

CONFIGURABLE APP: USER GUIDE

TABLE OF CONTENTS

1.0.....WHAT IS CONFIGURABLE APP ?	3
2.0...HOW DOES CONFIGURABLE APP FEATURE WORK?	4
3.0...PREREQUISITES.	5
4.0.....USER PROVIDED EXTERNAL JSON FIELDS	6
5.0. .WORKFLOW	7
6.0...VERSIONING	13
7.0. .USAGE OF CONFIGURABLE APP.	14

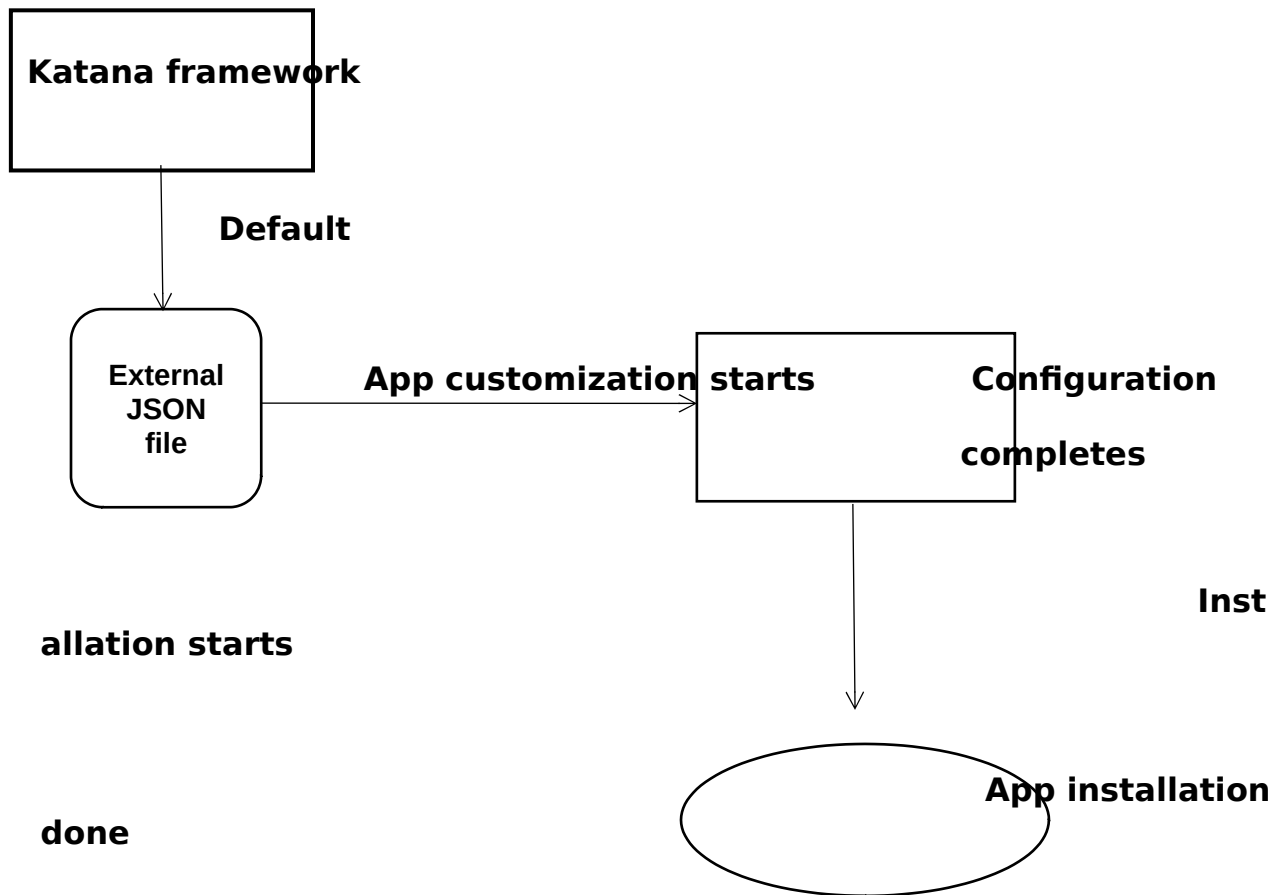
1.0 WHAT IS CONFIGURABLE APP?

Configurable app is a feature where users can install their apps under the katana by providing the requirements through JSON file. Developers who don't need default apps, they can remove these apps by using user defined JSON file. User defined required fields will be explained in coming pages.

