# Cloude based convert to django Explanation :-

# Django Project MVT:-

# Django on Cloud ( AWS / Azure / Google Cloud)

Model: -The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres)

View:- The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files

Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

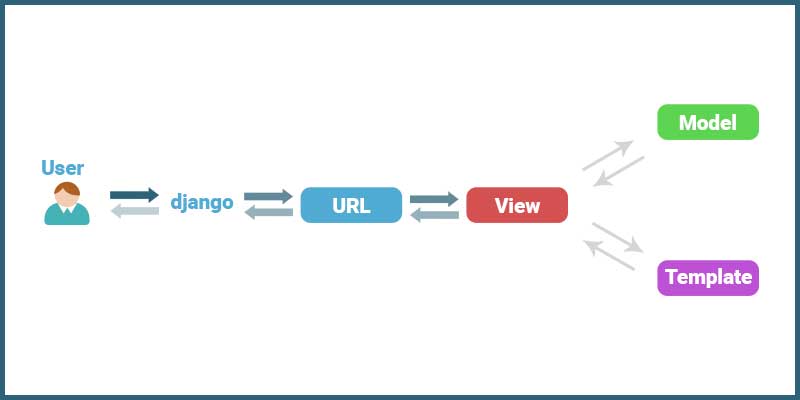
Project Structure :

A Django Project when initialized contains basic files by default such as manage.py, view.py, etc. A simple project structure is enough to create a single-page application. Here are the major files and their explanations. Inside the geeks\_site folder ( project folder )

# How to use Cloud Storage with Django Application:-

Connecting google storage to your Django application could sometimes be tricky Upload image to google cloud storage.

1. Getting a Public URI for our image from google.
2. You can also do all the functionality you have for a default storage.



# A picture containing graphical user interface Description automatically generated

**Implementation of using Cloud Django :-**

**Getting started with Django**

## Hosting platforms

|  |  |
| --- | --- |
| **Django deployment option** | **Get started** |
| **App Engine standard environment** | **** [**Running Django on App Engine standard environment**](https://cloud.google.com/python/django/appengine) |
| **App Engine flexible environment** | **** [**Running Django on App Engine flexible environment**](https://cloud.google.com/python/django/flexible-environment) |
| **Cloud Run** | **** [**Running Django on Cloud Run**](https://cloud.google.com/python/django/run)  **** [**Running Django on Cloud Run with Cloud Code for VS Code**](https://cloud.google.com/code/docs/vscode/quickstart-cloud-run)  **** [**Running Django on Cloud Run with Cloud Code for IntelliJ**](https://cloud.google.com/code/docs/intellij/quickstart-cloud-run) |
| **Google Kubernetes Engine (GKE)** | **** [**Running Django on Google Kubernetes Engine**](https://cloud.google.com/python/django/kubernetes-engine) |
| **Compute Engine** | **** [**Django in Google Cloud Marketplace**](https://cloud.google.com/marketplace/solution/bitnami-launchpad/djangostack?q=django) |

## Databases

The Django object-relational mapper (ORM) works best with an SQL relational database.

if you are starting a new project, [Cloud SQL](https://cloud.google.com/sql) is a good choice. You can deploy a [PostgreSQL](https://cloud.google.com/sql/docs/postgres/create-instance) or [MySQL](https://cloud.google.com/sql/docs/mysql/create-instance) database that's managed and scaled by Google, and supported by Django.

### [**Cross site request forgery (CSRF) protection**](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/web_application_security#cross_site_request_forgery_csrf_protection)**(data security)**

CSRF attacks allow a malicious user to execute actions using the credentials of another user without that user’s knowledge or consent. For example consider the case where we have a hacker who wants to create additional authors for our Local Library.

Django on Cloud Run

## **About this cloude run**

## [**1. Introduction**](https://codelabs.developers.google.com/codelabs/cloud-run-django#0)

[Cloud Run](https://cloud.google.com/run) is a managed compute platform that enables you to run stateless containers that are invocable via HTTP requests. Cloud Run is serverless: it abstracts away all infrastructure management, so you can focus on what matters most — building great applications.

It also natively interfaces with many other parts of the Google Cloud ecosystem, including [Cloud SQL](https://cloud.google.com/sql) for managed databases, [Cloud Storage](https://cloud.google.com/storage) for unified object storage, and [Secret Manager](https://cloud.google.com/secret-manager/) for managing secrets.

[Django](https://www.djangoproject.com/) is a high-level Python web framework.

In this tutorial, you will use these components to deploy a small Django project.

### **What you'll learn**

* How to use the Cloud Shell
* How to create a Cloud SQL database
* How to create a Cloud Storage bucket
* How to create Secret Manager secrets
* How to connect Google Cloud components to a Cloud Run service
* How to use the Google Container Registry
* How to deploy to Cloud Run
* How to run builds, migrations, and deployments in Cloud Build

## [**2. Setup and requirements**](https://codelabs.developers.google.com/codelabs/cloud-run-django#1)

## **Self-paced environment setup**

1. Sign in to [Cloud Console](http://console.cloud.google.com/) and create a new project or reuse an existing one. (If you don't already have a Gmail or Google Workspace account, you must [create one](https://accounts.google.com/SignUp).)

