



Sandpiper (/spaces/127/sandpiper) ▸ Discussions (...)

Private Space · Technology Standards (/spaces/9/technology-standards/technologyhome)

(/spaces/127/sandpiper)

Activity Stream (/spaces/127/sandpiper/?act=1)

People (/spaces/127/sandpiper/people)

Content ▼

Posted in: Development (/spaces/127/sandpiper/forums/5366/development)

Command Line Interface (CLI) Tools

✉ Unsubscribe from this discussion

📡 Subscribe to RSS (../..../..../spaces-cf/forums/rss-space-posts.ashx?

spaceID=127&topicID=5393&forumID=5366&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



12/12/2019

Introduction

Level 1 implementation of Sandpiper means syncing "file-based" data-objects. I think it makes sense to implement two CLI tools for Level 1. This functionality would be duplicated when we create the Admin screens, but should be much easier to implement than a full admin interface (since there is no UI and it just calls the exposed API). Plus it will be useful for early testing.

Add File-Based Objects

Implement the "add" command to add "file" data-objects to a slice in the pool. This command could be called by an internal PIM, for example, to "publish" completed delivery files.

For example:

```
sandpiper add \  
-u user          \ # database user  
-p password      \ # database password, prompt if not supplied  
-s "aap-slice"   \ # slice name  
-t "aces-file"   \ # file-type  
-f "acme_brakes_full_2019-12-12.xml" # file to add
```

This would add the ACES xml file as a data-object to the "aap-slice" slice.

The database connection information (including user and password) could be pulled from a "config" file. If the password is not provided, it would prompt for it.

Here is an example of a `config.yml` file.

```
database:
  psn: postgres://user:password@localhost:5432/sandpiper?sslmode=disabled
```

Sandpiper API

The following api is used to add an object with the `sandpiper add` command:

```
POST /slice/{slice-name}
```

The "request body" would look like this:

```
{
  "token": "--jwt here--",
  "oid": "da9fd323-151a-4035-82db-7b18e4ba6c79",
  "type": "aces-file",
  "payload": "MQkxOTk1CTEJNQkJCQkJCQkJCQkJCQk3NDE4Mg1NR0MJNDU3M ...",
  ...
}
```

Pull File-Based Objects

Implement the "pull" command to retrieve "file" data-objects from an optional slice in the pool. If the slice is not supplied it will create a sub-directory for each one it finds.

For example:

```
sandpiper pull \
-s "aap-slice" \ # optional slice
-t "aces-file" \ # optional file-type
-d "publish"     # required output directory
```

The generated file structure might be something like:

```
publish
|-- aces-file
    |-- aap-slice
```

Leaving off an option will pull all available.

cli-tools (/spaces/127/sandpiper/searchresults?keyword=%22cli-tools%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5394&SpaceID=127)



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



12/17/2019

In order to test the client CLI tools, it would help to have a server to test against.

Since we might want to develop these tools while working on the API design and before the server is completed, it makes sense to have a way to simulate the api (or at least the api endpoints needed by these tools).

I've started a program to mock the server-api, and it is attached. It's currently just a shell that partially simulates the following routes:

```
[{
  "method": "GET",
  "path": "/v1/login",
  "name": "main.login"
},
{
  "method": "GET",
  "path": "/v1/slices",
  "name": "main.getMySlices"
},
{
  "method": "POST",
  "path": "/v1/slices/:id",
  "name": "main.postObject"
},
{
  "method": "GET",
  "path": "/v1/routes",
  "name": "main.listRoutes"
}]
```

No database access is included. We simply return static json responses and result codes.

It is written in go, and includes a small README file.

📎 Attachments

cli (/spaces/127/sandpiper/searchresults?keyword=%22cli%22&searchtags=1)

mock-api (/spaces/127/sandpiper/searchresults?keyword=%22mock-api%22&searchtags=1)

simulation (/spaces/127/sandpiper/searchresults?keyword=%22simulation%22&searchtags=1)

tools (/spaces/127/sandpiper/searchresults?keyword=%22tools%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5398&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



1/5/2020

We've changed the database section in the `config.yaml` file since the original post was made. It now looks like this:

```
database:
  dialect: postgres
  database: sandpiper
  user: sandpiper      # Can override with "DB_USER" env variable
  password: autocare   # Use "DB_PASSWORD" env variable in production!!!
  host: localhost
  port: 5432
  sslmode: disable
  timeout_seconds: 5
  log_queries: true
```

Notice that we no longer include a URL (i.e. "psn") for database access. Instead, there is a `config.URL()` function which creates the properly formed URL string from supplied config values:

```
// URL creates a connection URL from a `database` section overriding User/Password
with env vars if found
func (d *Database) URL() string {
    // postgres://username:password@host:port/database?sslmode=disable
    return fmt.Sprintf("%s://%s:%s@%s:%s/%s?sslmode=%s",
        d.Dialect,
        env("DB_USER", d.User),
        env("DB_PASSWORD", d.Password),
        d.Host,
        d.Port,
        d.Database,
        d.SSLMode,
    )
}
```

The benefit is that we can now accept environment variables to override certain variables.