Features:

- Allowing users to make changes their app title/framework name) name under katana UI
- Logo changes as per the user requirement
- Panel color changes
- Allowing users to install only default apps or only custom apps or both by providing the branch details

HOW DOES CONFIGURABLE APP FEATURE WORK



3.1 PREREQUISITES:

Configurable app requires the following to run successfully:

1. A Linux system
2. Python – version 3.6 or above
3. Katanaframework

Katana framework by default would install the dependencies of configurable app.

USER PROVIDED EXTERNAL JSON FILE FEILDS:

Once user will manage.py runserver, automatically katana will up and reflect in UI as per katanaframework. If user wants to get their logo, name, color and app, has to provide external JSON file by providing the below fields and below command.

app_title : User will provide app title or framework name here to display their name in katana UI. This is not mandatory field.

custom_logo_path : Users can save their logo image in external drive and can provide path under the JSON to display their logo image under katana UI. This is not mandatory field. User can provide upto 1MB size of image and pixels will be width:35px and height:35px. Supported formats are JPEG, PNG, JPG.

panel_color : Provide the color name or hexar color code to reflect in UI. This is not mandatory feild.

katana_default_apps : Provide the true/false under this field.

True: It will install all default apps which were there under katana.

False: It will not install any default apps and load with Wapp management.

apps_rely_on_postgresdb : Provide the app name under this field and it will use postgres database for the app.

katana_default_apps_branch : After providing **katana_default_apps** as **true**, this field will allow user to provide their branch name. This is not mandatory to provide and if not mentioned, automatically, It will pull master branch.

Ex: develop, feature/xxxx.

user_custom_apps: User can provide the Git URL, Zip folder and app path to install custom apps. User can provide multiple URL's or Zip folders or app path given at same time under JSON file, it will install all the apps.

enable_fujitsu_logo : Provide the true/false under this field.

True: It will display the Fujitsu logo at the right side of the Navigation bar.

False: It will hide the Fujitsu logo.

Default value is True.

enable_utc_clock : Provide the true/false under this field.

True: It will display the UTC clock in the Navigation bar.

False: It will hide the UTC clock.

Default value is True.

WORKFLOW:

Katanaframework can be installed in pip installable way

1. Doesn't want to configure the custom app through JSON file, run the below command to get normal default/native apps.

- **manage.py runserver**

2. Wants to configure the custom app through JSON file, first priority will be the below command to run. After running below command, wait until server terminates automatically.

- **appmanage.py appconfig JSONPATH**

EX: Python3 appmanage.py appconfig
/home/rpenjuri/Downloads/userconfig.json

3. After custom app configuration by using above command(2), run the below command to bring up the UI.

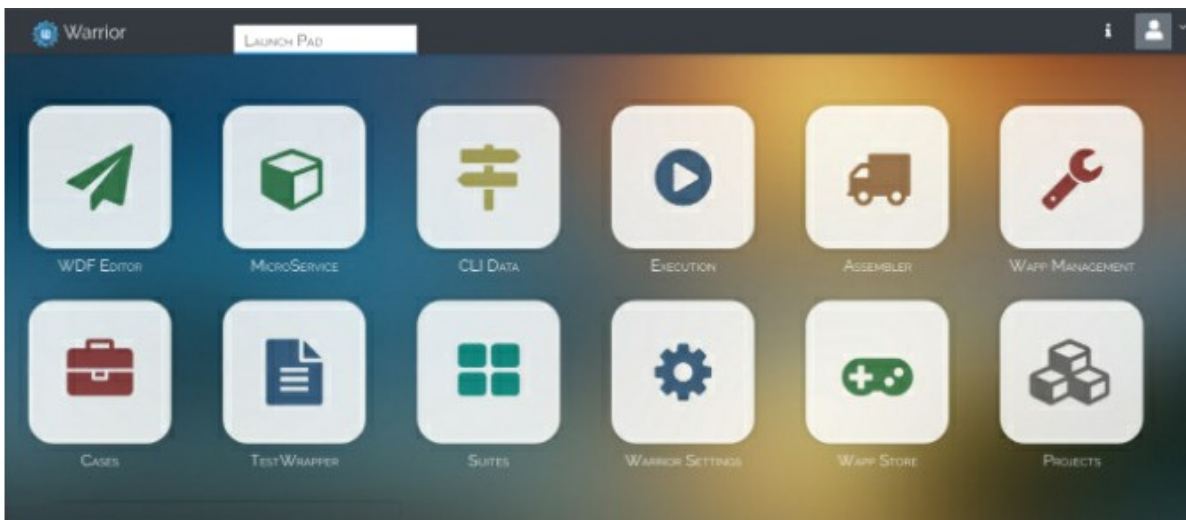
- **manage.py runserver**

Below example screenprints will gives clarity for the difference between default apps and custom apps.

