installation - docker

🚀 نصب Redis Stack با Docker

1. پیش‌نیازها

- نصب Docker روی سیستم
  (برای تست: `docker --version`)

---

2. اجرای Redis Stack (ساده‌ترین حالت)

```
docker run -d --name redis-server -p 6379:6379 redis
```

📌 توضیح:

- `-d` → اجرای کانتینر در background
- `--name redis-stack` → نام کانتینر
- `-p 6379:6379` → پورت اصلی Redis
- `-p 8001:8001` → پورت برای RedisInsight (پنل گرافیکی وب)
- `redis/redis-stack:latest` → ایمیج رسمی Redis Stack

---

3. دسترسی به Redis CLI داخل کانتینر

```
docker exec -it redis-stack redis-cli
```

```
docker exec -it redis-server redis-cli
```

---

4. مشاهده لاگ‌های کانتینر

```
docker logs -f redis-stack
```

---

5. اجرای Redis Stack با docker-compose (روش توصیه شده)

`docker-compose.yml`:

```
version: "3.9"
services:
  redis:
    image: redis/redis-stack:latest
    container_name: redis-stack
    ports:
      - "6379:6379"
      - "8001:8001"
```

بعد اجرا کن:

```
docker-compose up -d
```

---

6. دسترسی به داشبورد RedisInsight

- مرورگر → برو به: http://localhost:8001
- می‌تونی Redis رو مدیریت کنی، داده‌ها رو ببینی، Query بزنی.

🗄️ 📘 مستند آموزشی Redis

📘 مستند آموزشی Redis

1. مقدمه

Redis یک **پایگاه داده In-Memory** (روی RAM) و **Key-Value Store** هست.
این یعنی داده‌ها را با کلید (Key) ذخیره می‌کنه و خیلی سریع می‌تونی با کلید اون داده رو بخونی.

ویژگی‌ها:

- سرعت بالا (چند برابر دیتابیس‌های سنتی)
- پشتیبانی از انواع ساختار داده
- استفاده به عنوان Database، Cache، یا Message Broker
- پشتیبانی از TTL (زمان انقضا برای داده‌ها)

---

2. چرا Redis ؟

- **سرعت**: داده‌ها در RAM هستند.
- **سادگی**: دستورات ساده با کارایی بالا.
- **کاربردی**: مناسب برای Cache، Session Management، Pub/Sub، Leaderboard، Rate Limiting.
- **قابلیت توسعه**: توسط زبان‌های مختلف (Python, Node, Java, Go, …) پشتیبانی میشه.

---

3. جایگزین‌ها و مقایسه

- **Memcached** → کش ساده‌تر، فقط key-value.
- **RabbitMQ** → صف پیام (Message Queue) پیشرفته‌تر.
- **Kafka** → مدیریت جریان داده عظیم (Event Streaming).
- **Etcd / Consul** → مدیریت config و coordination در microservices.

Redis ترکیبی از چند ابزار بالا رو می‌تونه انجام بده.

---

4. ساختار کلی دستور

تمام دستورات Redis به شکل زیر هستند:

```
COMMAND key [value] [options]
```

مثال:

```
SET user:1 "Parham"
GET user:1
```

---

5. انواع داده‌ها در Redis

🔹 String 5.1

ساده‌ترین نوع داده.
دستورات مهم:

```
SET key value
GET key
INCR key   # افزایش عددی
DECR key   # کاهش عددی
APPEND key "text"
```

🔹 Hash 5.2

مثل Dictionary در Python (ذخیره چند فیلد در یک کلید).

```
HSET user:1 name "Ali" age 25
HGET user:1 name
HGETALL user:1
```

🔹 List 5.3

لیست پیوسته (مثل صف یا استک).

```
LPUSH tasks "task1"
```

```
RPUSH tasks "task2"
LPOP tasks
RPOP tasks
LRANGE tasks 0 -1   # نمایش کل لیست
```

Set 5.4 🔹

مجموعه (یونیک، بدون عنصر تکراری).

```
SADD tags "python" "django"
SMEMBERS tags
SISMEMBER tags "python"
```

