

# SHORTLINK APP PLAN



# Plan & Overview

Detail overview of What and How

Tech Stack option

Key Element overview



# Proposed Tech Stack

Proposed tech stack includes programming language and server deployment information to serve the requirement

## *Programming Language*

### - **Golang:** High Performance

It is industry-proven language that has helped hundreds & thousands of companies serve a million requests at ease and affordable way.

## *Database Model & System*

Key-Value storage system:

- **BadgerDB:** Proven for a write-heavy operation

- **PogrebDB:** Best for a read-heavy operation

## *Server Deployment Option*

As a FAAS ( function as a service ) model for each functional request:

- Create / Update / Delete / Fetch

Or, Single binary in multi Node architecture with / without a centralized database system.

# Multi Node Architecture

The centralized Database system of Badger & Pogreb DB with Multi-Node architecture to serve large requests



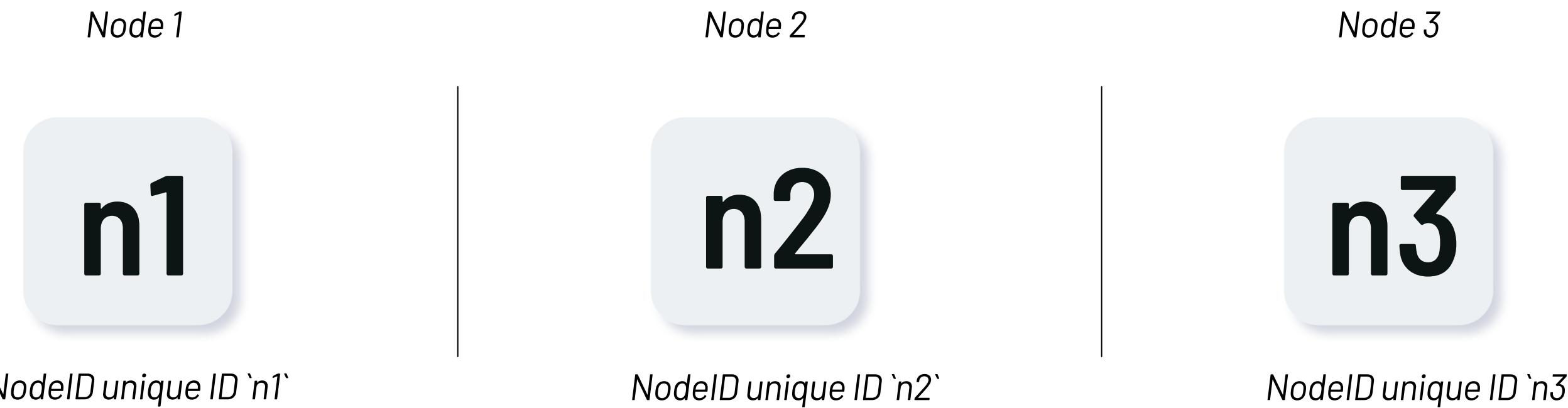
Centralized database app deployed at  
AWS / Heroku / DigitalOcean / Google  
Cloud platform



Deploying multiple Single non-  
embedded apps with an in-built cache  
solution performing end-end request

# Generating Unique Shortlink

HashID encode for generating monotonically increasing integers.



Creating global int variable with value 1 and monotonically increasing for every new insert with variable++. **Encoding each monotonically increasing int number using HashID to generate a short token** & Concatenating it with Node ID to form a unique short ID for every similar number in a different node instance.

Further, HashID takes a salt string to generate a token. Using the **User API token as salt** makes it more unique for every int in the same/multiple instances for each user to avoid any system-based increment conflict.

# Unique Identifier for Every User

Key Storage without Value for High Performance:

## Cookie-based Identification:

Generating a short random unique code and placing it as a cookie in the user's browser to identify the number of frequent visits by the user.

## **Reference:**

<https://github.com/rs/xid>

## **Generating Unique Code using "xid" which generates token based on:**

- 4-byte value representing the seconds since the Unix epoch,
- 3-byte machine identifier,
- 2-byte process id, and
- 3-byte counter, starting with a random value.

**Key:** [Short URL Code] + [Unique Cookie Code] + [Visit date time]

Checking Cookie existence in the browser & database, On match storing based on new date-time with the same cookie and short URL code, or generating new cookie code on new visit / mismatch of cookie.

# Report of Unique Shortlink

Fetching Report Mechanism based on Unique Identifier Key stored

***Short URL Code***

***Unique Cookie***

***Date Time***

Based on the Previous slide, You have already seen, we are storing each user's visit with a key and no value:

**Short\_URL\_Code - Unique\_Cookie - Date\_time**

- a). Get all the key which starts with "short\_url\_code" based on above stored "key" from the database.
- b). Make a temporary array of similar cookies matched from the above key and increase the counter of total click by 1, but only increase the counter of unique when a cookie is not present in the temporary array.
- c). Make a temporary array for each date and run the code inside (b) to calculate unique and total visits for a date stored in a key.

**Reference:** <https://github.com/go-awesome/shortlink/blob/main/handler/handler.go#L208>

# Quick Overview

**Golang** for programming the app

- Storing using `Key-Value` storage system
- Using Industry proven storage system for purpose:
- **BadgerDB** for write-heavy operation ( storing user's visit )
- **PogrebDB** for read-heavy operation ( reading shortcode )

**ShortCode Token:** Generating Shortcode by Encoding  
Monotonically increasing integer with User Token as salt  
string & concatenating it with NodetID

**Cookie** based identification of every new user.

# REVIEW SOURCE CODE

<https://github.com/go-awesome/shortlink>

 rebootcode	Production Ready
 handler	Production Ready
 helper	Production Ready
 .gitattributes	Initial commit
 .gitignore	Production Ready
 LICENSE	Production Ready
 architecture.md	Production Ready
 favicon.ico	Production Ready
 go.mod	Production Ready
 go.sum	Production Ready
 main.go	Production Ready
 readme.md	Production Ready
 rest.http	Production Ready
 sample.txt	Production Ready
 README.md	

## About The Project

