**Authorization** – what you can do (users access permissions on a computer or network)  
**Authentication** – Who you are (confirming the identity of a user or computer). Verify that a user or application is indeed the entity it claims to be (systems authenticates by checking the **password**)  
**Identification** – the capability to recognize a user of a system or app (**username** to identify the user uniquely).

Authentication work by comparing user’s **credentials** to those in the authorized user db  
Authentication factors:  
- **single factor authentication**: username & password  
- **multi-factor authentication**: knowledge(password) combined with **biometric** or **possession** factor  
  
Authentication in Django:  
- pre-included in the **settings.py** file under the INSTALLED\_APPS setting: **'django.contrib.auth'**  
Includes:

* Users, groups and permissions
* A configurable password hashing system
* Allows inheritance from its URLs, models, views, and forms
* A pluggable backend system
* Cookie-based user session handling

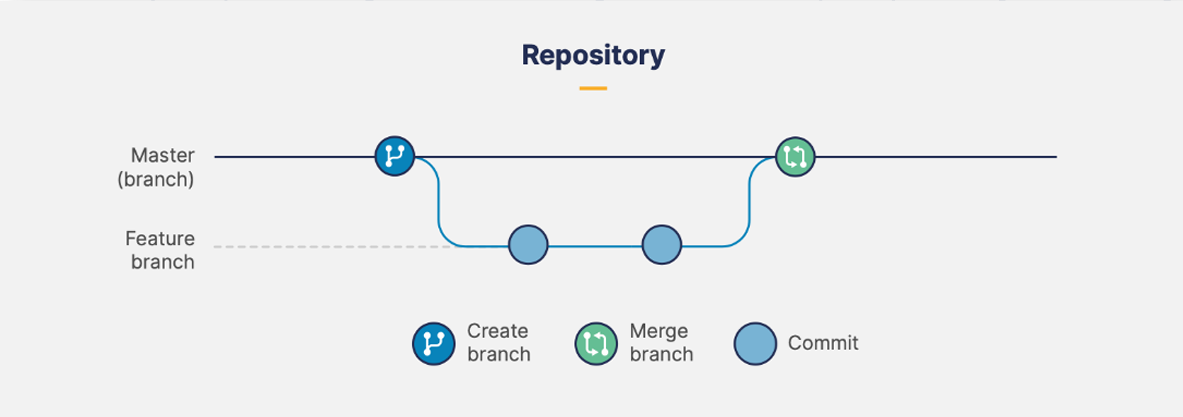
**cookies** - small pieces of data stored on a user's device by their web browser- used for session management  
- tracking and analysis  
- personalization  
- targeted advertising

**Cookie-based authentication**- cookies contain **Session ID**  
- enables the storage and retrieval of data on a per-site-visitor basis  
- data is stored on the server side, and the mechanism abstracts the process of sending and receiving cookies  
  
- SessionMiddleware manages sessions across requests  
- AuthenticationMiddleware associates users with requests by utilizing sessions

Permissions (Authorization)  
- Django offers built-in permissions  
- based on add, change, delete, view (automatically created for each model in the installed apps)

Most Common Web Security Problems:

* SQL Injection
* Cross-site Scripting (XSS)
* URL/HTTP manipulation attacks (Parameter Tampering)
* Cross-site Request Forgery (CSRF)
* Brute Force Attacks (also DDoS)
* Insufficient Access Control
* Missing SSL (HTTPS) / MITM Dev
* Phishing/Social Engineering

**Git**  
- distributed version control system  
version control – history of actions and provides ability to   
- local app, operates with both local and remote repositories  
- facilitates collaborative development workflows  
- offers a command-line interface known as **Git Bash**  
  
Branching:  
  
  
**Git commands**

**git init** - initialize a new Git repository in the current directory  
**git remote add** [remote name] [remote url] - create a remote reference for a Git repository  
**git clone** [remote url] - clone an existing Git repository onto local machine  
**git pull** - fetch and merge the latest changes from the remote repository  
**git status** - check the status of your local repository (viewing local changes)  
**git add [filename]** - add files to the staging area, preparing them to be included in the next commit  
**git add .** - add everything in the current directory  
**git add -A** - add everything in the current directory and its subdirectories to the staging area  
**git commit –m** "[message]" - commit changes from staging area to the local repository and create a new commit with a descriptive message  
**git push** [remote name] [local name] - send 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 add (add for save) -> git commit (save) -> git push (upload)

**GitHub**

* the largest source code hosting platform
* offers both public and private repositories for individual users and teams
* main purpose is to host repos & to synch with local git
* Git synchronizes versions, GitHub hosts the repositories
* GitHub Services:
  + Git Source Code Repository
  + Issue Tracker
  + Project Board (Kanban Style)
  + Wiki Pages (Documentation)