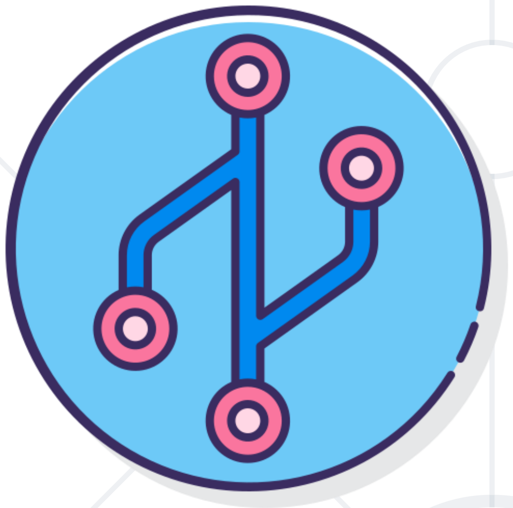
Deployment Setup



SoftUni Team

Technical Trainers

Software University

https://softuni.org

Have a Question?

sli.do

#python-web

Table of Contents

Git

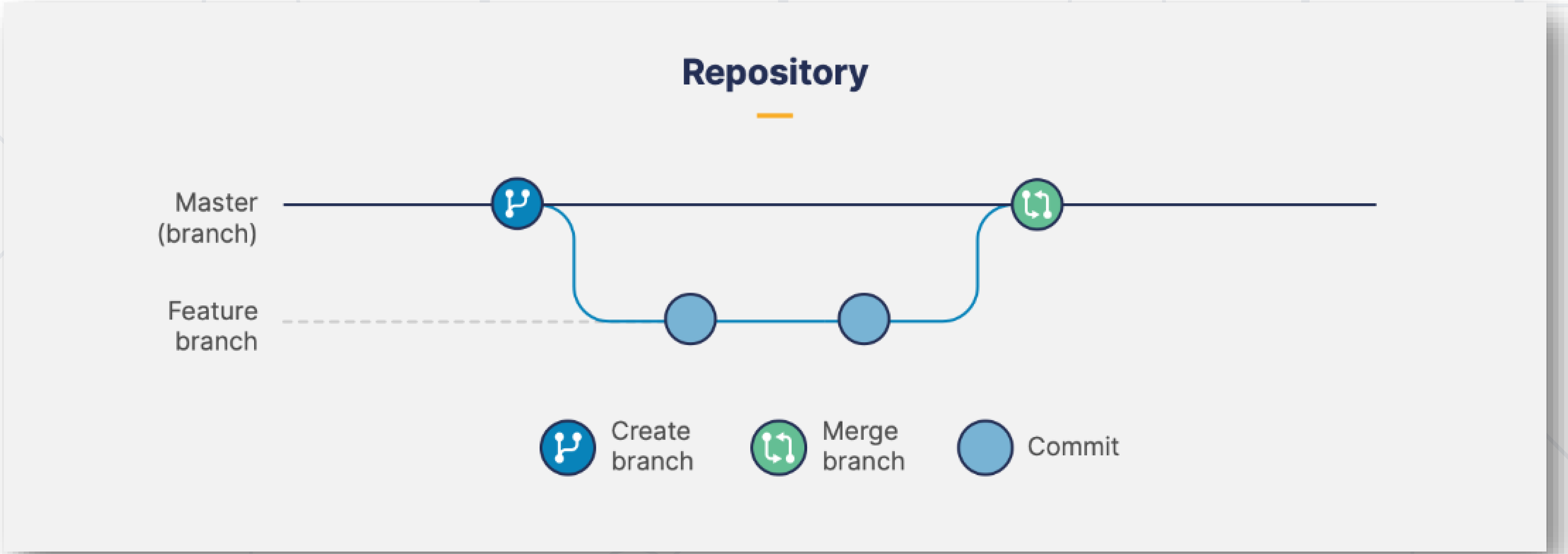
Distributed Version Control System

What is Git?

- **Git** is a **distributed version control system**, renowned as the **most widely** used **globally**
- It is **free** and **open-source**, allowing developers to **efficiently manage** their **codebases**
- **Git** operates with both **local** and **remote repositories**, facilitating **collaborative development workflows**

What is Git?

- **Version Control** and **Branching**
  - How **teams** work



What is Git?