Sorted Set 5.5 🔹

مثل Set اما با امتیاز (Score) مرتب می‌شود. عالی برای رتبه‌بندی.

```
ZADD scores 100 "Ali" 200 "Reza"
ZRANGE scores 0 -1 WITHSCORES
ZREVRANGE scores 0 -1 WITHSCORES
```

Pub/Sub 5.6 🔹

سیستم انتشار/اشتراک (Message Broker ساده).

```
SUBSCRIBE news
PUBLISH news "hello world"
```

Key Management 5.7 🔹

```
DEL key
EXISTS key
EXPIRE key 60   # انقضا 60 ثانیه
TTL key         # زمان باقی‌مانده
```

---

6. نکات مهم

- Redis روی RAM کار می‌کنه → داده‌های حجیم در Redis نریز.
- همیشه برای داده‌های موقتی TTL بذار.
- از Redis همراه دیتابیس اصلی استفاده کن (نه جایگزین کامل).
- در معماری Microservices میشه Redis رو به عنوان **Cache** یا **Message Queue** گذاشت.

---

7. جمع‌بندی

Redis ابزاری سریع و قدرتمنده که ترکیبی از **Database + Cache + Queue** هست.
یادگیری Redis هم باعث میشه **مهارت بک‌اندت بالا بره**، هم در **پرفورمنس سیستم‌ها** خیلی تفاوت ایجاد کنه.

types + commands

🔹 1. String Commands

رایج‌ترین نوع داده در Redis.

```
SET key value
```

یک کلید می‌سازه و مقدارش رو ذخیره می‌کنه.

SET name "Parham"

- (الان کلید `name` ساخته شد با مقدار `"Parham"`)

```
GET key
```

مقدار کلید رو میاره.

- GET name # خروجی: "Parham"

```
INCR key
```

مقدار کلید رو **۱ واحد زیاد می‌کنه** (فقط برای اعداد).

SET counter 10

- INCR counter # میشه 11

```
DECR key
```

مقدار کلید رو **۱ واحد کم می‌کنه**.

- DECR counter # میشه 10

```
APPEND key value
```

متن جدید رو به آخر مقدار کلید اضافه می‌کنه.

APPEND name " Ilaghi"

- GET name # خروجی: "Parham Ilaghi"

🔹 2. Hash Commands

مثل دیکشنری یا JSON.

```
HSET key field value
```

یک فیلد جدید در Hash می‌ذاره.

- HSET user:1 name "Ali" age 23

```
HGET key field
```

مقدار یک فیلد رو میاره.

- HGET user:1 name # خروجی: "Ali"

```
HGETALL key
```

همه‌ی فیلدها و مقدارها رو نشون میده.

- HGETALL user:1# خروجی: name Ali age 23

🔹 3. List Commands

لیست پیوسته → مثل صف (Queue) یا پشته (Stack).

`LPUSH key value`

مقدار رو به **ابتدای لیست** اضافه می‌کنه.

LPUSH tasks "task1"
LPUSH tasks "task2"

- # لیست: ["task2", "task1"]

`RPUSH key value`

مقدار رو به **انتهای لیست** اضافه می‌کنه.

RPUSH tasks "task3"

- # لیست: ["task2", "task1", "task3"]

`LPOP key`

مقدار **ابتدای لیست** رو برمی‌گردونه و حذف می‌کنه.

- LPOP tasks# خروجی: "task2"

`RPOP key`

مقدار **انتهای لیست** رو برمی‌گردونه و حذف می‌کنه.

- RPOP tasks# خروجی: "task3"

`LRANGE key start end`

بخشی از لیست رو نشون میده.

- LRANGE tasks 0 -1 # همه عناصر

🔹 4. Set Commands

مجموعه (unique, بدون تکرار).

```
SADD key value
```

مقدار جدید به Set اضافه می‌کنه.

- SADD tags "python"
- SADD tags "django"

```
SMEMBERS key
```

همه اعضای Set رو نشون میده.

