Setting up server with Django

# Introduction

This guide provides step-by-step instructions to set up a Django-based server for serving ticket data securely, using different authentication methods and formats. Choosing the right method depends on your project's requirements. Here's an overview to help you decide:

## CSV vs. JSON

- \*\*CSV\*\*: Ideal for projects where data is exported for offline analysis or integration with external systems (e.g., spreadsheets or accounting software).  
- \*\*JSON\*\*: Best suited for real-time applications or APIs that integrate with modern web or mobile apps. JSON is lightweight, structured, and easily parsed by applications.

## Authentication Methods

1. \*\*Bearer Token Authentication\*\*  
 - \*\*Pros\*\*: Simple, effective for stateless APIs, and widely used in RESTful services.  
 - \*\*Use Case\*\*: Suitable for machine-to-machine communication or scenarios where the client can securely store the token (e.g., server-side applications).  
 - \*\*Setup Chapter\*\*: CSV with Bearer Token Authentication or JSON with Bearer Token Authentication.  
  
2. \*\*Username and Password Authentication\*\*  
 - \*\*Pros\*\*: More familiar to users and allows human-readable credentials. Easy to integrate into existing user authentication systems.  
 - \*\*Use Case\*\*: Ideal for applications requiring user authentication, such as admin dashboards or tools managed by multiple team members.  
 - \*\*Setup Chapter\*\*: CSV with Username and Password Authentication or JSON with Username and Password Authentication.

## When to Use?

- \*\*CSV with Bearer Token Authentication\*\*: Use for exporting secure, token-protected CSV files for system integrations or reporting tools.  
- \*\*CSV with Username and Password Authentication\*\*: Ideal for admin tools where users log in with credentials to download CSV data.  
- \*\*JSON with Bearer Token Authentication\*\*: Use for building stateless, token-secured APIs that serve structured ticket data to applications.  
- \*\*JSON with Username and Password Authentication\*\*: Best for scenarios requiring authenticated access to structured data, often used in admin or reporting applications.

This guide provides standalone chapters for each setup, so you can jump directly to the section that matches your needs.

Introduction 1

CSV vs. JSON

Authentication Methods

When to Use?

Guide to Setting Up a CSV File Server with Auth Token 4

1. Prerequisites
2. Configure the models.py
3. Add CSV Export Logic
4. Configure URLs
5. Test Your Setup

Guide to Setting Up a CSV File Server with Username and Password Authentication 7

1. Prerequisites
2. Configure the Models
3. Add Basic Authentication Logic
4. Create the CSV Export View
5. Configure URLs
6. Test the Setup

Guide to Setting Up a JSON Server with Bearer Token Authentication 10

1. Prerequisites
2. Configure the Models
3. Add Bearer Token Authentication Logic
4. Create the JSON Export View
5. Configure URLs
6. Test the Setup

Guide to Setting Up a JSON Server with Username and Password Authentication 13

1. Prerequisites
2. Configure the Models
3. Add Basic Authentication Logic
4. Create the JSON Export View

Configure URLs

Test the Setup

# Guide to Setting Up a CSV File Server with Auth Token

## Prerequisites

Before starting, ensure you have the following:  
1. Python 3.x installed on your system.  
2. A fresh Django project. If you don't have one, run:

django-admin startproject ticket\_server  
cd ticket\_server  
python manage.py startapp tickets

3. Add the `tickets` app to your Django project:  
 - Open `settings.py` in the `ticket\_server` folder.  
 - Add `'tickets'` to the `INSTALLED\_APPS` list.  
4. Install required libraries:

pip install django phonenumbers

## Configure the models.py

1. Navigate to the `tickets` app folder and open `models.py`.

2. Add the following code to define the `GlobalConfig` and `Ticket` models:

from django.db import models  
import uuid  
  
class GlobalConfig(models.Model):  
 name = models.CharField(max\_length=120, default="default")  
 auth\_code = models.CharField(  
 max\_length=120,  
 default="3fK8#Tg7lQp45aFz!nXw9YrMv2Dc%hJ0VrL6oK7aBnC5jXqRmSaW8tYpL3sN1vB"  
 )  
  
