Deployment

v1.9

Sirenia

September 20, 2019

Contents

Typically we'd run a *connected configuration* with a Context Management Registry (Kwanza) and its interface and the let the Driver Platform (Manatee) instances which runs on the local desktops connect to the CMR for its configuration. It is however also possible to run the Manatee without the CMR on a *standalone configuration* by providing it with an initial configuration.

1 Connected configuration

The software components needed to run this setup are;

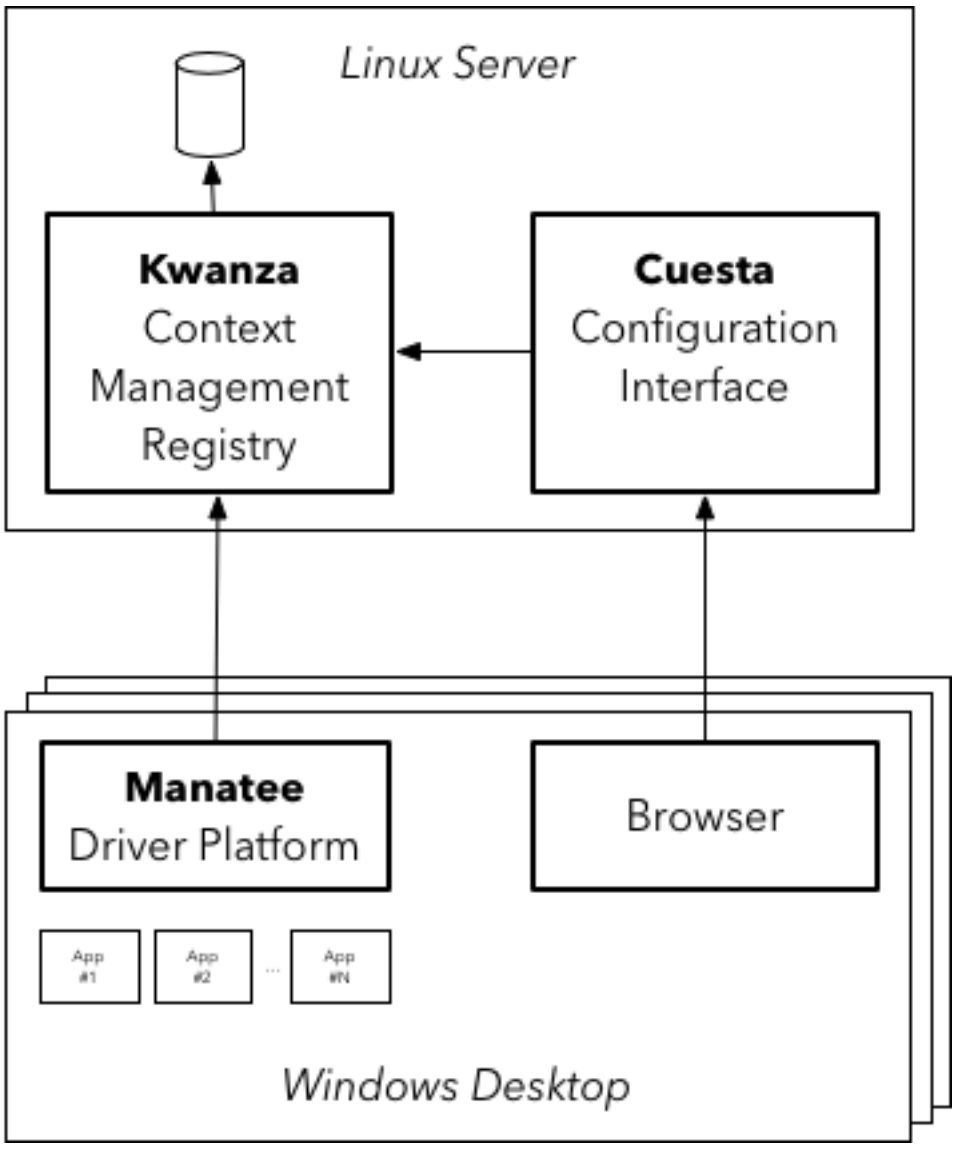

Figure 1: Deployment

- A database - we currently support postgresql, rethinkdb and an embedded db for configurations
- *Kwanza* Context Management Registry - a server which provides access to the db
- *Cuesta* Configuration interface for Kwanza - a web application for configuring flows, manages configurations in Kwanza
- *Manatee* Driver Platform - runs on local Windows desktops and interfaces with applications, gets it configuration from Kwanza

1.1 Kwanza

Kwanza is distributed as a single binary or as a docker image. Contact us for access to the distribution platform for Kwanza if you haven't already got a version.

