

# OpsDB Dev Startup Doc

OpsDB is the central automation system.

It is composed of 4 projects presently:

## Required Repositories

- <https://github.com/ghowland/web6.0/> - Web Server
- <https://github.com/ghowland/yudien/> - Data-Oriented Pipe-Based Language
- <https://github.com/ghowland/opsdb/> - Operational Automation logic
- <https://github.com/ghowland/ddd> - Data Definition Data - Guaranteed Data Correctness

## Setting Up Go

In your user home directory, create a go directory:

```
mkdir ~/go
```

Install Go:

```
brew install go
```

Dependency management:

Make sure you have `dep`, which is what we use for our dependency management:

```
go get github.com/golang/dep
```

Install `virtualgo` which we use to ensure dependency isolation:

```
go get -u github.com/GetStream/vg  
vg setup
```

```
source ~/.bashrc
```

## Running OpsDB

Starting from your Web6.0 workspace directory:

```
cd ~/go/src/github.com/  
mkdir ghowland  
git clone https://github.com/ghowland/web6.0/  
cd ghowland/web6.0  
vg init  
vg ensure
```

Now, configure web6 with the information for your database and ldap server. Web6 will first look in /etc/web6/web6.json and then in ~/secure/web6.json. Let's put the configuration in ~/secure/web6.json:

```
mkdir -p ~/secure
```

Place the json below into a new file named ~/secure/web6.json:

```
{  
  "ldap": {  
    "host" : "master.ldap.company.com",  
    "port" : 389,  
    "login_dn": "cn=admin,dc=company,dc=com",  
    "password": "SUPER_SECRET_PASSWORD",  
    "user_search": "ou=employees,dc=company,dc=com",  
    "group_search": "ou=teams,dc=company,dc=com"  
  },  
  "opsdb": {  
    "connect_opts": "user=postgres dbname=opsdb password='password'  
host=localhost sslmode=disable",  
    "database": "opsdb"  
  },  
  "ldap_override": {  
    "admin": {"name": "admin", "data": "example data"}  }
```

```
}  
  
}
```

Make sure you change `SUPER_SECRET_PASSWORD` to the correct LDAP password. Ask someone.

Run the command, after building it if it isn't yield compiled or has changed:

```
make run
```

You will see this:

```
Initializing Yudien  
Server :9000 listening...
```

If you are using an IDE, you will need to manually update your `$GOPATH` to include the virtualgo directory created for your project. For web6.0, that directory is `~/virtualgo/web6.0`, so your `$GOPATH` will look like this:

```
/home/yourname/.virtualgo/web6.0:/home/yourname/go/
```

In Goland you will need to add both directories to your gopath separately. Make sure you get the order correct.

## Install PostgreSQL

Install Postgres (make sure you install Postgres 10):

```
brew install postgresql
```

Create an `opsdb` database and user.

Import the database:

