# Table of Contents

Table of Contents 1

ASPARK-STARTER 2

1. WHAT IS IT ?! 2
2. INSTALLATION AND CONFIGURATION 2
   - 2.1. Fetch the source 2
   - 2.2. Prerequisites 2
   - 2.3. Ensure you have all the prerequisite binaries 2
   - 2.4. Build the first aspark-starter instance 2
   - 2.5. Check the runnable actions 3
   - 2.6. Run the examples 3
   - 2.7. Start hacking 3
3. PROJECT STATUS 4

# ASPARK-STARTER

## 1. WHAT IS IT ?!

A demo application which will help you to grasp the Apache Spark concept.

## 2. INSTALLATION AND CONFIGURATION

This section presents the steps needed to perform to deploy the aspark-starter tool. Note, that the commands are for Ubuntu, thus if you are on different OS choose , google the names of the packages applicable for your OS.

### 2.1. Fetch the source

Fetch the source from git hub as follows:

```
# create your product dir:
mkdir -p /opt/csitea/
cd /opt/csitea/

# fetch the source
git clone git@github.com:YordanGeorgiev/aspark-starter.git

# DO NOT CD into the new dir !!!!
```

### 2.2. Prerequisites

The must have binaries are:
bash, perl, zip
The nice to have are:
tmux, vim ,ctags
The examples are for Ubuntu - use you OS package manager …

```
apt-get autoclean
apt-get install --only-upgrade bash
sudo apt-get install -y perl



apt-get upgrade
```

### 2.3. Ensure you have all the prerequisite binaries

Ensure you have all the prerequisite binaries by issuing the following command

```
# bootstrap the product instance dir
bash aspark-starter/src/bash/aspark-starter/install-prerequisites-for-aspark-starter-on-ubuntu.sh
```

### 2.4. Build the first aspark-starter instance

Build the aspark-starter instance by running the bootstrap script

```
# bootstrap the product instance dir
```

```
bash aspark-starter/src/bash/aspark-starter/bootstrap-aspark-starter.sh

# the script should prompt you to cd
```

## 2.5. Check the runnable actions

You could check the functions which could be run - aka "actions" by issuing the following command.

```
# check the runnable with the -a cmd arg actions
find . -name '*.func.sh' | sort
```

## 2.6. Run the examples

You can run all the examples by first checking which actions are configured for the next test run and perform the actual test run as follows:

```
# check the actions to run
  cat src/bash/aspark-starter/tests/run-aspark-starter-tests.lst

# STDUOT
# sbt-compile-verbose
# sbt-clean-compile
# sbt-compile
# sbt-stage
# sbt-run

bash src/bash/aspark-starter/test-aspark-starter.sh

# now the tool will start producing output

# 2017-09-14 08:26:11      START test-aspark-starter test run report
#
# result  start-time  stop-time   action-name
#    ok    08:26:11 08:26:59 sbt-compile-verbose
#    ok    08:27:00 08:27:25 sbt-clean-compile
#    ok    08:27:25 08:27:34 sbt-compile
#    ok    08:27:35 08:27:49 sbt-stage
#    ok    08:27:49 08:27:59 sbt-run
```

## 2.7. Start hacking

Start hacking … or wait check at least the test call running all the functions of the tool …

```
# opionally if you are in the vim camp open the "project relative files list file"
vim meta/.dev.aspark-starter

# Ctrl + Z , to put it on the backgound
# check the actions to test ( uncoment line in include in test run )
less src/bash/aspark-starter/tests/run-aspark-starter-tests.lst

# Ctrl + Z to put in the background
# Action  !!! - aka now run the tests
bash src/bash/aspark-starter/test-aspar-starter.sh
```

## 3. PROJECT STATUS

You could track the advancement of the project from the following url:
https://docs.google.com/spreadsheets/d/e/2PACX-1vR0wo5N32EpubwxBfeFxi6X-eOmXwOPg4WSyA4qBSz1Yu0EyU34jl0xICgWzrFUSeEA_aC4RF7LRqx9/pubhtml
Note that the content on the url is updated on project actual status update ( i.e. meaningful work , milestones achieve )