Scenario1: *Default apps with develop branch*

```
{  
  "app_title": "",  
  "custom_logo_path": "",  
  "panel_color": "",  
  "katana_default_apps": "true",  
  "apps_rely_on_postgresdb": [],  
  "katana_default_apps_branch": "develop",  
  "user_custom_apps":{  
  }  
}
```

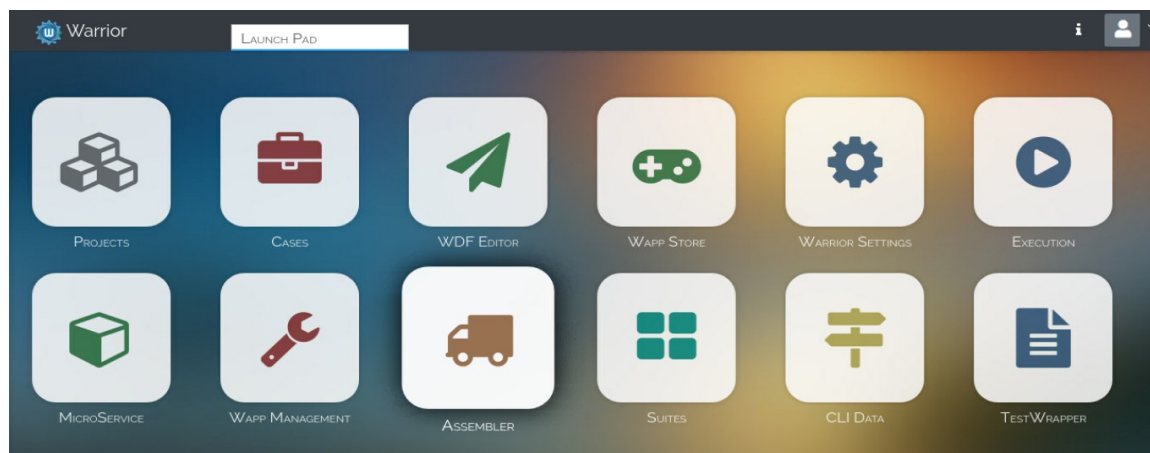
This will installs all default apps by taking the develop branch



Scenario2: *Default apps with master branch*

```
{  
  "app_title": "",  
  "custom_logo_path": "",  
  "panel_color": "",  
  "katana_default_apps": "true",  
  "apps_rely_on_postgresdb": [],  
  "katana_default_apps_branch": "",  
  "user_custom_apps": {  
  }  
}
```

This will install all default apps by considering the master branch



Scenario3: *Custom app configuration with git link*

```
{  
  "app_title": "Welcome",  
  "custom_logo_path": "/home/rpenjuri/Downloads/testing.jpg",  
  "panel_color": "Green",  
  "katana_default_apps": "false",  
  "apps_rely_on_postgresdb": [],  
  "katana_default_apps_branch": "",  
  "user_custom_apps": {  
    "sample_app": "https://github.com/terrakom/sample_app.git"  
  }  
}
```

This will install the custom app by considering the all configuration.

Note: Wapp management app became native app and by default, with your

configuration app, Wapp will reflect in UI



Scenario4: Custom app configuration by user provided path under user_custom_apps

```
{
  "app_title": "Welcome",
  "custom_logo_path": "/home/rpenjuri/Downloads/testing.jpg",
  "panel_color": "Green",
  "katana_default_apps": "false",
  "apps_rely_on_postgresdb": [],
  "katana_default_apps_branch": "",
  "user_custom_apps": {
    "container": "/home/rpenjuri/Downloads/container"
  }
}
```

This will install the custom app by considering the all configuration.

Note: Wapp management app became native app and by default, with your configuration app, Wapp will reflect in UI