- **Git** offers a **command-line interface** known as **Git Bash**
  - **Compatible** with various operating systems, including **Linux**, **macOS**, and **Windows** through **msysGit**
- For more information, visit the **official website** or explore **tutorials** at:
  - *https://git-scm.com*
  - *https://www.atlassian.com/git/tutorials/setting-up-a-repository*

Using Git

- **Console-based Client**
  - **Git** can be used via the **command line interface**, such as **Git Bash**, which provides a **powerful** and **flexible** way to **interact** with **Git repositories**
- Windows/Mac **GUI Client – SourceTree**
  - **SourceTree** is a **graphical user interface** (**GUI**) **client** for **Git**, available for **Windows** and **macOS**
  - It offers a **visually intuitive** way to **manage repositories** and **perform Git** operations
  - ***https://www.sourcetreeapp.com/***

Installing Git

- On **Windows**:
  - Install **Git** for **Windows** by downloading it from:
    - *https://git-scm.com/downloads*
  - During installation, **ensure** that the following **options** are **selected** (they are usually selected **by default**):
    - "**Use Git Bash Only**"
    - "**Checkout Windows-style, commit Unix-style endings**"

Installing Git

- On **Linux**:
  - Install **Git** using your distribution's **package manager**
  - On **Debian**/**Ubuntu**-based systems:

```
sudo apt install git
```

  - On **Fedora**/**CentOS**-based systems:

```
sudo yum install git
```

  - On **Arch**-based systems:

```
sudo pacman -S git
```

Basic Git Commands

- **Initializing** a **new** Git repository in the **current** directory

```
git init
```

- **Creating** a **remote reference** for a Git repository (assign a **short name** to a **remote** repository URL)

```
git remote add [remote name] [remote url]
```

- **Cloning** an **existing** Git repository onto your **local** machine

```
git clone [remote url]
```

Basic Git Commands

- **Fetching** and **merging** the latest changes from the **remote** repository

```
git pull
```

- **Checking** the status of your **local repository** (viewing local changes)

```
git status
```

Basic Git Commands

- **Adding** **specific** **files** to the **staging area**, **preparing** them to be included in the **next** **commit**

```
git add [filename]
```

- **Adding** **everything** in the **current** directory

```
git add .
```

- **Adding** **all changes** in the **current** directory and its **subdirectories** to the **staging area**

```
git add -A
```

Basic Git Commands

- **Committing** the **changes** staged in the **staging area** to the **local** repository, creating a **new commit** with a descriptive **message**

```
git commit -m "[your message here]"
```

- **Sending commits** to a **remote** repository by specifying the **remote** repository's **name** (e.g., "**origin**") and the **branch** name to **push** (e.g., "**master**")

```
git push [remote name] [local name]
```

GitHub

World's Largest Source Code Hosting Platform

What is GitHub?

- ***GitHub*** is one of the **most popular source code hosting platforms** globally
  - Widely used by **developers** and **organizations** for **collaborating** on software projects
  - Offers **free plans** for both **public** and **private repositories** for individual users and teams
  - Allows individuals and organizations to **keep** their code **private** and **restrict access** to authorized users

What is GitHub?

- **GitHub Services**:
  - Git Source Code **Repository**
  - **Issue** Tracker
  - Project **Board** (Kanban Style)
  - Wiki **Pages** (Documentation)

Using GitHub

- **GitHub Desktop Client**
  - **GitHub Desktop** is a **GUI client** specifically designed for **GitHub users**
  - It provides a **streamlined interface** for **managing repositories hosted** on **GitHub** and **simplifies** common **Git workflows**
  - https://desktop.github.com

Gunicorn

What is Gunicorn?

- **Gunicorn** stands for "**Green Unicorn**"
- **Gunicorn** is a popular **WSGI** (**Web Server Gateway Interface**) **HTTP server** for running Python web applications
- It is commonly used to **deploy** **Django**, **Flask**, and other **Python-based** web frameworks

Gunicorn Advantages

- A **lightweight** Python **HTTP server** that's designed to be **fast**, **reliable**, and **easy** to use
- It follows the **WSGI specification**, which allows it to **communicate** with **web applications** built using **Python web** frameworks

Gunicorn Advantages

- **Gunicorn** is **pre-forked**, meaning it can handle **multiple connections concurrently** without consuming too much memory or CPU resources
- It's commonly used in **production environments** to **serve Python web** applications, providing a **stable** and **efficient platform** for handling **web traffic**