 def \_\_str\_\_(self):  
 return f"{self.name}"  
  
def generate\_ticket\_id():  
 return str(uuid.uuid4())  
  
class Ticket(models.Model):  
 new\_idd = models.UUIDField(primary\_key=True, default=generate\_ticket\_id, editable=False)  
 ticket\_id = models.CharField(max\_length=255)  
 times\_scanned = models.IntegerField(default=0)  
 description = models.TextField(blank=True, null=True)  
  
 def \_\_str\_\_(self):  
 return self.ticket\_id

3. Run migrations to create the database tables:

python manage.py makemigrations  
python manage.py migrate

## Add CSV Export Logic

1. Open `views.py` in the `tickets` app folder and add the following imports and helper functions:

from django.http import HttpResponse  
from .models import Ticket, GlobalConfig  
  
# Helper function to get the first alphabetical GlobalConfig  
def get\_global\_config():  
 return GlobalConfig.objects.order\_by('name').first()  
  
# Verify Bearer Token Authentication  
def verify\_bearer\_token(request):  
 config = get\_global\_config()  
 auth\_header = request.headers.get('Authorization', '')  
 token\_type, \_, token = auth\_header.partition(' ')  
 return token\_type == 'Bearer' and token == config.auth\_code

2. Add the CSV generation and token-protected view:

import csv  
from django.views.decorators.http import require\_http\_methods  
  
def show\_csv(request):  
 response = HttpResponse(content\_type='text/csv')  
 response['Content-Disposition'] = 'attachment; filename="tickets.csv"'  
  
 writer = csv.writer(response)  
 writer.writerow(['Ticket ID', 'Scanned', 'Description'])  
  
 for ticket in Ticket.objects.all():  
 writer.writerow([ticket.ticket\_id, ticket.times\_scanned, ticket.description])  
  
 return response  
  
@require\_http\_methods(["GET"])  
def bearer\_csv(request):  
 if not verify\_bearer\_token(request):  
 return HttpResponse("Unauthorized", status=401)  
 return show\_csv(request)

## Configure URLs

1. Create a `urls.py` file in the `tickets` app folder (if it doesn’t exist) and add the following code:

from django.urls import path  
from .views import bearer\_csv  
  
urlpatterns = [  
 path('csv/', bearer\_csv, name='bearer\_csv'),  
]

2. Link the app’s `urls.py` to the main project:

Open the `urls.py` file in the `ticket\_server` folder.

from django.urls import include, path  
  
urlpatterns = [  
 path('tickets/', include('tickets.urls')),  
]

## Test Your Setup

1. Run the server:

python manage.py runserver

2. Create a `GlobalConfig` entry in the Django admin panel:  
 - Run:

python manage.py createsuperuser

- Log in at `http://127.0.0.1:8000/admin/`.  
 - Add a new `GlobalConfig` object with your desired `auth\_code`.

3. Add some tickets to test the functionality:  
 - Either through the admin panel or directly in the shell:

python manage.py shell

from tickets.models import Ticket  
Ticket.objects.create(ticket\_id="123456", description="Test ticket”)

4. Test the token-protected endpoint:  
 - Use `curl` or a tool like Postman to make a request:

curl -H "Authorization: Bearer <your\_auth\_code>" http://127.0.0.1:8000/tickets/csv/

# Guide to Setting Up a CSV File Server with Username and Password Authentication

## Prerequisites

Before starting, ensure you have the following:  
1. Python 3.x installed on your system.  
2. A fresh Django project. If you don't have one, run:

django-admin startproject ticket\_server  
cd ticket\_server  
python manage.py startapp tickets

