>
```