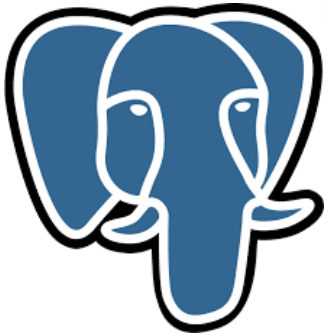


Continuous delivery in Postgres-centric solutions

Affinitas

Artem Panchoyan

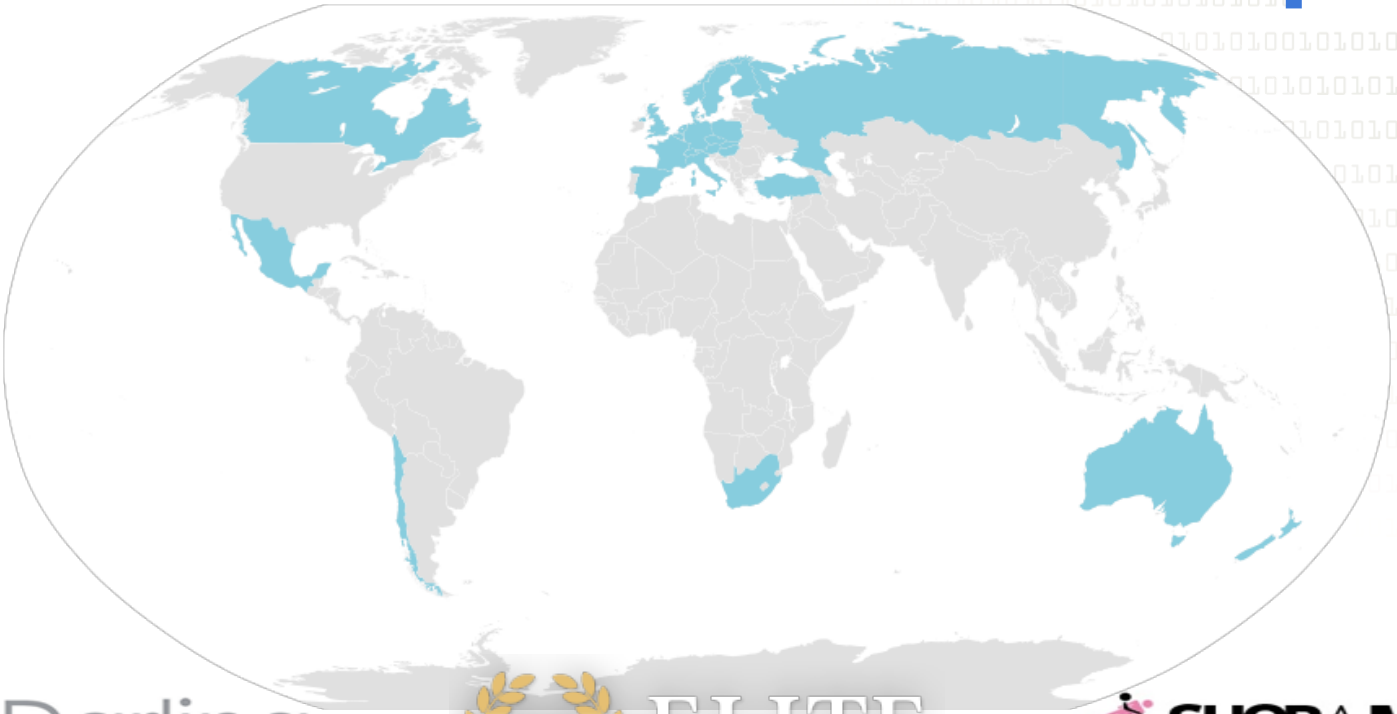
Agenda

- Who and what we are
- What's this about
- What is the problem
- How we are planning to fix it

Affinitas

15 mln users in 25 countries managed from Berlin HQ

Who and what we are



eDarling

ELITE SINGLES

SHOPAMAN
WO FRAUEN MÄNNER SHOPPEN

Affinitas

Moreover, we are

Who and what we are

300

people

10

DB engineers

80

Databases

TBs

of data

Affinitas

Agenda

Who and what we are

What's this about

What is the problem

How we are planning to fix it

Affinitas

This is about fast delivery

First, what it's not about:

- It's not about Postgres performance
- It's not about new features of Postgres
- First part is not even about Postgres itself!

So what is it about?