EDIT: It turns out, of course, that the cli tools are api-based and so don't need access to the database. The above is true for the api config, but the cli config is simply user, password and sandpiper url address.

[config.yml \(/spaces/127/sandpiper/searchresults?keyword=%22config.yml%22&searchtags=1\)](/spaces/127/sandpiper/searchresults?keyword=%22config.yml%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5454&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



2/16/2020

Here is the bulk of the code for the `sandpiper add` command. It shows how easy it is to use our API.

```

// Add attempts to add a new file-based grain to a slice
func Add(c *args.Context) error {
    var grain *sandpiper.Grain

    // save parameters in a `params` struct for easy access
    p, err := getParams(c)
    if err != nil {
        return err
    }

    // connect to the api server (saving token)
    api, err := Connect(p.addr, p.user, p.password)
    if err != nil {
        return err
    }

    // lookup sliceID by name
    slice, err := api.SliceByName(p.sliceName)
    if err != nil {
        return err
    }

    // load basic info from existing grain (if found) using alternate key
    grain, err = api.GrainExists(slice.ID, p.grainType, p.grainKey)
    if err != nil {
        return err
    }

    // if grain exists, remove it first (prompt for delete unless "noprompt" flag)
    if grain.ID != uuid.Nil {
        if p.prompt {
            grain.Display() // show what we're overwriting
            if !AllowOverwrite() {
                return errors.New("grain could not be added without overwrite")
            }
        }
        err := api.DeleteGrain(grain.ID)
        if err != nil {
            return err
        }
    }

    // encode supplied file for grain's payload
    data, err := payload.FromFile(p.fileName)
    if err != nil {
        return err
    }

    // create the new grain
    grain = &sandpiper.Grain{

```

```
SliceID: &slice.ID,  
Type:    p.grainType,  
Key:     p.grainKey,  
Source:  filepath.Base(p.fileName),  
Encoding: "gzipb64",  
Payload: data,  
}  
  
// finally, add the new grain  
return api.Add(grain)  
}
```

sandpiper-add-cli (/spaces/127/sandpiper/searchresults?keyword=%22sandpiper-add-cli%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5749&SpaceID=127) Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



:

4/9/2020

We've completed work on the CLI utility (which includes "add", "list" and "pull" commands). Attached is the latest documentation.

We are now able to add ACES files (for example) to a slice and then save it back to the file system. The payloads are stored automatically in a compressed format (which will also transmit safely across the wire in JSON).

While working on the "pull" command (that saves grain payloads to disk), it became clear that:

1. **Security is important.** Although designed for local use (against your own server), anyone with the CLI utility (and the proper "admin" credentials) could connect to a Sandpiper server on the Internet and save files to their local machine.
2. **Primary & Secondary Roles.** The CLI utility works equally well against a primary pool and a secondary pool. It was originally conceived to "add" on the primary, and "pull" on the secondary, but there is no simple way (or reason, I think) to limit its use.

This second point has me thinking more about whether or not to have different code for primary vs secondary APIs. As we start work on the secondary role, the vast majority of the code is identical. We need users, subscriptions, slices and grains, for example, in both.

While I don't like the idea of co-mingling data in the same database, I think we can save a "role" in the database (or in the config file) to allow us to handle any differences.


The sync process is really the main difference, and there are arguments for having this be a separate process from the API. I'll post more on this in a different thread when I have some time (or we can discuss it on our call tomorrow).

Attachments

`cli (/spaces/127/sandpiper/searchresults?keyword=%22cli%22&searchtags=1)`

`server-roles (/spaces/127/sandpiper/searchresults?keyword=%22server-roles%22&searchtags=1)`

`sync (/spaces/127/sandpiper/searchresults?keyword=%22sync%22&searchtags=1)`

 1 Like

Reply (/forums/post?tid=5393&ReplyPostID=5918&SpaceID=127)

Answer



Krister Kittelson (<https://autocare.communifire.com/people/krister-kittelson>)

4/9/2020



Hurray! Thank you, good to hear the in/out is on :) This is exciting to see it coming together.

On the db topic, I think commingling the data is probably an acceptable path, given the right restrictions like you mentioned (which would vary some depending on the flavor of db). And the local security is similarly a doable thing, I think it's up to the server administrator to avoid giving out admin credentials freely, as with any software.