- SMEMBERS tags # خروجی: {"python", "django"}

```
SISMEMBER key value
```

بررسی می‌کنه آیا مقدار وجود داره یا نه.

- SISMEMBER tags "python" # خروجی: 1 → یعنی وجود داره

🔹 5. Sorted Set Commands

مثل Set اما اعضا مرتب میشن بر اساس Score (خیلی عالی برای leaderboard).

```
ZADD key score value
```

مقدار جدید با امتیاز ذخیره میشه.

- ZADD scores 100 "Ali" 200 "Reza"

```
ZRANGE key start end WITHSCORES
```

اعضا رو به ترتیب از کم به زیاد نشون میده.

- ZRANGE scores 0 -1 WITHSCORES# خروجی: Ali 100 , Reza 200

```
ZREVRANGE key start end WITHSCORES
```

اعضا رو از زیاد به کم نشون میده.

- ZREVRANGE scores 0 -1 WITHSCORES# خروجی: Reza 200 , Ali 100

🔹 6. Pub/Sub Commands

انتشار/اشتراک پیام (Message Broker ساده).

```
SUBSCRIBE channel
```

مشترک یک کانال میشه.

- SUBSCRIBE news

```
PUBLISH channel message
```

پیام رو به کانال ارسال می‌کنه.

- PUBLISH news "Hello World!"

🔹 7. Key Management Commands

مدیریت کلیدها.

```
DEL key
```

کلید رو حذف می‌کنه.

- DEL name

```
EXISTS key
```

بررسی می‌کنه کلید وجود داره یا نه.

- EXISTS name

`EXPIRE key seconds`

زمان انقضا برای کلید می‌ذاره.

- SET token abc123
- EXPIRE token 60 # بعد از 60 ثانیه حذف میشه

`TTL key`

زمان باقی‌مانده‌ی عمر کلید رو نشون میده.

- TTL token

commands

📘 آموزش کامل Redis Commands (صفر تا صد)

1. مقدمه

Redis یک پایگاه داده **In-Memory Key-Value** است که داده‌ها را در RAM ذخیره می‌کند و به همین دلیل بسیار سریع است.
Redis نه‌تنها برای **Cache** بلکه برای مدیریت **Leaderboard**، **Queue**، **Session** و حتی **Message Broker** استفاده می‌شود.

---

2. ساختار کلی دستورات

دستورات Redis معمولاً این قالب را دارند:

```
COMMAND key [value] [options]
```

مثال ساده:

```
SET name "Parham"
GET name
```

---

3. مدیریت کلیدها (Key Management)

دستورات پایه برای کار با کلیدها:

- `DEL key` → حذف کلید
- `EXISTS key` → بررسی وجود کلید
- `EXPIRE key seconds` → تعیین زمان انقضا (TTL)
- `TTL key` → زمان باقی‌مانده عمر کلید
- `KEYS pattern` → پیدا کردن کلیدها بر اساس الگو
- `RENAME oldkey newkey` → تغییر نام کلید

---

4. داده‌های String

ساده‌ترین نوع داده در Redis.

- `SET key value` → ذخیره مقدار
- `GET key` → دریافت مقدار
- `MSET key1 value1 key2 value2` → ذخیره چند کلید همزمان
- `MGET key1 key2` → دریافت چند مقدار همزمان
- `SETEX key seconds value` → ذخیره مقدار با انقضا
- `GETSET key value` → مقدار جدید ذخیره و مقدار قبلی برمی‌گردد
- `INCR key` → افزایش عدد
- `DECR key` → کاهش عدد
- `APPEND key value` → اضافه کردن متن به انتهای مقدار

---

5. داده‌های Hash

مثل دیکشنری یا JSON.

- `HSET key field value` → ذخیره فیلد
- `HGET key field` → دریافت مقدار یک فیلد
- `HMSET key field1 value1 field2 value2` → ذخیره چند فیلد
- `HMGET key field1 field2` → دریافت چند فیلد
- `HGETALL key` → دریافت همه فیلدها
- `HDEL key field` → حذف فیلد
- `HLEN key` → تعداد فیلدها

---

6. داده‌های List

لیست پیوسته، مناسب صف (Queue) یا پشته (Stack).