- It's about continuous delivery on Postgres
- It's about how to effectively develop on Postgres on a daily basis and deliver high quality stable code to end users

For us Postgres is not only a DBMS. It's a

Development environment

...and maybe someday a data operating system

Agenda

Who and what we are

What's this about

What is the problem

How we are planning to fix it

Affinitas

Open issues

What is the problem

Preparation/Requirements

- Delivery of feature can't happen till all version is ready
- How to handle ad-hoc requests
- How to plan resources for versions

Development

- What IDE choose to write the code
- How to debug it
- How to run it
- How to test it

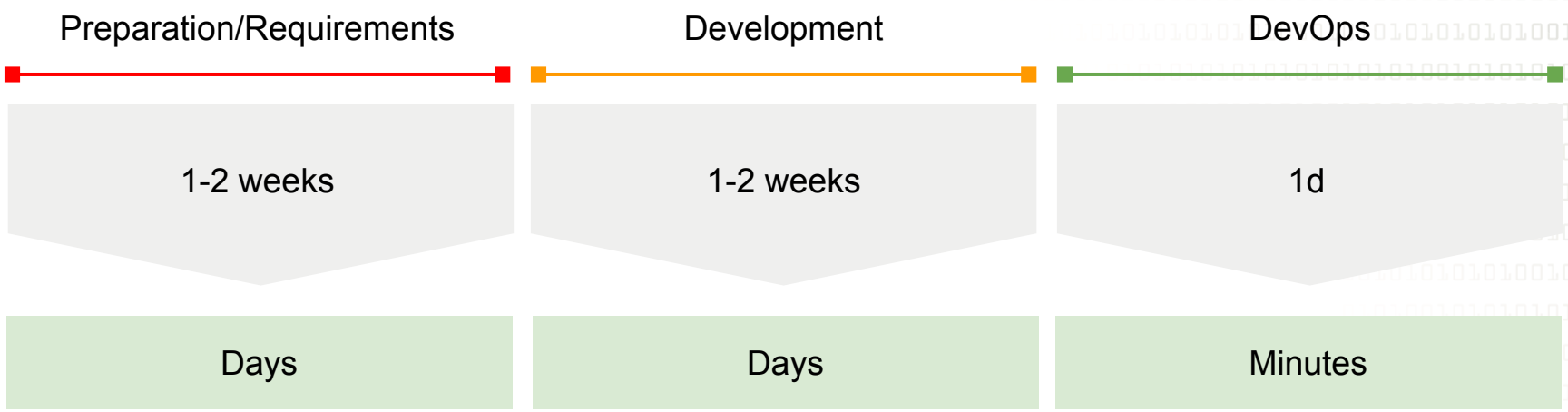
DevOps

- How to roll-out to different environments
- How to safely change DB structure
- How to keep control over DB change

Affinitas

Area of focus and goals

What is the problem



Affinitas

Agenda

Who and what we are

What's this about

What is the problem

How we are planning to fix it

Affinitas

How are we planning to fix it

https://github.com/affinitas/

pgpm

Affinitas

This is about fast delivery

How are we planning to fix it

- It deploys packages
- It installs itself
- It provides api for correct package importing using search_path (to be released v0.0.4)
- It resolves dependencies (to be released v0.0.4)
- It helps versioning and source control (under development - v0.1.0)

Affinitas

pgpm package configuration

How are we planning to fix it

```
{
   "description"    : "Servicetool API" ,
   "name"           : "edarling_servicetool_api",
   "version"        : "01_09_00",
   "class"          : "postgres_sql",
   "subclass"       : "versioned",
   "types_path"     : "types",
   "functions_path" : "functions"
}
```

