# Voting Application

**Overview of the Application**

The provided Flask application is a simple voting system that allows users to cast votes for two options: "Virat" and "Dhoni." It uses MongoDB for data storage to keep track of the votes and includes functionality to view results and reset the votes.

**Components**

1. **Frontend**:
   * **Technology**: HTML, CSS, JavaScript (not explicitly included in the provided code but typically needed).
   * **Functionality**:
     + A user interface for casting votes and displaying results.
     + Forms or buttons to submit votes (e.g., "Vote for Virat" and "Vote for Dhoni").
     + A section to display the current voting results.
2. **Backend**:
   * **Technology**: Flask (Python web framework).
   * **Functionality**:
     + **Home Route (/)**: Renders the main voting page.
     + **Vote Route (/vote)**: Accepts POST requests to count votes. Validates the option and updates the vote count in MongoDB.
     + **Results Route (/results)**: Returns the current vote counts as JSON.
     + **Reset Route (/reset)**: Resets the vote counts for both options to zero.
     + **Database Interaction**: Uses pymongo to interact with MongoDB, including initialization of vote counts.
3. **Database**:
   * **Technology**: MongoDB.
   * **Functionality**:
     + Stores vote counts for "Virat" and "Dhoni" in a collection named votes within a database named voting\_db.
     + Initializes vote counts if the collection is empty.

**Summary of the Application**

This application is a simple voting system implemented using Flask and MongoDB. Users can vote for either Virat or Dhoni, and the application keeps track of the total votes for each option. It provides an interface to view the current results and reset the votes if needed. The key features include:

* **Vote Casting**: Users can submit their votes for one of the two candidates.
* **Vote Counting**: The application updates the vote counts in the MongoDB database.
* **Result Display**: Users can view the current vote counts in JSON format.
* **Reset Functionality**: The ability to reset votes to zero for both options.

Overall, this application serves as a basic demonstration of a web-based voting system, showcasing how to integrate Flask with MongoDB for data storage.

# Ansible playbook

---

- name: deploy voting application and store data in Mango DB

hosts: localhost

remote\_user: caasops

become: yes

tasks:

- name: Deploy vote application using flask web framework

shell: "kubectl create -f flask-objects.yaml"

tags: vote-deployment

- name: create service for vote application

shell: "kubectl create -f flask-svc.yaml"

tags: vote-service

- name: create virtualservice and gw for vote application

shell: "kubectl create -f voteapp-vs-gw.yaml"

tags: vote-vs-gw

- name: deploy mongodp to story information

shell: "kubectl create -f mango-objects.yaml"

tags: mangodb-deployment

- name: take backup of hosts

copy:

src: /etc/hosts

dest: /etc/hosts-bkp

tags: host-backup

- name: add host entry in hosts

blockinfile:

path: /etc/hosts

block: "100.96.44.204 www.mymangoapp.com"

tags: host-entry