- `LPUSH key value` → اضافه به ابتدای لیست
- `RPUSH key value` → اضافه به انتهای لیست
- `LPOP key` → حذف و برگرداندن عنصر اول
- `RPOP key` → حذف و برگرداندن عنصر آخر
- `LRANGE key start end` → مشاهده بخشی از لیست
- `LLEN key` → طول لیست
- `LREM key count value` → حذف مقدار مشخص از لیست

---

7. داده‌های Set

مجموعه بدون عنصر تکراری.

- `SADD key value` → اضافه کردن عضو
- `SMEMBERS key` → مشاهده همه اعضا
- `SISMEMBER key value` → بررسی وجود عضو
- `SCARD key` → تعداد اعضا
- `SREM key value` → حذف عضو
- `SUNION set1 set2` → اجتماع دو مجموعه
- `SINTER set1 set2` → اشتراک دو مجموعه
- `SDIFF set1 set2` → اختلاف دو مجموعه

---

8. داده‌های Sorted Set

مثل Set اما با Score مرتب می‌شود (مناسب برای leaderboard).

- `ZADD key score member` → اضافه کردن عضو با امتیاز
- `ZRANGE key start end WITHSCORES` → نمایش اعضا (از کم به زیاد)
- `ZREVRANGE key start end WITHSCORES` → نمایش اعضا (از زیاد به کم)
- `ZSCORE key member` → امتیاز یک عضو
- `ZREM key member` → حذف عضو
- `ZCARD key` → تعداد اعضا
- `ZRANK key member` → رتبه یک عضو

---

9. داده‌های Bitmap

ذخیره داده‌های باینری و کار با بیت‌ها.

- `SETBIT key offset value` → تغییر یک بیت
- `GETBIT key offset` → گرفتن یک بیت
- `BITCOUNT key` → شمارش تعداد بیت‌های ۱

---

10. داده‌های HyperLogLog

ساختار برای شمارش تقریبی اعضای یونیک.

- `PFADD key element` → اضافه کردن عنصر
- `PFCOUNT key` → شمارش تقریبی تعداد اعضای یونیک
- `PFMERGE destkey key1 key2` → ادغام چند HyperLogLog

---

11. داده‌های Stream

ذخیره جریان داده (event streaming).

- `XADD stream * field value` → اضافه کردن رویداد
- `XRANGE stream start end` → دریافت رویدادها
- `XREAD COUNT n STREAMS streamID` → خواندن n رویداد جدید

---

12. Pub/Sub (انتشار/اشتراک پیام)

برای ارتباط بین سرویس‌ها.

- `SUBSCRIBE channel` → مشترک شدن در کانال
- `PUBLISH channel message` → ارسال پیام به کانال
- `UNSUBSCRIBE channel` → لغو اشتراک

---

13. مدیریت سرور و Connection

- `PING` → تست اتصال
- `AUTH password` → احراز هویت
- `SELECT db` → انتخاب دیتابیس (پیش‌فرض db0)
- `FLUSHDB` → پاک کردن دیتابیس فعلی
- `FLUSHALL` → پاک کردن همه دیتابیس‌ها
- `INFO` → نمایش اطلاعات سرور

---

14. نکات مهم

- Redis داده‌ها را در RAM ذخیره می‌کند → مراقب حجم داده باش.
- همیشه برای داده‌های موقت از `EXPIRE` یا `SETEX` استفاده کن.
- در معماری Microservices می‌توان از Redis به‌عنوان Cache، Queue یا Message Broker استفاده کرد.
- Redis تنها یک پایگاه داده نیست، بلکه یک ابزار چندمنظوره برای بهبود عملکرد سیستم‌هاست.