 Like

Reply (/forums/post?tid=5393&ReplyPostID=5919&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)

4/9/2020

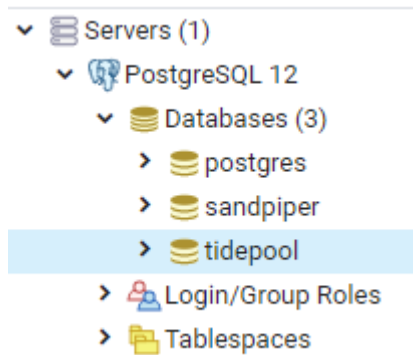


On 4/9/2020 4:26 PM, Krister Kittelson said:

I think commingling the data is probably an acceptable path...

That word (commingling) just doesn't look right! And for that reason alone I think it is a poor design choice :)

Actually, the current design would not work well with both roles in the same database. You could probably do it with separate schemas (<https://www.postgresql.org/docs/current/ddl-schemas.html>) (which is nothing more than a namespace within a database) but I've never liked that approach. I think it is much cleaner just to keep them separate. They can both be on the same Postgresql server, though, if that's desired. For example, we have both sandpiper (primary) and tidepool (secondary) databases attached to this server:



👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5920&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



4/9/2020

The pdf conversion of the sandpiper CLI documentation didn't format the code blocks very well. I've fixed that and put it up on our web docs [here](https://sandpiper.onrender.com/docs/sandpiper-utility/) (<https://sandpiper.onrender.com/docs/sandpiper-utility/>).

web-docs (/spaces/127/sandpiper/searchresults?keyword=%22web-docs%22&searchtags=1)

👍 Like

Reply (/forums/post?tid=5393&ReplyPostID=5921&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



4/16/2020

Added "server-role"...

```
C:\Users\dougw\autocare\sandpiper>task server
./api

  ____
 /  __| |  _ _ _ _ _ _ _ _ | | _ _ _ ( _ _ _ _ _ _ _ _
 \  __ \ / _ ` | ' _ \ / _ ` | ' _ \ | | ' _ \ / _ ` |
   __) | ( | | | | | ( | | | ) | | | ) | _ _ / |
 | ____/ \ _ _ _ _ | | | \ _ _ _ _ | _ _ / | | _ _ _ |
               | |       | |

Sandpiper API Server (v0.1.2-44-g1e2090d-dirty)
Copyright (c) The Sandpiper Authors. All rights reserved.

Database: "sandpiper"
DB Version: 2
Server role: "primary"

⇒ http server started on [::]:8080
```

server-role (/spaces/127/sandpiper/searchresults?keyword=%22server-role%22&searchtags=1)

 Like

Reply (/forums/post?tid=5393&ReplyPostID=5926&SpaceID=127) Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)

5/17/2020

I added a new sandpiper cli command:

sandpiper init

This command creates a new database, asking questions about what to name it, what server-role it has (primary or secondary) and then seeds the database with the first company and user records.

We now have a **simple way to get started**. All you need to do is download and install PostgreSQL, run `sandpiper init` and launch the server.

db-seed (/spaces/127/sandpiper/searchresults?keyword=%22db-seed%22&searchtags=1)

sandpiper-init (/spaces/127/sandpiper/searchresults?keyword=%22sandpiper-init%22&searchtags=1)

 Like



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



5/18/2020

I added a new sandpiper cli command:

```
sandpiper secrets
```

This command simply creates the two "secrets" required by the server. These can be copy/pasted into the config.yaml file or used as environment variables in a shell script that launches the server.

```
PS C:\Users\dougw\autocare\sandpiper\cmd\cli> ./sandpiper secrets
sandpiper (v0.1.2-67-g5facfce-dirty)
Copyright 2020 The Sandpiper Authors. All rights reserved.

GENERATE RANDOM SERVER 'SECRETS'

# ENVIRONMENT VARIABLES (remove double quotes from Docker .env files):

export APIKEY_SECRET="YThhMTI1YmUzN2Q0MmVlNDQxOTM3NmJkZDIxNDA4ZjI="
export JWT_SECRET="NzVlYWYxYzg1NjZiMDlkM2JiMTI2OTkyNjQwZDFjMzA5MDE0OGI0MWU5ZDcxMmQ2MzI4NTU3ZmEyMmNiYjling=="

# CONFIG FILE (YAML) ENTRIES:

api_key_secret: YThhMTI1YmUzN2Q0MmVlNDQxOTM3NmJkZDIxNDA4ZjI=
secret: NzVlYWYxYzg1NjZiMDlkM2JiMTI2OTkyNjQwZDFjMzA5MDE0OGI0MWU5ZDcxMmQ2MzI4NTU3ZmEyMmNiYjling==
```

I also changed `sandpiper init` to generate the config.yaml files with a the database configuration and random secrets using this code.

```
generate-secrets (/spaces/127/sandpiper/searchresults?keyword=%22generate-secrets%22&searchtags=1)
```

👍 1 Like

