Platys – Platform in a box

Guido Schmutz, Ulises Fasoli

gschmutz

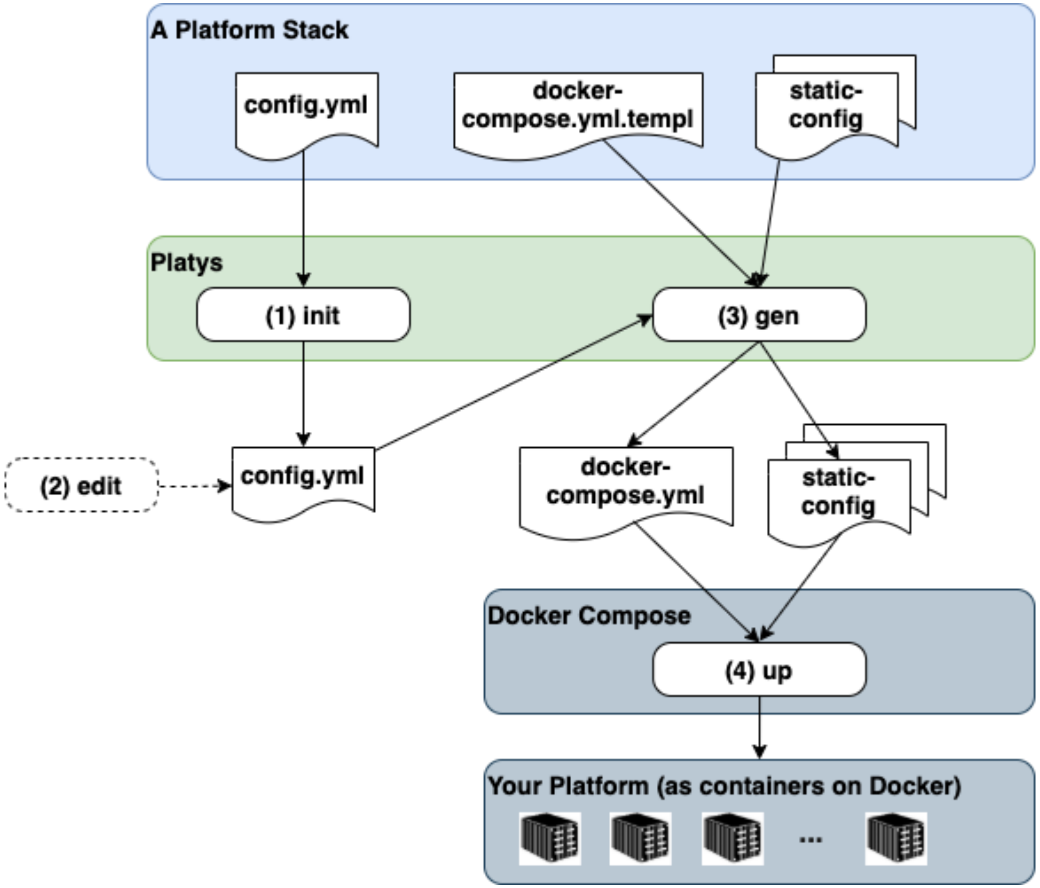
http://guidoschmutz.wordpress.com

BASEL I BERN I BRUGG I BUKAREST I DÜSSELDORF I FRANKFURT A.M. I FREIBURG I.BR. I GENF
HAMBURG I KOPENHAGEN I LAUSANNE I MANNHEIM I MÜNCHEN I STUTTGART I WIEN I ZÜRICH

trivadis

Trivadis `platys` – Data Platform in a Box

- a tool for generating and provisioning Modern Data Platforms based on Docker and Docker Compose
- main use is for small-scale Data Lab projects, Proof-of-Concepts (PoC) or Proof-of-value (PoV) projects as well as trainings
- A concrete *Platform* is always generated based on a given *Platform Stack*
- See here for how to install `platys`
- https://github.com/TrivadisPF/platys



gschmutz

Trivadis `platys` – How does it work?

1. Create a folder as your platform working directory

```
mkdir platys-demo
dd platys-demo
```

2. Initialize the platform using the `platys init` command

```
platys init -n platys-demo-platform
     -sn trivadis/platys-modern-data-platform -sv 1.2.0
```

3. A `config.yml` file is created locally, holding all the possible options of the modern-data-platform 1.2.0 stack. See here for a documentation of the various options.

Trivadis platys – How does it work?

4. Use an editor to set the needed services to true.

```
# Default values for the generator
# this file can be used as a template for a custom configuration
# or to know about the different variables available for the generator
  platys:
    platform-name: ‘platys-demo'
    stack-image-name: 'trivadis/platys-modern-data-platform'
    stack-image-version: '1.2.0'

  # ===== Apache Zookeeper ========
  KAFKA_enable: true
  # one of enterprise, community
  KAFKA_edition: 'community'
  KAFKA_volume_map_data: false
  KAFKA_broker_nodes: 3
  KAFKA_delete_topic_enable: false
  KAFKA_auto_create_topics_enable: false
```

Trivadis `platys` – How does it work?

