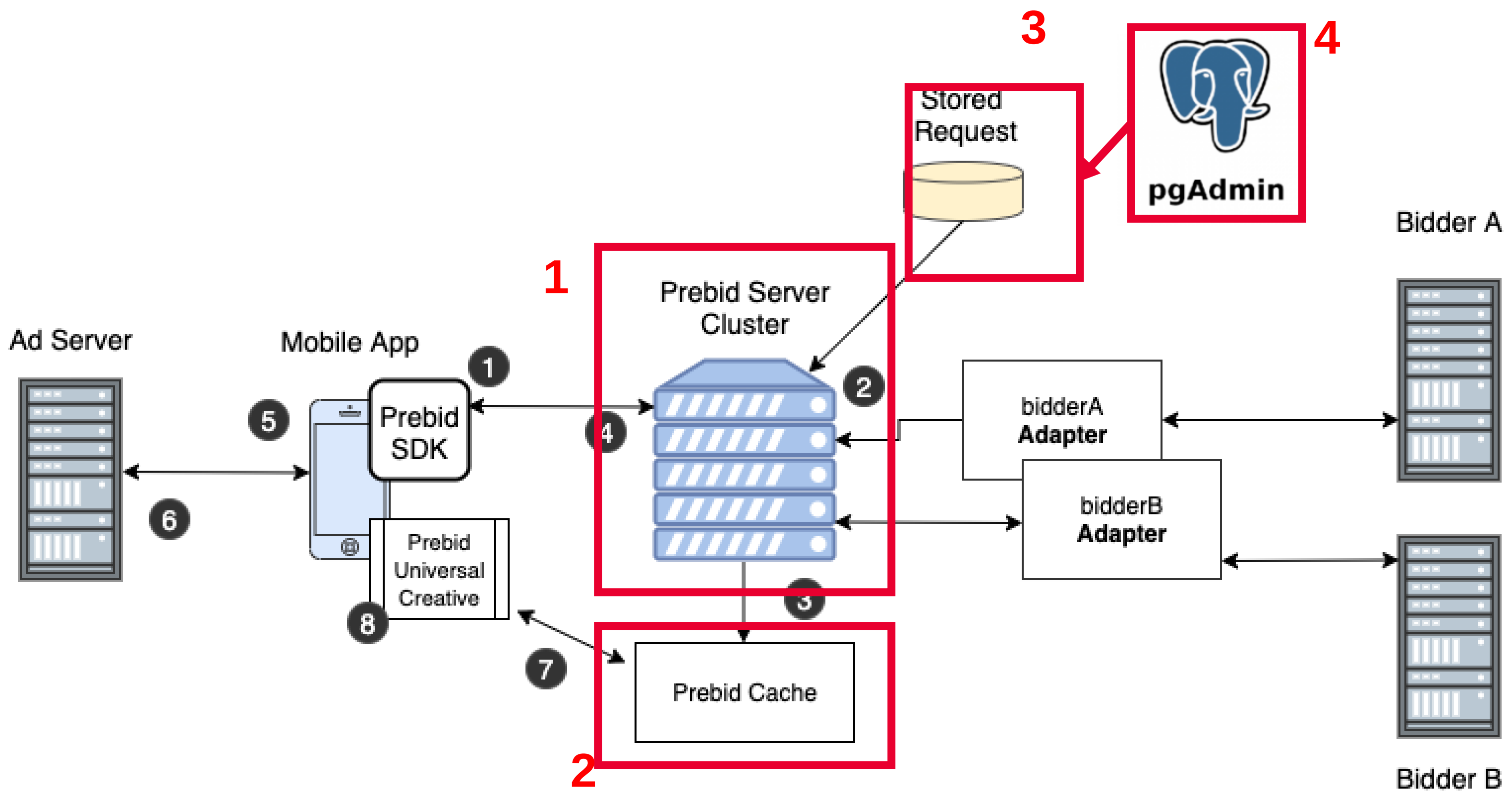
# Setup Prebid Server and Prebid Cache

**Department name: Consumer Cloud Service European Operations Center (Germany GBG)**
**Author's name: Chia Leung Ho c00585749**
**Date: 05 May 2023**

Security Level:

## Contents

# 1. Prebid Architecture

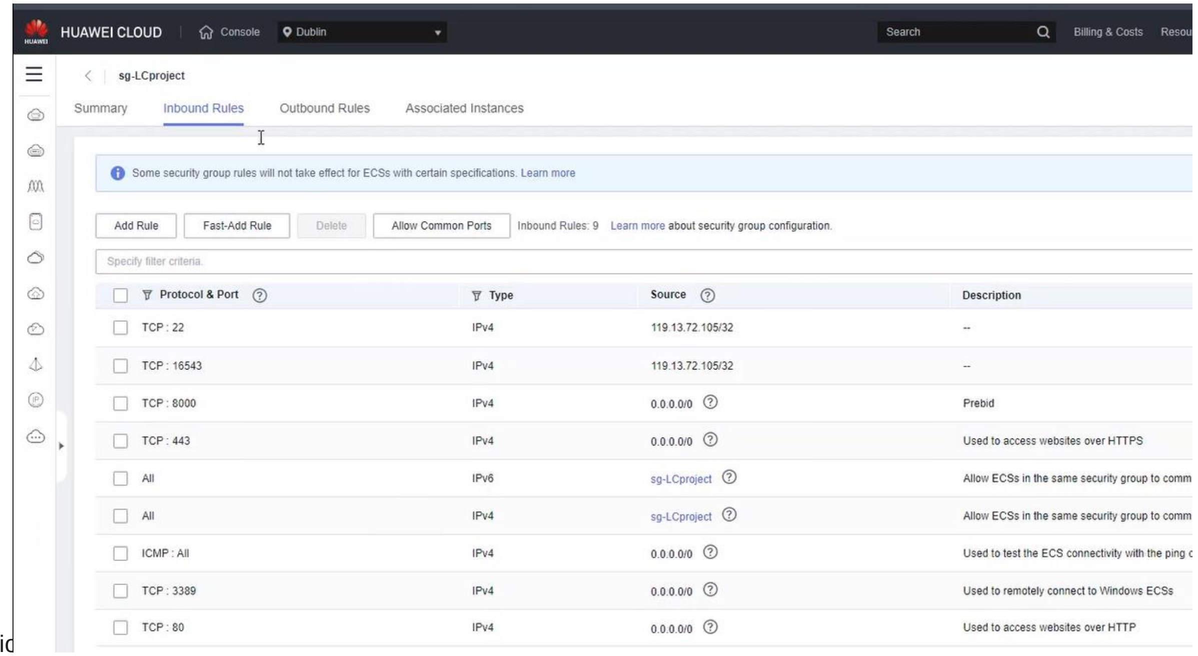


We are going to need:

1. Prebid Server
2. Prebid Cache
3. Postgres Database to store these data:
    - Stored_Imp
    - Stored_Req
    - Stored_Response
4. PgAdmin to view the data. **This is not needed for the actual production environment.**

# 2. Cloud Server configuration

- Choose your own public cloud provider (eg: AWS, Microsoft Azure, Google Cloud Platform, Huawei Cloud)
- Make sure that the following ports are opened for inbound traffic
  - > Port 22 – For SSH (Not needed for production environment, bind to your own Public IP address)
  - > Port 8000 – For Prebid Server
  - > Port 16543 – For PgAdmin (Not needed for production environment)



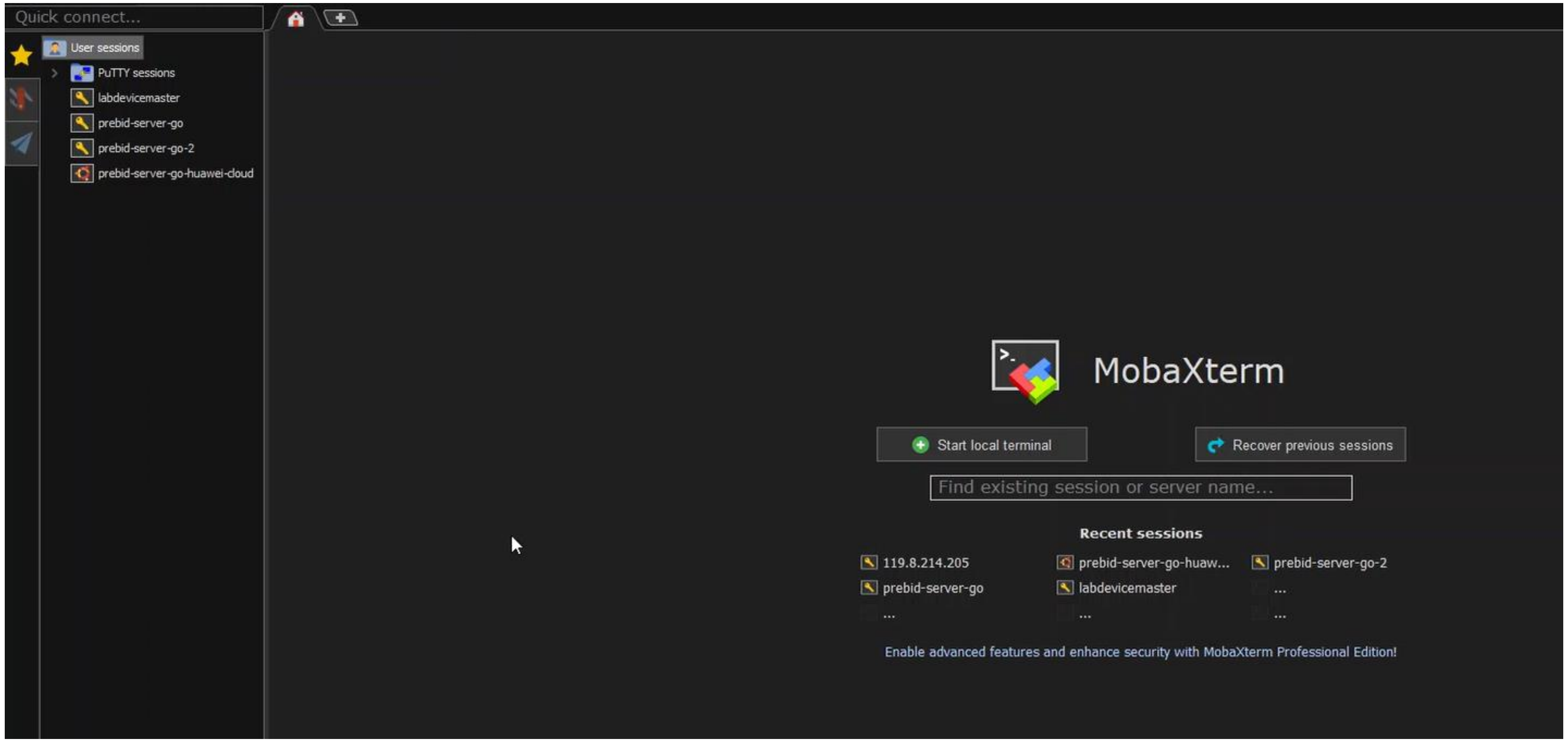
Some security group rules will not take effect for ECSs with certain specifications. Learn more