How to Use Gunicorn

- **Install Gunicorn**
  - **Gunicorn** can be installed using **pip**
  - You can install it **globally** or within a **virtual** environment

```
pip install gunicorn
```

- **Run Gunicorn**
  - **Navigate** to the directory containing your application's **main module**

```
gunicorn [app_name].wsgi:application
```

Gunicorn Configurations

- **Gunicorn** provides **various configuration options** that can be specified using **command-line arguments** or a **configuration file**
- This allows you to **customize settings** such as the **number** of **worker processes**, **bind address**, and **logging** behavior

```
gunicorn [app_name].wsgi:application --workers=4 --
bind=0.0.0.0:8000
```

Integration with Web Servers

- **Gunicorn** is often used **behind** a **reverse proxy server** like **Nginx** or **Apache**
- The **web server forwards incoming HTTP requests** to **Gunicorn**, which then **processes** them and **returns responses**
- This setup allows for **better performance**, **security**, and **scalability** of **web applications**

Nginx

What is Nginx?

- NGINX is a powerful web server **software** that can also function as a reverse proxy, load balancer, and HTTP cache
- It's designed to efficiently **serve** static **content**, handle **high volumes** of concurrent **connections**, and effectively **distribute incoming** requests to backend servers

What is Nginx?

- NGINX is commonly used as a frontend server to serve **web applications** and **websites**, acting as a gateway between **clients** and **backend** application servers
- It's **widely** used by large-scale websites, content delivery networks (**CDNs**), and cloud hosting providers due to its performance, reliability, and flexibility

How to Use Nginx

- **Install NGINX**
  - **NGINX** can be installed on **various operating systems**
    - Using **package managers**
    - Or by **downloading** and **compiling** the source code
    - On **Ubuntu**/**Debian**, you can install NGINX using:

```
sudo apt update
sudo apt install nginx
```

- **Start NGINX**

```
sudo systemctl start nginx
```

How to Use Nginx on Windows

- You can install **NGINX** on **Windows**, although it's **less common** compared to **Unix-based** operating systems like **Linux**
  - [*https://nginx.org/en/docs/windows.html*](https://nginx.org/en/docs/windows.html)
- NGINX provides official **pre-built Windows binaries**, making it relatively **straightforward** to **install** and **run** on **Windows servers** or **development environments**

How to Use Nginx on Windows

- **Important** Notes:
  - **NGINX** is **primarily designed** for **Unix-based** systems and **may** have **limitations** or **differences** in **behavior** on **Windows**
  - **NGINX** on **Windows** is often used for **development** or **testing** **purposes** rather than **production** deployments

Nginx Configurations

- **Configure NGINX**
  - NGINX's **configuration file** is typically located at
    - `/etc/nginx/nginx.conf`
  - You can **edit** this file to **configure** NGINX's **behavior**, such as
    - Specifying **server blocks** (**virtual hosts**)
    - Setting up **SSL**/**TLS** **certificates**
    - Defining **proxy passes**
    - Configuring **caching**

Nginx Configuration File - Simple Example

```
# Set the user and group that NGINX will run as
user nginx;
worker_processes auto;

# Define the location of the NGINX error log
error_log /var/log/nginx/error.log;

# Define the location of the NGINX access log
access_log /var/log/nginx/access.log;

# Define the location of the NGINX PID file
pid /var/run/nginx.pid;

# Configure HTTP server
http {
    # Define server block for HTTP requests
    server {
        # Listen on port 80 for HTTP requests
        listen 80;

        # Define the server name (hostname or IP address)
        server_name example.com www.example.com;
...
```

Nginx Configuration File - Simple Example

```
...
        # Define the location of the static files
        location /static/ {
            alias /path/to/static/files/;
        }

        # Define the location of the media files
        location /media/ {
            alias /path/to/media/files/;
        }

        # Proxy requests to Gunicorn
        location / {
            proxy_pass http://127.0.0.1:8000;  # Adjust the Gunicorn address/port
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

Nginx Configurations

- **Test Configuration**
  - After making **changes** to the **NGINX configuration**, you should **test** the **configuration** for syntax errors using:

```
sudo nginx -t
```

  - If the configuration **test** is **successful**, **reload NGINX** to apply the changes:

```
sudo systemctl reload nginx
```

Integration with Web Applications

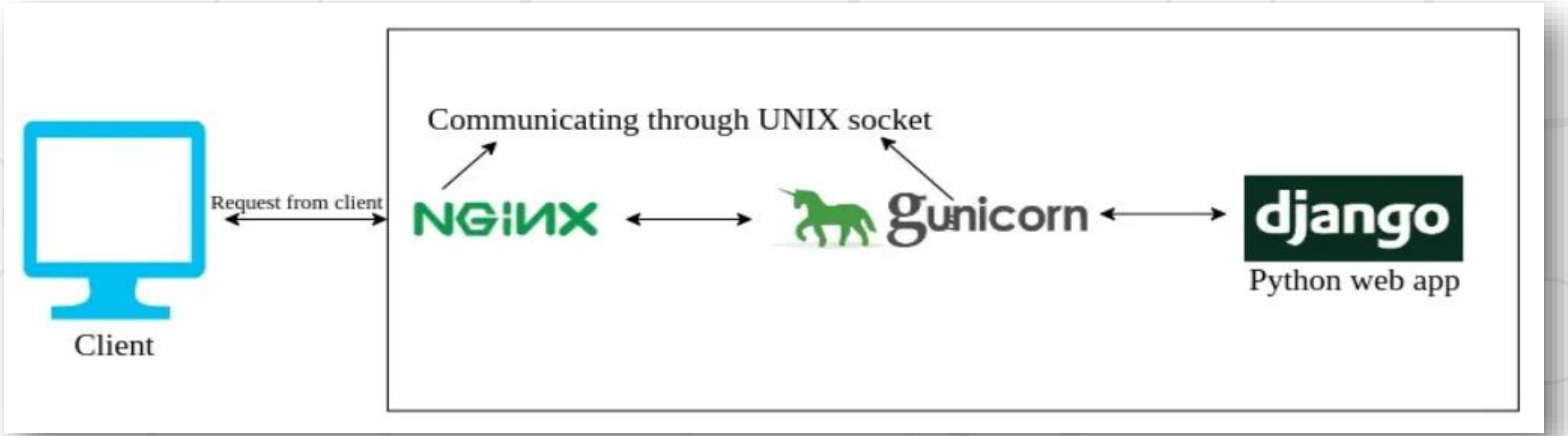
- **NGINX** is often used in **conjunction** with application servers like **Gunicorn** or **uWSGI**
- **NGINX** acts as a **reverse proxy**, forwarding **incoming HTTP requests** to the **application server**, which then **processes** the **requests** and **returns responses** to **NGINX** for delivery to **clients**

Integration with Web Applications

- **Serve Static Files**
  - **NGINX** is also **highly efficient** at **serving static files**
    - **HTML**, **CSS**, **JavaScript**, and **images**
  - You can **configure NGINX** to **serve static content** directly from disk
    - Improving **performance** and **reducing** the **load** on **backend application servers**

Django, Gunicorn, and Nginx

- **Optimizing Django Deployments**
  - Harnessing the **Power** of **NGINX** and **Gunicorn**



Django, Gunicorn, and Nginx

- **Nginx** and **Gunicorn** are **powerful combinations** to **run** your **Django application** and handle **high-traffic** websites
- To **set up** a Django application to **run behind Nginx** and **Gunicorn**
  - You need to **configure Nginx** as a **reverse proxy** for **Gunicorn**

Django, Gunicorn, and Nginx

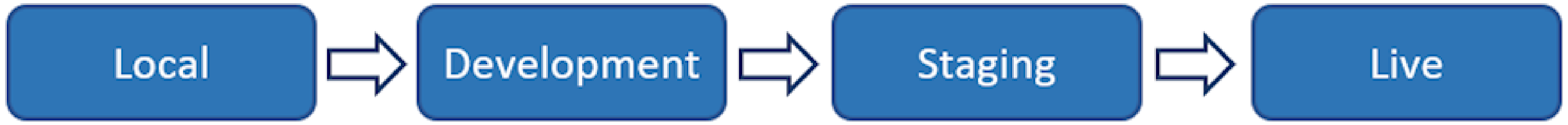
- When used **together**
  - **Nginx** can **handle** the **tasks** of **serving static** files and **managing SSL** certificates
  - **Gunicorn** can **handle** the **dynamic content** generated by the **Django application**
  - **Nginx** can also be used to **load balance** the **requests** between **multiple Gunicorn worker processes**

Deployment Setup

Prepare a Django Project for Deployment

What is Deployment?

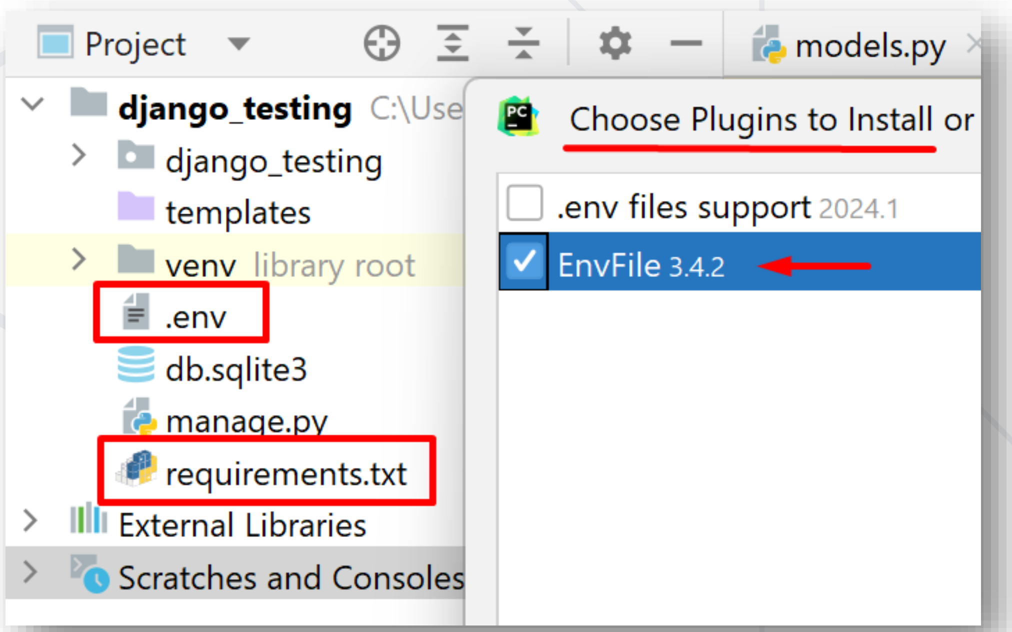
- **Deployment** involves **transferring changes** or **updates** from **one environment**, such as a **development** or **staging environment**, to **another**, typically a **production environment**



Represents the progression of changes through different environments in the deployment process

Prepare for Deployment

- **Before deploying** your Django project, there are several **important steps** to follow
  - Create **Configuration** Files:
    - In the outer project folder, create two new files:
      - `.env`
      - `requirements.txt`



Prepare for Deployment

- **`requirements.txt`** file
  - This file **lists** all Python **dependencies required** for your Django project to **run**
  - It's **essential** for ensuring **consistent environments** across **development**, **testing**, and **production**
  - You can **generate** it using the following command:

```
pip freeze > requirements.txt
```

Prepare for Deployment

- **.env** file
  - This file will contain **environment variables** and **sensitive information** such as **database credentials**, **API keys**, and **secret keys**
  - Make sure **not** to **commit** this file to **version control** for **security reasons**

```
# .env file
SECRET_KEY=your_secret_key_here
...
```

Prepare for Deployment

- **Settings** Configuration
  - Verify that your Django **settings** are **correctly configured** for **production**, including **security-related settings** such as **`DEBUG`**, **`SECRET_KEY`**, **`ALLOWED_HOSTS`**, etc.
- **Database** Configuration
  - Check if your **database settings** are **configured** properly for **production** use, such as ensuring that the **database engine** is **compatible** and that the **connection settings** are **correct**

Prepare for Deployment

- **Static** and **Media Files** Configuration
  - Ensure that your **static** and **media** **files configurations** are **appropriate** for **deployment**, including settings related to **file storage**, **serving** **static files**, and **handling** **media uploads**
- **Security** Considerations
  - Check for common **security vulnerabilities** and **misconfigurations** that could **compromise** the **security** of your application when deployed to a **production** **environment**

Prepare for Deployment

```
py manage.py check --deploy
```

- This command performs a **series** of **checks** to ensure your project is **ready** for **deployment** to a **production environment**
- When you **run** it, Django **checks** various aspects of your project **configuration** and code to **identify potential issues** that could cause problems in a **production environment**

More at: *https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/*

Logging, Monitoring, Error Handling

Logging

- **Logging** involves recording information about the application's **runtime behavior**, including **errors**, **warnings**, and **informational** messages
- Django provides **built-in support** for **logging**, allowing developers to **configure** **logging settings** in the project's **settings** file (`settings.py`)

A Basic Logging Configuration