1.1.1 Pre-requisites

- a valid SSL certificate (and private key), e.g. from letsencrypt.org, it should match the location (url or ip address) you plan on running Kwanza from
- [optionally] a database (not needed if you're using the embedded db option)

1.1.2 Installing and running the Kwanza binary

Run the `kwanza serve` command with suitable arguments. Running `kwanza serve --help` will display available options (here shown for v2.0.0):

```
$ ./kwanza serve --help
Command will start registry, connecting to the given database.
  The supported databases are:

  - git, use url of the form https://<location-of-remote-git-repo> or file://<lo
  - postgresql, pg://<user>:<pass>@<host>:<port>/<name-of-database>

Usage:
  kwanza serve [flags]

Flags:
      --cluster             Enable cluster mode
      --color string        Optional color to be shown in admin interface etc (d
  -d, --database string     Database connection url (default "git:///tmp/kwanza"
      --expvarsPort int     The port from which to serve expvars, -1 to disable
  -h, --help                help for serve
      --install             Install the service with the given arguments
  -p, --port int            The port Kwanza will listen for requests on (default
  -s, --salt string         The salt used for checking hashed passwords (default
      --typeprefix string   Optional type prefix to add to all types
      --uninstall           Uninstall the service
      --verbose             Enable verbose (debug level) logging
```

```
Global Flags:
      --auth [none|jwt]   The mechanism by which to authenticate. Allowed values
      --cert string       The location (relative to pwd) of the certificate/publ
      --config string     config file (default is $HOME/.kwanza.yaml)
      --key string        The location (relative to pwd) of the private key. (de
```

E.g.

```
$ ./kwanza serve --auth jwt --database pg://foo:bar@localhost:1234 --port 6000
```

will start an instance which

- connects to a posqtgresql database running on `localhost:1234` with username `foo` and password `bar`
- serve http requests on port `6000` and grpc on port `6001`
- use `jwt` as authentication mechanism

1.2 Cuesta

Cuesta is the configuration interface for Kwanza. It consists of static html files and javascript and can therefore be served by any webserver. Simply place the files in location served by an http server and it should work. It is also available as a docker image (nginx serving the files).

Only configuration (hardcoded at build-time) is the location of the Kwanza to connect to. Cuesta must therefore be built specifically for a deployment.

1.3 Docker with kwanza and cuesta

Perhaps the easiest approach is to run a `docker` setup. In that case inspiration may be had from the following `docker-compose` configuration:

```
version: '3'
services:

  kwanza:
    image: registry.gitlab.com/sirenia/dist/kwanza:v2.4.1
    restart: unless-stopped
    environment:
      KWANZA_DATABASE: pg://postgres:postgres@postgres/kwanza
      KWANZA_CERT: "/cert/cert.pem"
```

```
      KWANZA_KEY: "/cert/key.pem"
      KWANZA_SALT: <salt-goes-here>
      KWANZA_AUTH: jwt
    ports:
      - "8000:8000"    # HTTP(S)
      - "8001:8001"    # TCP (gRPC)
      - "6060:6060"    # Profiling
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
      - "/usr/local/etc/sirenia/kwanza/conf:/etc/sirenia/kwanza"
    depends_on:
      - postgres

  cuesta:
    image: registry.gitlab.com/sirenia/dist/cuesta:v1.3.0
    restart: unless-stopped
    environment:
      CUESTA_CERT: "cert.pem"
      CUESTA_KEY: "key.pem"
      KWANZA_URL: "https://<server-fqn>:8000/v1"
      KWANZA_STREAMURL: "wss://<server-fqn>:8000/v1/stream"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - "/usr/local/etc/sirenia/cert:/cert"
    depends_on:
      - kwanza

  postgres:
    image: postgres:10
    restart: always
    ports:
      - "5444:5432"
    environment:
      PGDATA: "/data"
    volumes:
      - "/root/postgresdata:/data"
```

This will run a postgres, a kwanza and a cuesta instance and connect them appropriately. You still need to supply postgres logins, certificates and the like as well as access to the docker images for kwanza and cuesta.

1.4 Manatee

The Driver Platform (Manatee) is distributed as a machine-wide installer (`.msi` file) or as a per-user installer (`.exe` file) as well as a hybrid of the two.

1.4.1 Machine-wide

The machine-wide installer should be used for rolling out Manatee in enterprise environments or in deployments where a normal desktop user does not have administrative rights. Using the machine-wide installer means that:

- The application will *not* self-update.
- All users for a given machine will get the application installed.
- The application will *not* automatically be started when the installer is done.

The installation process is straightforward, simply run the `.msi` file. It should not show any UI and will install shortcuts in the startmenu.

1.4.2 Per-user

The per-user installer should be used when individual users themselves are responsible for installing software on their own machine. It has the properties that:

- The application will automatically update when new versions are released.
- The application willl only get installed for the one user running the installer.
- The installer will start the application when it has been installed.

The installation procedure is the same as for the machine-wide installer.

1.4.3 Hybrid

The hybrid installer is an `.msi` file which is intended to be run either by each individual user or by an administrator. It will install the application for the each user *when the user next logs in on the machine* using the per-user installer. Its intended use is for situations where an `.msi` installation is the only viable option (enterprise environments) but the automatic updates are considered critical.