Affinitas

pgpm installation

```
$ pgpm install <connection_string>
```

```
CREATE TYPE _pgpm.package_version as
(
    major smallint NOT NULL DEFAULT 0,
    minor smallint NOT NULL DEFAULT 0,
    patch smallint NOT NULL DEFAULT 0,
    pre character varying(255),
    metadata character varying(255)
);
-- class of package. Can refer to function/types schema, DDL schema
CREATE TABLE _pgpm.package_classes
(
    pkg_c_id serial NOT NULL,
    pkg_c_name character varying(255),
    pkg_c_created timestamp without time zone DEFAULT now(),
    pkg_c_last_modified timestamp without time zone DEFAULT now(),
    CONSTRAINT package_class_pkey PRIMARY KEY (pkg_c_id),
);
INSERT INTO _pgpm.package_classes (pkg_c_name)
    VALUES ('postgres_sql');
INSERT INTO _pgpm.package_classes (pkg_c_name)
    VALUES ('postgres_ddl');

-- subclass of package. Can refer either versioned schema (that adds suffix at the end) or non-versioned
CREATE TABLE _pgpm.package_subclasses
(
    pkg_sc_id serial NOT NULL,
    pkg_sc_name character varying(255),
    pkg_sc_created timestamp without time zone DEFAULT now(),
    pkg_sc_last_modified timestamp without time zone DEFAULT now(),
    CONSTRAINT package_class_pkey PRIMARY KEY (pkg_sc_id),
);
INSERT INTO _pgpm.package_subclasses (pkg_sc_name)
    VALUES ('versioned');
INSERT INTO _pgpm.package_subclasses (pkg_sc_name)
    VALUES ('basic');

-- info on packages
CREATE TABLE _pgpm.packages
(
    pkg_id serial NOT NULL,
    pkg_name character varying(255),
    pkg_description text,
    pkg_version _pgpm.package_version,
    pkg_class integer,
    pkg_subclass integer,
    pkg_dependencies
    pkg_created timestamp without time zone DEFAULT now(),
    pkg_last_modified timestamp without time zone DEFAULT statement_timestamp(),
    CONSTRAINT package_pkey PRIMARY KEY (pkg_id),
    CONSTRAINT package_class_fkey FOREIGN KEY (pkg_class) REFERENCES _pgpm.package_classes (pkg_c_id)
    CONSTRAINT package_subclass_fkey FOREIGN KEY (pkg_subclass) REFERENCES _pgpm.package_subclasses
(pkg_sc_id)
);
```

How are we planning to fix it

Affinitas

pgpm deployment

```
$ pgpm deploy <connection_string>
[-m | --mode <mode>]
[-o | --owner <owner_role>]
[-u | --user <user_role>...]
[-f | --file <file_name>...]
```

Affinitas

```
            _old_schema_exists = True
            _old_schema_rev = 0
            while _old_schema_exists:
                cur.execute("SELECT EXISTS (SELECT schema_name FROM information_schema.schemata "
                            "WHERE schema_name = %s);", (schema_name + '_' + str(_old_schema_rev),))
                _old_schema_exists = cur.fetchone()[0]
                if _old_schema_exists:
                    _old_schema_rev += 1
            _old_schema_name = schema_name + '_' + str(_old_schema_rev)
            print('Schema already exists. It will be renamed to {0} in moderate mode. Renaming...'
                  .format(_old_schema_name))
            _rename_schema_script = "\nALTER SCHEMA " + schema_name + " RENAME TO " + _old_schema_name + ";\n"
            cur.execute(_rename_schema_script)
            print('Schema {0} was renamed to {1}.'.format(schema_name, _old_schema_name))
            create_db_schema(cur, schema_name, user_roles, owner_role)
        else:
            _drop_schema_script = "\nDROP SCHEMA " + schema_name + " CASCADE;\n"
            cur.execute(_drop_schema_script)
            print('Droping old schema {0}'.format(schema_name))
            create_db_schema(cur, schema_name, user_roles, owner_role)

    # Reordering and executing types
    if types_files_count > 0:
        if arguments['--file']:
            print('Deploying types definition scripts in existing schema without dropping it first '
                  'is not support yet. Skipping')
        else:
            type_ordered_scripts, type_unordered_scripts = reorder_types(types_script)
            # print('\n'.join(type_ordered_scripts)) # uncomment for debug
            # print('\n'.join(type_unordered_scripts)) # uncomment for debug
            if type_ordered_scripts:
                cur.execute('\n'.join(type_ordered_scripts))
            if type_unordered_scripts:
                cur.execute('\n'.join(type_unordered_scripts))
            print('Types loaded to schema {0}'.format(schema_name))
    else:
        print('No type scripts to deploy')

    # Executing functions
    if functions_files_count > 0:
        print('Running functions definitions scripts')
        cur.execute(functions_script)
        print('Functions loaded to schema {0}'.format(schema_name))
    else:
        print('No function scripts to deploy')

    # Commit transaction
    conn.commit()

    close_db_conn(cur, conn, arguments.get('<connection_string>'))

else:
    print(arguments)

if __name__ == '__main__':
```

How are we planning to fix it

Thank you
https://tech.affinitas.de

artem.panchoyan@affinitas.de
skype: ponchic

Affinitas