```
import logging
import os

# Define logging configuration
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        # Define console handler for printing log messages to console
        'console': {
            'level': 'DEBUG',  # Adjust log level as needed (DEBUG, INFO, WARNING, ERROR, CRITICAL)
            'class': 'logging.StreamHandler',
        },
        # Define file handler for logging messages to a file
        'file': {
            'level': 'INFO',  # Adjust log level as needed
            'class': 'logging.FileHandler',
            'filename': os.path.join(BASE_DIR, 'logs', 'django.log'),  # Specify log file location
        },
    },
...  # Continues on the next page
```

A Basic Logging Configuration

```
...
    'loggers': {
        # Root logger
        '': {
            'handlers': ['console', 'file'],  # Specify which handlers to use
            'level': 'DEBUG',  # Adjust log level as needed
            'propagate': True,
        },
        # Django logger
        'django': {
            'handlers': ['console', 'file'],
            'level': 'DEBUG',  # Adjust log level as needed
            'propagate': False,
        },
        # Project-specific logger (optional)
        'myproject': {
            'handlers': ['console', 'file'],
            'level': 'DEBUG',  # Adjust log level as needed
            'propagate': False,
        },
    },
}
```

Logging Levels, Handlers, Formatters

- Logging **levels** (e.g., **DEBUG**, **INFO**, **WARNING**, **ERROR**, **CRITICAL**) help **categorize** log **messages** based on their **severity**
- Developers can **customize logging behavior** by
  - specifying **handlers** (e.g., **file handler**, **email handler**)
  - specifying **formatters** to define how log messages are **processed** and **displayed**

Monitoring

- **Monitoring** involves **observing** and **measuring** various aspects of the application's **performance** and **health** to ensure it meets defined **criteria** and service-level **objectives**
- **Monitoring tools** and **techniques** help **identify performance bottlenecks**, detect **errors** and **anomalies**, and **track** system **metrics** (e.g., CPU usage, memory usage, response times)

Error Handling

- **Error handling** refers to the process of **identifying**, **reporting**, and **responding** to **errors** that occur during the execution of the application
- Django provides **mechanisms** for **handling errors**, such as **raising exceptions**, using Django's **built-in error views**, and **customizing error-handling behavior** using **middleware** and **decorators**

Error Handling

- **Effective error handling** involves providing
  - **Informative error messages** to users
  - **Logging** detailed **information** about **errors** for **debugging** purposes
  - Implementing **strategies** for **gracefully recovering** from **errors** to **maintain** application **availability**

Sentry

- Sentry is an **error monitoring platform** that **captures** and **aggregates error logs, stack traces**, and other **diagnostic information** to help you **identify** and **debug issues** in your application
  - *https://docs.sentry.io/platforms/python/*

Deployment Platforms

Azure, AWS, Heroku

Where to Deploy a Python Project?

- You can **deploy** a **single project** to **multiple hosting** platforms
- Some of the most **popular deployment platforms**:
  - *Azure*
  - *Amazon Web Services (AWS)*
  - *Heroku*
  - *PythonAnywhere*

Deployment Platforms

- **Azure**, **AWS**, **Heroku**, and **PythonAnywhere** are **cloud computing platforms** that offer **services** for **deploying** and **hosting applications**
- They provide a range of **features** and **services** to **facilitate** the **deployment**, **scaling**, and **management** of applications, including **support** for various programming **languages**, **databases**, and development **frameworks** like **Django**

Comprehensive Deployment Guides

Software University

- **Deploy** a Django App with **App Service** and **Azure** Database for PostgreSQL - **Flexible Server**
  - ***Link to tutorial***
- **Deploy** a Django Web App with **Nginx** to **AWS** - **EC2**
  - ***Link to tutorial***
- **Deploy** a Django Web App Using **Heroku**
  - ***Link to tutorial***

Summary

- **Git** - A Version Control System
- **GitHub** - A Source Code Hosting Platform
- **Gunicorn** - Python WSGI HTTP Server
- **Nginx** - Open-Source Web Server
- **Cloud Computing** Platforms
  - **Azure, AWS, Heroku**

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank
Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR
Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License