5. Run `platys gen` to generate your platform

```
platys gen
```

6. A folder with the name of the platform is created and in which all the artefacts for the Docker Compose stack can be found

7. Navigate into that folder and start the stack using `docker-compose up`

```
cd platys-demo-platform
docker-compose up -d
```

Trivadis `platys` – Tips and Tricks

trivadis

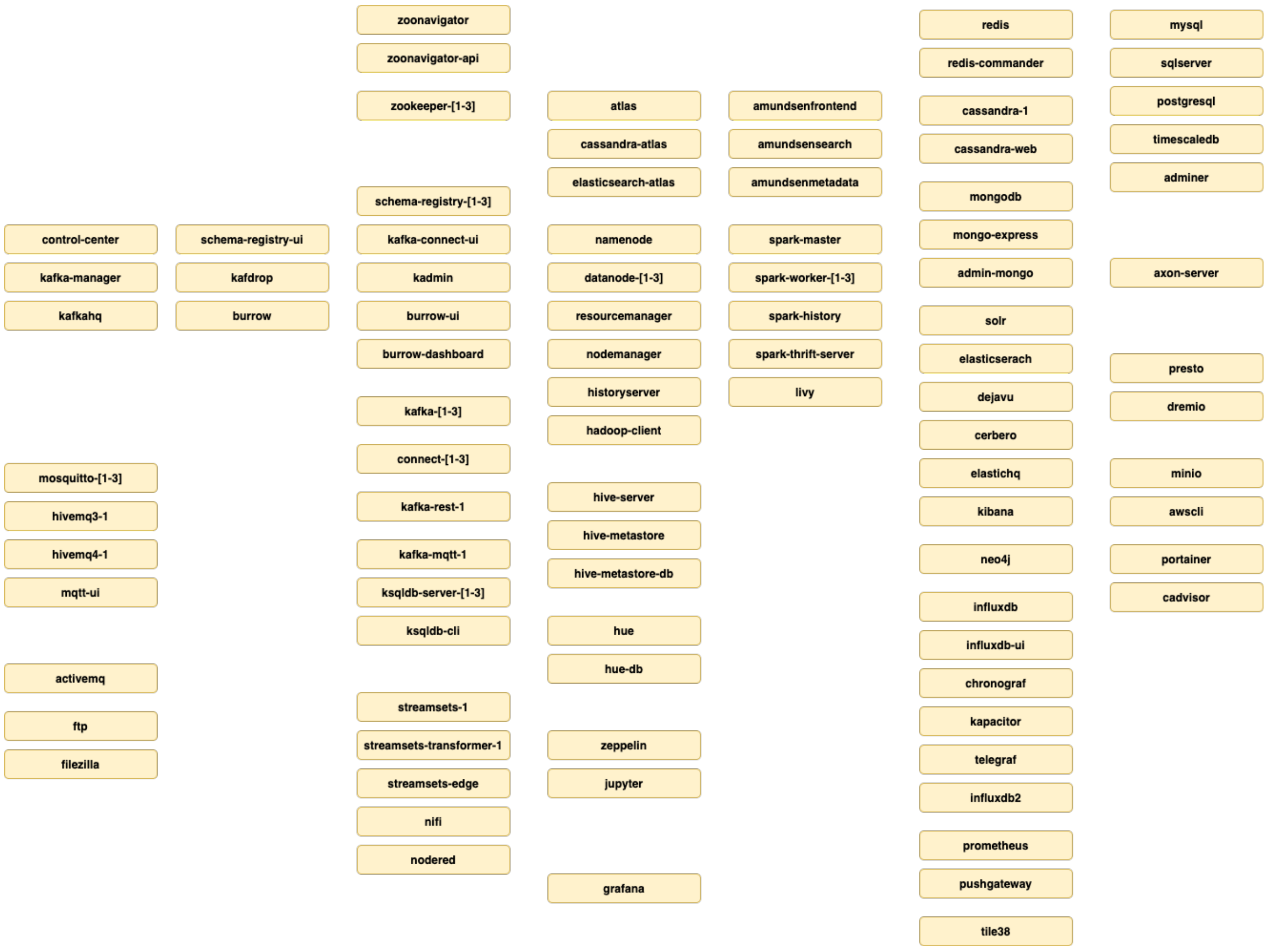
- In the init you can also pass a list of services to directly set to true to omit having to edit the `config.yml`

```
platys init -n platys-demo-platform
     -sn trivadis/platys-modern-data-platform -sv 1.2.0
     -f -s KAFKA,SPARK,JUPYTER
```

- You can list the available service of a given platform stack by using the `list_services` command

```
platys list_services -sn trivadis/platys-modern-data-platform
                     -sv 1.2.0
```

modern-data-platform – Service Overview trivadis



control-center
kafka-manager
kafkahq

mosquitto-[1-3]
hivemq3-1
hivemq4-1
mqtt-ui

activemq

ftp
filezilla

schema-registry-ui
kafdrop
burrow

zoonavigator
zoonavigator-api

zookeeper-[1-3]

schema-registry-[1-3]
kafka-connect-ui
kadmin
burrow-ui
burrow-dashboard

kafka-[1-3]

connect-[1-3]

kafka-rest-1

kafka-mqtt-1
ksqldb-server-[1-3]
ksqldb-cli

streamsets-1
streamsets-transformer-1
streamsets-edge
nifi
nodered

atlas
cassandra-atlas
elasticsearch-atlas

namenode
datanode-[1-3]
resourcemanager
nodemanager
historyserver
hadoop-client

hive-server
hive-metastore
hive-metastore-db

hue
hue-db

zeppelin
jupyter

grafana

amundsenfrontend
amundsensearch
amundsenmetadata

spark-master
spark-worker-[1-3]
spark-history
spark-thrift-server
livy

redis
redis-commander

cassandra-1
cassandra-web

mongodb
mongo-express
admin-mongo

solr
elasticserach
dejavu
cerbero
elastichq
kibana

neo4j

influxdb
influxdb-ui
chronograf
kapacitor
telegraf
influxdb2

prometheus
pushgateway

tile38

mysql
sqlserver
postgresql
timescaledb
adminer

axon-server

presto
dremio

minio
awscli

portainer
cadvisor

gschmutz

Making a **WORLD** possible in which **intelligent IT** facilitates **LIFE** and **WORK** as a **matter** of **course.**