🐍 django_redis

👩‍💻 آموزش صفر تا صد Redis در Django

1️⃣ Redis چیست؟

- **Redis** = مخفف Remote Dictionary Server
- یک دیتابیس **In-Memory** (در حافظه) خیلی سریع است.
- کاربردها:
  - **کش (Caching)** → سرعت دادن به کوئری‌ها و صفحات
  - **مدیریت سشن‌ها** → ذخیره‌سازی نشست کاربران به‌جای DB یا فایل
  - **صف کارها (Task Queue)** → اجرای کارهای پس‌زمینه (با Celery یا RQ)
  - **ارتباط (Pub/Sub (real-time** → نوتیفیکیشن‌ها، چت و ...

2️⃣ نصب Redis

🔹 روی لینوکس (Ubuntu/Debian)

```
sudo apt update
sudo apt install redis-server
sudo systemctl enable redis
sudo systemctl start redis
```

🔹 روی macOS

```
brew install redis
brew services start redis
```

🔹 روی ویندوز

بهترین روش: استفاده از **Docker**

```
docker run -d --name redis -p 6379:6379 redis:latest
```

3️⃣ نصب کلاینت پایتون

داخل محیط مجازی پروژه Django:

```
pip install redis
```

4️⃣ اتصال Django به Redis

روش پیشنهادی: django-redis

```
pip install django-redis
```

اضافه کردن به `settings.py`:

```
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",  # 1 دیتابیس شماره
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
        }
    }
}
```

👉 تست کش:

```
from django.core.cache import cache

cache.set("hello", "world", timeout=30)
print(cache.get("hello"))  # خروجی: "world"
```

5️⃣ استفاده از Redis برای سشن‌ها

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
SESSION_CACHE_ALIAS = "default"
```

👉 از این به بعد سشن کاربران توی Redis ذخیره میشه.

6️⃣ کش کردن ویوها

مثال:

```
from django.views.decorators.cache import cache_page
from django.http import HttpResponse

@cache_page(60 * 15)  # کش برای ۱۵ دقیقه
def my_view(request):
    return HttpResponse("سلام از Redis Cache!")
```

7️⃣ کش کردن کوئری‌ها

```
from django.core.cache import cache
from myapp.models import Product

def get_products():
    products = cache.get("all_products")
    if not products:
        products = list(Product.objects.all())
        cache.set("all_products", products, 60*10)  # ۱۰ دقیقه
    return products
```

8️⃣ صف کارها با Celery + Redis

نصب:

```
pip install celery[redis]
```

فایل `proj/celery.py`:

```
import os
from celery import Celery

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'proj.settings')

app = Celery('proj')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()

app.conf.broker_url = "redis://127.0.0.1:6379/0"
```

تعریف یک تسک (`tasks.py`):

```
from celery import shared_task

@shared_task
def add(x, y):
    return x + y
```

👉 اجرای Worker:

```
celery -A proj worker -l info
```

👉 فراخوانی تسک:

```
from myapp.tasks import add
add.delay(2, 3)  # اجرا در پس‌زمینه
```

9️⃣ ارتباط Real-time با Django Channels (Redis)

- نصب:

```
pip install channels channels_redis
```

- تنظیمات در `settings.py`:

```
ASGI_APPLICATION = "proj.asgi.application"

CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("127.0.0.1", 6379)],
        },
    },
}
```

👉 حالا می‌تونید چت، نوتیفیکیشن و اپ‌های بلادرنگ بسازید.

🔟 مانیتور Redis

- از طریق خط فرمان:

```
redis-cli monitor
```

- ابزار گرافیکی: RedisInsight

🎯 جمع‌بندی

با Redis در Django می‌توانید:

- ✅ کش برای دیتابیس و ویوها داشته باشید
- ✅ سشن‌ها را ذخیره کنید
- ✅ صف کارهای پس‌زمینه با Celery بسازید
- ✅ اپلیکیشن real-time با Channels راه بیندازید

می‌خوای من برات یک **پروژه آماده Django + Redis (با کش، Channels، Celery)** بسازم که مستقیم ران کنی و ببینی؟