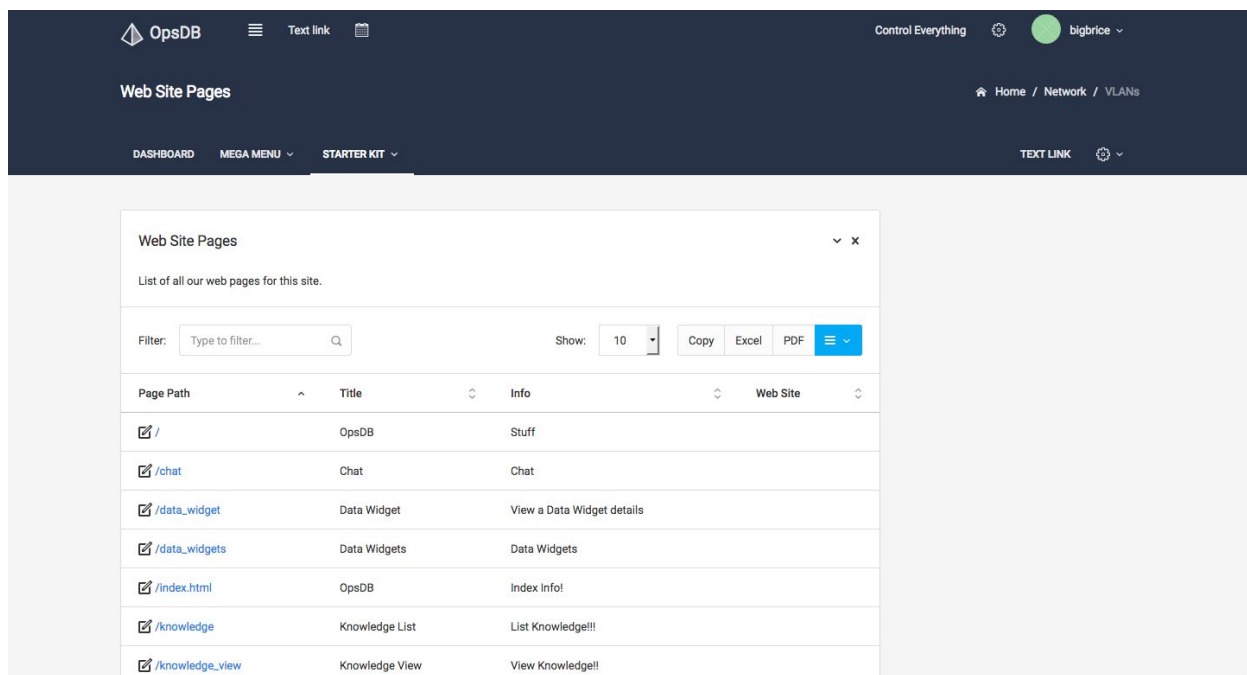
```
psql -U opsdb opsdb <
~/go/src/github.com/ghowland/web6/sql/web6.sql
```

## Testing OpsDB

Once web6 is running, then use a browser to look at a page:

[http://localhost:9000/web\\_site\\_pages](http://localhost:9000/web_site_pages)

It should look like this:



The screenshot displays the OpsDB Web Site Pages interface. The top navigation bar includes the OpsDB logo, a hamburger menu, a 'Text link' button, and a 'Control Everything' button. The main header shows 'Web Site Pages' and a breadcrumb trail 'Home / Network / VLANs'. Below the header, there are tabs for 'DASHBOARD', 'MEGA MENU', and 'STARTER KIT'. The main content area is titled 'Web Site Pages' and contains a list of all web pages for this site. The list has a filter input, a 'Show' dropdown set to '10', and buttons for 'Copy', 'Excel', 'PDF', and a menu icon. The table lists the following pages:

Page Path	Title	Info	Web Site
/	OpsDB	Stuff	
/chat	Chat	Chat	
/data_widget	Data Widget	View a Data Widget details	
/data_widgets	Data Widgets	Data Widgets	
/index.html	OpsDB	Index Info!	
/knowledge	Knowledge List	List Knowledge!!	
/knowledge_view	Knowledge View	View Knowledge!!	

## Hacking on OpsDb

As your hacking on opsdb, many times you will need to change one of the dependencies and use that change from somewhere else. For example, you may need to change code in the yudien repository and make use of those changes in web6.

For example, let's say you are also working on Yudien and installed it in your GOPATH with the following commands:

```
cd ~/go/src/github.com/ghowland
git clone git@github.com:ghowland/yudien.git
cd yudien
vg init
vg ensure
```

You now want web6 to use this version of yudien:

```
cd ~/go/src/github.com/ghowland/web6.0
vg localInstall github.com/ghowland/yudien
```

Hack on yudien and web6. Submit a pull request for yudien. Once it's been pulled into master, run:

```
vg ensure -- -update github.com/ghowland/yudien
```

This will update the Gopkg.lock file with the latest commit id of yudien. Submit this change and your other changes as a pull request. If you continue hacking on yudien, make sure to run localInstall again.

## Coding Style

To avoid (most) arguments of style, like most other Go projects, we use [gofmt](#) to format our code automatically. Good editors can run the appropriate command for you each time the file is saved.