3. Add the `tickets` app to your Django project:  
 - Open `settings.py` in the `ticket\_server` folder.  
 - Add `'tickets'` to the `INSTALLED\_APPS` list.  
4. Install required libraries:

pip install django phonenumbers

## Configure the Models

1. Navigate to the `tickets` app folder and open `models.py`.

2. Add the following code to define the `GlobalConfig` and `Ticket` models:

from django.db import models  
import uuid  
  
class GlobalConfig(models.Model):  
 name = models.CharField(max\_length=120, default="default")  
 username = models.CharField(max\_length=120, default="example\_user")  
 password = models.CharField(max\_length=120, default="example\_password")  
  
 def \_\_str\_\_(self):  
 return f"{self.name}"  
  
def generate\_ticket\_id():  
 return str(uuid.uuid4())  
  
class Ticket(models.Model):  
 new\_idd = models.UUIDField(primary\_key=True, default=generate\_ticket\_id, editable=False)  
 ticket\_id = models.CharField(max\_length=255)  
 times\_scanned = models.IntegerField(default=0)  
 description = models.TextField(blank=True, null=True)  
  
 def \_\_str\_\_(self):  
 return self.ticket\_id

3. Apply the migrations:

python manage.py makemigrations  
python manage.py migrate

## Add Basic Authentication Logic

1. Open `views.py` in the `tickets` app folder and add the following imports and helper functions:

import base64  
from django.http import HttpResponse  
from .models import Ticket, GlobalConfig  
  
# Helper function to get the first alphabetical GlobalConfig  
def get\_global\_config():  
 return GlobalConfig.objects.order\_by('name').first()  
  
# Verify Basic Authentication  
def verify\_basic\_auth(request):  
 config = get\_global\_config()  
 auth\_header = request.headers.get('Authorization', '')  
 token\_type, \_, encoded\_credentials = auth\_header.partition(' ')  
  
 if token\_type != 'Basic' or not encoded\_credentials:  
 return False  
  
 decoded\_credentials = base64.b64decode(encoded\_credentials).decode()  
 username, \_, password = decoded\_credentials.partition(':')  
 return username == config.username and password == config.password

## Create the CSV Export View

1. Add a function to generate and serve the CSV file:

import csv  
from django.views.decorators.http import require\_http\_methods  
  
def show\_csv(request):  
 response = HttpResponse(content\_type='text/csv')  
 response['Content-Disposition'] = 'attachment; filename="tickets.csv"'  
  
 writer = csv.writer(response)  
 writer.writerow(['Ticket ID', 'Scanned', 'Description'])  
  
 for ticket in Ticket.objects.all():  
 writer.writerow([ticket.ticket\_id, ticket.times\_scanned, ticket.description])  
  
 return response

2. Add the Basic Authentication-protected view:

@require\_http\_methods(["GET"])  
def basic\_csv(request):  
 if not verify\_basic\_auth(request):  
 return HttpResponse("Unauthorized", status=401)  
 return show\_csv(request)

## Configure URLs

1. Create a `urls.py` file in the `tickets` app folder (if it doesn’t exist) and add the following code:

from django.urls import path  
from .views import basic\_csv  
  
urlpatterns = [  
 path('csv/basic/', basic\_csv, name='basic\_csv'),  
]

2. Link the app’s `urls.py` to the main project:

Open the `urls.py` file in the `ticket\_server` folder.

from django.urls import include, path  
  
urlpatterns = [  
 path('tickets/', include('tickets.urls')),  
]

## Test the Setup

1. Run the server:

python manage.py runserver

1. Create a `GlobalConfig` entry in the Django admin panel:  
    - Run:

python manage.py createsuperuser

- Log in at `http://127.0.0.1:8000/admin/`.  
 - Add a new `GlobalConfig` object with the desired username and password:  
 - `username`: `test\_user`  
 - `password`: `test\_password`  
3. Add some test tickets:

python manage.py shell

from tickets.models import Ticket

Ticket.objects.create(ticket\_id="123456", description="Test ticket”)