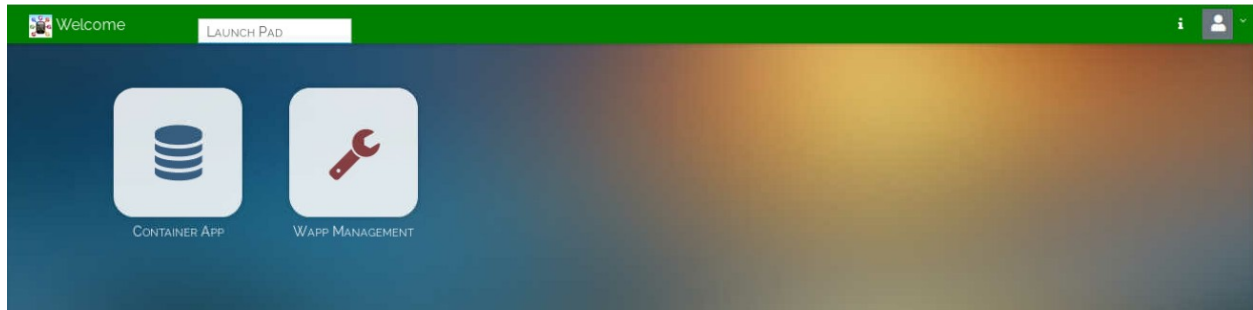


Scenario5: Custom app configuration by Zip folder path under user_custom_apps

```
{
  "app_title": "Welcome",
  "custom_logo_path": "/home/rpenjuri/Downloads/testing.jpg",
  "panel_color": "Green",
  "katana_default_apps": "false",
  "apps_rely_on_postgresdb": [],
  "katana_default_apps_branch": "",
  "user_custom_apps":{
    "container":"/home/rpenjuri/Downloads/container.zip"
  }
}
```

This will install the custom app by considering the all configuration.

Note: Wapp management app became native app and by default, with your configuration app, Wapp will reflect in UI



User can mention all three formats(zip, git and location path) under user_custom_apps at same time to install the multiple apps at same time.

Note: Only **zip format** is allowed in json file(Don't mention tar or with other extensions)

VERSIONING:

Version number is incremented after every release and apps versions will also changes. After user will configure any app through configurable app feature and wants to switch to other version, provided an option to upgrade or downgrade the app through CLI.

Workflow:

1. If you have already configured an app and trying to upgrade or downgrade the version, use the below command. It will ask user in CLI

to reinstall same app with different version with the options (Y/n). If user enters “Y”, reinstall will happen and if enters “n”, reinstallation will skip and proceed with old configuration.

- **appmanage.py appconfig JSONPATH**

Note: If user doesn't provide any input in 10 seconds, it will take default value.

Default Values

Upgrade and downgrade: Y
Installing same app: N

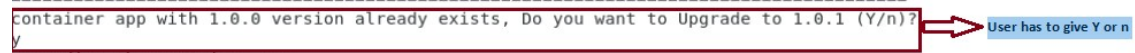
2. After configuration by using above command(1), run the below command to bring up the UI.

- **manage.py runserver**

Below scenarios will gives clarity on upgrading and downgrading the app versions

Scenario1: *Upgrading already configured or installed app through CLI*

```
(doc) rpenjuri :~/Documents/doc/Katanaframework/katana> $ appmanage.py appconfig /home/rpenjuri/Downloads/userconfig.json
=====
configuring katana apps...
Please do not quit (or) kill the server manually, wait until the server closes itself...!
=====
container app with 1.0.0 version already exists, Do you want to Upgrade to 1.0.1 (Y/n)?
y
Upgrading the container app
Checking Compatability for:container
container app installed successfully.
[100%]
Done!
```



After configuration as shown in above screenprint, run the server by using “**manage.py runserver**” command to bring up UI for checking upgraded app.

Scenario2: *Downgrading already configured or installed app through CLI*

```
(doc) rpenjuri :~/Documents/doc/Katanaframework/katana> $ appmanage.py appconfig /home/rpenjuri/Downloads/userconfig.json
=====
configuring katana apps...
Please do not quit (or) kill the server manually, wait until the server closes itself...!
=====
container app with 1.0.1 version already exists, Do you want to Downgrade to 1.0.0 (Y/n)?
y
Downgrading the container app
Checking Compatability for:container
container app installed successfully.
[100%]
Done!
```

After configuration as shown in above screenprint, run the server by using **“manage.py runserver”** command to bring up UI for checking downgraded app.

Scenario3: *Configuring same version app again*

```
(doc) rpenjuri :~/Documents/doc/Katanaframework/katana> $ appmanage.py appconfig /home/rpenjuri/Downloads/userconfig.json
=====
configuring katana apps...
Please do not quit (or) kill the server manually, wait until the server closes itself...!
=====
container app with the same version (1.0.0) already exists, Do you want to Reinstall (Y/n)?
n
Reinstall: Skipped
[100%]
Done!
```

After configuration as shown in above screenprint, run the server by using **“manage.py runserver”** command to bring up UI for checking app.

USAGE OF CONFIGURABLE APP:

- Provides an option to the developer to include only custom app without any default apps
- Developer can customize company name, logo, color (This replaces existing warrior logo, name)
- When katana is started, apps will be installed as per the configuration in this file.
- Will decrease the code maintenance issues
- Will decrease the multiple code repositories for each katana app