**Note:** You can easily access Cloud Console by memorizing its URL, which is console.cloud.google.com.



Application

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Remember the project ID, a unique name across all Google Cloud projects (the name above has already been taken and will not work for you, sorry!). It will be referred to later in this codelab as PROJECT\_ID.

**Note:** If you're using a Gmail account, you can leave the default location set to **No organization**. If you're using a Google Workspace account, then choose a location that makes sense for your organization.

1. Next, you'll need to [enable billing](https://console.cloud.google.com/billing) in Cloud Console in order to use Google Cloud resources.

## [**4. Create a template project**](https://codelabs.developers.google.com/codelabs/cloud-run-django#3)

You'll use the default Django project template as your sample Django project.

To create this template project, use Cloud Shell to create a new directory named django-cloudrun and navigate to it:

pip install Django

Save the list of packages installed to requirements.txt

pip freeze > requirements.txt

Then, create a new template project:

Django-admin startproject my project .

## [**5. Create the backing services**](https://codelabs.developers.google.com/codelabs/cloud-run-django#4)

You'll now create your backing services: a Cloud SQL database, a Cloud Storage bucket, and a number of Secret Manager values.

Securing the values of the passwords used in deployment is important to the security of any project, and ensures that no one accidentally puts passwords where they don't belong (for example, directly in settings files, or typed directly into your terminal where they could be retrieved from history.)

First, set two base environment variables, one for the project ID:

PROJECT\_ID=

When working with infrastructure, it's useful to have all your components in the same location, so they can talk to each other more efficiently. While some components are globally available, not all components are available in all regions.

Learn more:

* [Cloud Run (fully managed) locations](https://cloud.google.com/run/docs/locations#managed)
* [Cloud SQL locations](https://cloud.google.com/sql/docs/postgres/locations)
* [Cloud Storage Locations](https://cloud.google.com/storage/docs/locations)

## **Create the database**

Now, create a Cloud SQL instance:

## **Create the storage bucket**

## **Store configuration as secret**

Having set up the backing services, you'll now store these values in a file protected using Secret Manager.

Secret Manager allows you to store, manage, and access secrets as binary blobs or text strings. It works well for storing configuration information such as database passwords, API keys, or TLS certificates needed by an application at runtime.

## [**8. Run the migration steps**](https://codelabs.developers.google.com/codelabs/cloud-run-django#7)

To create the database schema in your Cloud SQL database and populate your Cloud Storage bucket with your static assets, you need to run migrate and collectstatic.

These base Django migration commands need to be run within the context of your built container with access to your database.

You will also need to run createsuperuser to create an administrator account to log into the Django admin.

## **Allow access to components:-**

For this step, we're going to use Cloud Build to run Django commands, so Cloud Build will need access to the Django configuration stored in Secret Manager.

As earlier, set the IAM policy to explicitly allow the Cloud Build service account access to the secret settings:

## [**9. Deploy to Cloud Run**](https://codelabs.developers.google.com/codelabs/cloud-run-django#8)

With the backing services created and populated, you can now create the Cloud Run service to access them.

The initial deployment of your containerized application to Cloud Run is created using the following command:

## [**10. Congratulations!**](https://codelabs.developers.google.com/codelabs/cloud-run-django#9)

You have just deployed a complex project to Cloud Run!

* Cloud Run automatically and horizontally scales your container image to handle the received requests, then scales down when demand decreases. You only pay for the CPU, memory, and networking consumed during request handling.
* Cloud SQL allows you to provision a managed PostgreSQL instance that is maintained automatically for you, and integrates natively into many Google Cloud systems.
* Cloud Storage lets you have cloud storage in a way that is accessible seamlessly in Django.
* Secret Manager allows you to store secrets, and have them accessible by certain parts of Google Cloud and not others.

### **Authentication:**

It handles user accounts, groups, cookie-based user sessions, etc.

### **Admin:**

It reads metadata in your models to provide a robust interface which can be used to manage content on your site.

### **Internationalization:**

Django provides support for translating text into various languages, locale-specific formatting of dates, times, numbers, and time zones.

### **Security:**

Django provides safeguard against the following attacks:

* Cross-Site Request Forgery (CSRF)
* Cross-site scripting
* SQL injection
* Clickjacking

Cloud Platform (GCP)Deploy Django securely on cloud i.e. [AWS marketplace](https://aws.amazon.com/marketplace/pp/B01MY92D6A?qid=1496236431400&sr=0-5&ref_=srh_res_product_title), [Azure](https://azuremarketplace.microsoft.com/en-us/marketplace/apps/cognosys.secured-django-on-ubuntu-14-04-lts) and [Google](https://secureanycloud.com/coming-soon/)