4. Test the Basic Authentication-protected endpoint:

curl -u test\_user:test\_password http://127.0.0.1:8000/tickets/csv/basic/

# Guide to Setting Up a JSON Server with Bearer Token Authentication

## Prerequisites

Before starting, ensure you have the following:  
1. Python 3.x installed on your system.  
2. A fresh Django project. If you don't have one, run:

django-admin startproject ticket\_server  
cd ticket\_server  
python manage.py startapp tickets

3. Add the `tickets` app to your Django project:  
 - Open `settings.py` in the `ticket\_server` folder.  
 - Add `'tickets'` to the `INSTALLED\_APPS` list.  
4. Install required libraries:

pip install django phonenumbers

## Configure the Models

1. Navigate to the `tickets` app folder and open `models.py`.

2. Add the following code to define the `GlobalConfig` and `Ticket` models:

from django.db import models  
import uuid  
  
class GlobalConfig(models.Model):  
 name = models.CharField(max\_length=120, default="default")  
 auth\_code = models.CharField(max\_length=120, default="example\_token")  
  
 def \_\_str\_\_(self):  
 return f"{self.name}"  
  
def generate\_ticket\_id():  
 return str(uuid.uuid4())  
  
class Ticket(models.Model):  
 new\_idd = models.UUIDField(primary\_key=True, default=generate\_ticket\_id, editable=False)  
 ticket\_id = models.CharField(max\_length=255)  
 times\_scanned = models.IntegerField(default=0)  
 description = models.TextField(blank=True, null=True)  
  
 def \_\_str\_\_(self):  
 return self.ticket\_id

3. Apply the migrations:

python manage.py makemigrations  
python manage.py migrate

## Add Bearer Token Authentication Logic

1. Open `views.py` in the `tickets` app folder and add the following imports and helper functions:

from django.http import JsonResponse  
from .models import Ticket, GlobalConfig  
  
# Helper function to get the first alphabetical GlobalConfig  
def get\_global\_config():  
 return GlobalConfig.objects.order\_by('name').first()  
  
# Verify Bearer Token Authentication  
def verify\_bearer\_token(request):  
 config = get\_global\_config()  
 auth\_header = request.headers.get('Authorization', '')  
 token\_type, \_, token = auth\_header.partition(' ')  
 return token\_type == 'Bearer' and token == config.auth\_code

## Create the JSON Export View

1. Add a view to generate and serve JSON data:

from django.views.decorators.http import require\_http\_methods  
  
@require\_http\_methods(["GET"])  
def bearer\_json(request):  
 if not verify\_bearer\_token(request):  
 return HttpResponse("Unauthorized", status=401)  
  
 tickets = Ticket.objects.values('ticket\_id', 'times\_scanned', 'description')  
 return JsonResponse(list(tickets), safe=False)

## Configure URLs

1. Create a `urls.py` file in the `tickets` app folder (if it doesn’t exist) and add the following code:

from django.urls import path  
from .views import bearer\_json  
  
urlpatterns = [  
 path('json/bearer/', bearer\_json, name='bearer\_json'),  
]

2. Link the app’s `urls.py` to the main project:

Open the `urls.py` file in the `ticket\_server` folder.

from django.urls import include, path  
  
urlpatterns = [  
 path('tickets/', include('tickets.urls')),  
]

## Test the Setup

1. Run the server:

python manage.py runserver

1. Create a `GlobalConfig` entry in the Django admin panel:  
    - Run:

python manage.py createsuperuser

1. - Log in at `http://127.0.0.1:8000/admin/`.  
    - Add a new `GlobalConfig` object with the desired `auth\_code`:  
    - `auth\_code`: `example\_token`
2. Add some test tickets:

python manage.py shell

from tickets.models import Ticket  
Ticket.objects.create(ticket\_id="123456", description="Test ticket”)

1. Test the Bearer Token-protected endpoint:

curl -H "Authorization: Bearer example\_token" http://127.0.0.1:8000/tickets/json/bearer/

# Guide to Setting Up a JSON Server with Username and Password Authentication

## Prerequisites

Before starting, ensure you have the following:  
1. Python 3.x installed on your system.  
2. A fresh Django project. If you don't have one, run:

django-admin startproject ticket\_server  
cd ticket\_server  
python manage.py startapp tickets