Neo/vim:

Install [vim-go](#) and add the following to your ~/.vimrc

```
let g:go_fmt_command = "goimports"
let g:go_metalinter_autosave_enabled = ['vet', 'golint']
```

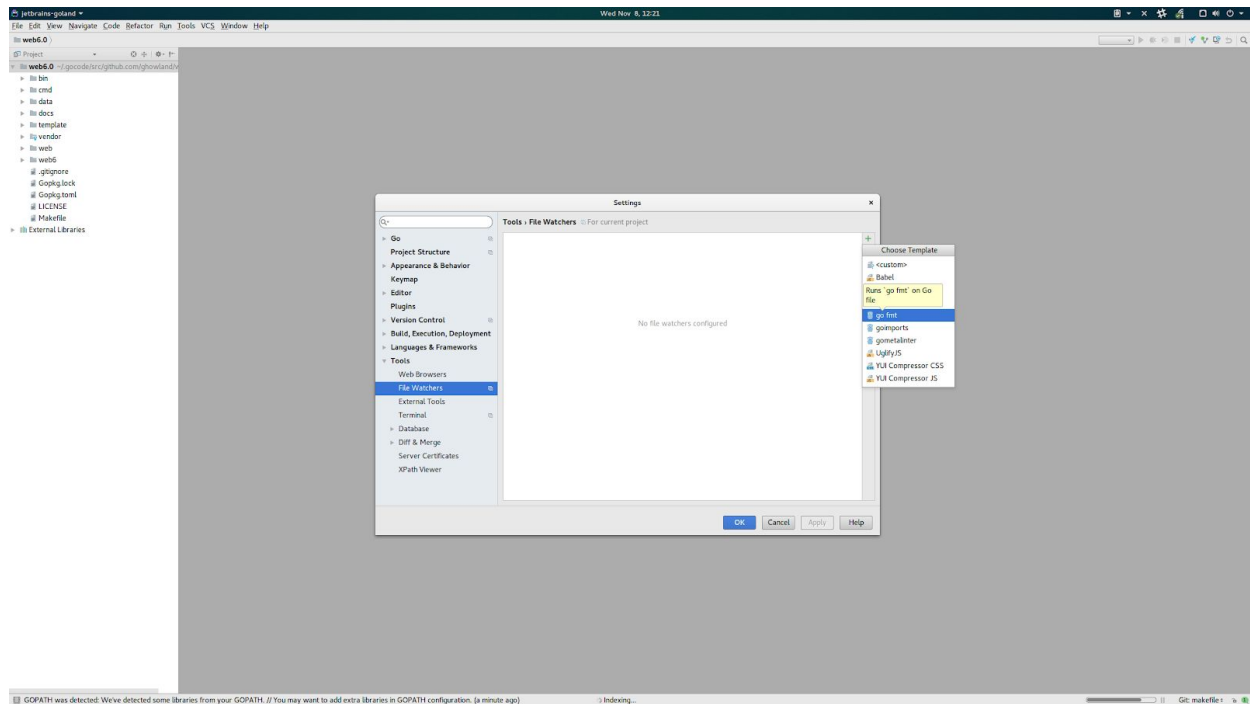
Gogland:

**Go to File->Settings->Tools->File Watchers**

(Older versions have this setting under File->Settings->Go->On Save)

**Press + Add 'gofmt' and 'goimports' and 'gometalinters'**

(Screenshot below of Golang 1.0 Preview)



## How To

### Run the schema exporter:

The schema exporter needs to be run any time the schema changes. Dataman uses this to make queries to the database, and will not know about any new or changed fields until this is done.

Edit the following file to match your Database configuration

```
vi  
~/go/src/github.com/jacksontj/dataman/src/storage_node/storagenode/  
config.yaml
```

### Run the schema exporter

```
./schemaexport --databases=<DATABASE_NAME> > schema.json
```

Copy the file in your web6.0/data directory

```
cp schema.json ~/go/src/github.com/ghowland/web6.0/data/
```





