**Cloud-Native Web Voting Application with Kubernetes:**

Our cloud-native web application presents a unique blend of technologies, providing users with internet access to cast their votes for their preferred programming language among six options: C#, Python, JavaScript, Go, Java, and NodeJS.

Tech Stack

* Frontend: The application's frontend boasts a responsive, user-friendly interface crafted with React and JavaScript to facilitate seamless voting.
* Backend and API: The backend is fueled by Go (Golang) and functions as the API to handle voting requests. MongoDB, configured with a replica set for data redundancy and high availability, serves as the database backend.

**Kubernetes Resources**:

Effectively deploying and managing this application leans on Kubernetes and its diverse set of resources:

* Namespace: Kubernetes namespaces establish isolated environments for different application components, ensuring clear separation and organization.
* Secret: Kubernetes secrets securely store sensitive data like API keys or credentials required by the application.
* Deployment: Kubernetes deployments dictate the number of application instances and provide guidelines for updates and scaling.
* Service: Kubernetes services steer incoming traffic to the correct instances, guaranteeing user access to the application.
* StatefulSet: For components demanding statefulness, like the MongoDB replica set, Kubernetes StatefulSets maintain order and unique identities.
* PersistentVolume and PersistentVolumeClaim: These Kubernetes resources oversee the necessary storage for the application, ensuring data persistence and scalability.

**Learning Opportunities:**

Engaging in the creation and deployment of this cloud-native web voting application through Kubernetes offers valuable learning opportunities:

* Containerization: Gain practical exposure to containerization tools such as Docker for packaging applications and their dependencies.
* Kubernetes Orchestration: Master the art of efficiently managing, deploying, and scaling containerized applications in a production environment using Kubernetes.
* Microservices Architecture: Explore the advantages and challenges of a microservices architecture, where frontend and backend components are decoupled and independently scalable.
* Database Replication: Acquire the skills to configure and oversee a MongoDB replica set, ensuring data redundancy and high availability.
* Security and Secrets Management: Embrace best practices for safeguarding sensitive data through Kubernetes secrets.
* Stateful Applications: Delve into the intricacies of deploying stateful applications within a container orchestration framework.
* Persistent Storage: Gain insights into how Kubernetes handles and provisions persistent storage for stateful applications.

By embarking on this project, you'll cultivate a deeper comprehension of cloud-native application development, containerization, Kubernetes, and the array of technologies involved in constructing and deploying modern web application.

### **Steps to Deploy**

<https://medium.com/@ghazanfarali9595/deploy-cloud-native-voting-app-on-kubernetes-855ae4cc1e54>

**MongoDB Initialization:**

cat << EOF | kubectl exec -it mongo-0 -- mongo

rs.initiate();

sleep(2000);

rs.add("mongo-1.mongo:27017");

sleep(2000);

rs.add("mongo-2.mongo:27017");

sleep(2000);

cfg = rs.conf();

cfg.members[0].host = "mongo-0.mongo:27017";

rs.reconfig(cfg, {force: true});

sleep(5000);

EOF

**Data Entry in MongoDb:**

cat << EOF | kubectl exec -it mongo-0 -- mongo

use langdb;

db.languages.insert({"name" : "csharp", "codedetail" : { "usecase" : "system, web, server-side", "rank" : 5, "compiled" : false, "homepage" : "https://dotnet.microsoft.com/learn/csharp", "download" : "https://dotnet.microsoft.com/download/", "votes" : 0}});

db.languages.insert({"name" : "python", "codedetail" : { "usecase" : "system, web, server-side", "rank" : 3, "script" : false, "homepage" : "https://www.python.org/", "download" : "https://www.python.org/downloads/", "votes" : 0}});

db.languages.insert({"name" : "javascript", "codedetail" : { "usecase" : "web, client-side", "rank" : 7, "script" : false, "homepage" : "https://en.wikipedia.org/wiki/JavaScript", "download" : "n/a", "votes" : 0}});

db.languages.insert({"name" : "go", "codedetail" : { "usecase" : "system, web, server-side", "rank" : 12, "compiled" : true, "homepage" : "https://golang.org", "download" : "https://golang.org/dl/", "votes" : 0}});

db.languages.insert({"name" : "java", "codedetail" : { "usecase" : "system, web, server-side", "rank" : 1, "compiled" : true, "homepage" : "https://www.java.com/en/", "download" : "https://www.java.com/en/download/", "votes" : 0}});

db.languages.insert({"name" : "nodejs", "codedetail" : { "usecase" : "system, web, server-side", "rank" : 20, "script" : false, "homepage" : "https://nodejs.org/en/", "download" : "https://nodejs.org/en/download/", "votes" : 0}});

db.languages.find().pretty();

EOF