3. Add the `tickets` app to your Django project:  
 - Open `settings.py` in the `ticket\_server` folder.  
 - Add `'tickets'` to the `INSTALLED\_APPS` list.  
4. Install required libraries:

pip install django phonenumbers

## Configure the Models

1. Navigate to the `tickets` app folder and open `models.py`.

2. Add the following code to define the `GlobalConfig` and `Ticket` models:

from django.db import models  
import uuid  
  
class GlobalConfig(models.Model):  
 name = models.CharField(max\_length=120, default="default")  
 auth\_code = models.CharField(max\_length=120, default="example\_token")  
  
 def \_\_str\_\_(self):  
 return f"{self.name}"  
  
def generate\_ticket\_id():  
 return str(uuid.uuid4())  
  
class Ticket(models.Model):  
 new\_idd = models.UUIDField(primary\_key=True, default=generate\_ticket\_id, editable=False)  
 ticket\_id = models.CharField(max\_length=255)  
 times\_scanned = models.IntegerField(default=0)  
 description = models.TextField(blank=True, null=True)  
  
 def \_\_str\_\_(self):  
 return self.ticket\_id

3. Apply the migrations:

python manage.py makemigrations  
python manage.py migrate

## Add Basic Authentication Logic

1. Open `views.py` in the `tickets` app folder and add the following imports and helper functions:

import base64  
from django.http import JsonResponse  
from .models import Ticket, GlobalConfig  
  
# Helper function to get the first alphabetical GlobalConfig  
def get\_global\_config():  
 return GlobalConfig.objects.order\_by('name').first()  
  
# Verify Basic Authentication  
def verify\_basic\_auth(request):  
 config = get\_global\_config()  
 auth\_header = request.headers.get('Authorization', '')  
 token\_type, \_, encoded\_credentials = auth\_header.partition(' ')  
  
 if token\_type != 'Basic' or not encoded\_credentials:  
 return False  
  
 decoded\_credentials = base64.b64decode(encoded\_credentials).decode()  
 username, \_, password = decoded\_credentials.partition(':')  
 return username == config.username and password == config.password

## Create the JSON Export View

1. Add a view to generate and serve JSON data:

from django.views.decorators.http import require\_http\_methods  
  
@require\_http\_methods(["GET"])  
def basic\_json(request):  
 if not verify\_basic\_auth(request):  
 return HttpResponse("Unauthorized", status=401)  
  
 tickets = Ticket.objects.values('ticket\_id', 'times\_scanned', 'description')  
 return JsonResponse(list(tickets), safe=False)

## Configure URLs

1. Create a `urls.py` file in the `tickets` app folder (if it doesn’t exist) and add the following code:

from django.urls import path  
from .views import basic\_json  
  
urlpatterns = [  
 path('json/basic/', basic\_json, name='basic\_json'),  
]

2. Link the app’s `urls.py` to the main project:  
Open the `urls.py` file in the `ticket\_server` folder.

from django.urls import include, path  
  
urlpatterns = [  
 path('tickets/', include('tickets.urls')),  
]

## Test the Setup

1. Run the server:

python manage.py runserver

1. Create a `GlobalConfig` entry in the Django admin panel:  
    - Run:

python manage.py createsuperuser

1. - Log in at `http://127.0.0.1:8000/admin/`.  
    - Add a new `GlobalConfig` object with the desired username and password:  
    - `username`: `test\_user`  
    - `password`: `test\_password`
2. Add some test tickets:

python manage.py shell

from tickets.models import Ticket  
Ticket.objects.create(ticket\_id="123456", description="Test ticket”)

1. Test the Basic Authentication-protected endpoint:

curl -u test\_user:test\_password http://127.0.0.1:8000/tickets/json/basic/

You have successfully set up a JSON server protected by Basic Authentication. Users must provide valid credentials in the request header to access the data.