Inbound Rules: 9 Learn more about security group configuration.

| Protocol & Port | Type | Source | Description |
|---|---|---|---|
| TCP : 22 | IPv4 | 119.13.72.105/32 | -- |
| TCP : 16543 | IPv4 | 119.13.72.105/32 | -- |
| TCP : 8000 | IPv4 | 0.0.0.0/0 | Prebid |
| TCP : 443 | IPv4 | 0.0.0.0/0 | Used to access websites over HTTPS |
| All | IPv6 | sg-LCproject | Allow ECSs in the same security group to comm |
| All | IPv4 | sg-LCproject | Allow ECSs in the same security group to comm |
| ICMP : All | IPv4 | 0.0.0.0/0 | Used to test the ECS connectivity with the ping c |
| TCP : 3389 | IPv4 | 0.0.0.0/0 | Used to remotely connect to Windows ECSs |
| TCP : 80 | IPv4 | 0.0.0.0/0 | Used to access websites over HTTP |

## 3. SSH Client

- Feel free to use your favourite SSH Client. MobaXterm is used in this tutorial.



## 4. Install docker and docker-compose (Skip if you already have it installed)

- Refer to https://docs.docker.com/engine/install/ubuntu/
- Optionally, refer to the following sample script (For Ubuntu).

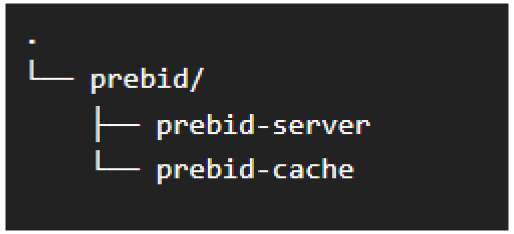
```
sudo apt update -y
sudo apt upgrade -y
sudo apt-get install ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose
```

## 5. Clone Prebid Server and Prebid Cache from GitHub

- Prebid Server official repository - https://github.com/prebid/prebid-server
- Prebid Cache official repository - https://github.com/prebid/prebid-cache
- Create the necessary folders:

```
mkdir prebid
cd prebid
mkdir prebid-server
mkdir prebid-cache
```

- Your folder structure should look like this:



```
.
└── prebid/
    ├── prebid-server
    └── prebid-cache
```

- cd into the corresponding directory and clone from GitHub

```
cd prebid-cache
git clone https://github.com/prebid/prebid-cache.git .
cd ..
cd prebid-server
git clone https://github.com/prebid/prebid-server.git .
```

## 6. Modify Prebid Cache Dockerfile

- Replace the content of `prebid/prebid-cache/Dockerfile` with the following Docker script:

```
# syntax=docker/dockerfile:1
FROM golang:1.18.1-alpine3.15
RUN apk add git
WORKDIR /app
COPY go.mod ./
COPY go.sum ./
RUN go mod download
COPY . ./
RUN go build .
EXPOSE 8000
CMD ["./prebid-cache"]
```

# 7. Modify and create Prebid Server configuration files

- Replace the content of `prebid/prebid-server/Dockerfile` with the following Docker script:

```
# syntax=docker/dockerfile:1
FROM golang:1.18.1-alpine3.15
RUN apk add git
WORKDIR /app
COPY go.mod ./
COPY go.sum ./
RUN go mod download
COPY . ./
RUN go build .
EXPOSE 8000
CMD ["./prebid-server"]
```

## 7. Modify and create Prebid Server configuration files

- Create 3 new files in prebid/prebid-server
  - > docker-compose.yaml
  - > pbs.yaml
  - > database-seed.sql

- Replace their content with their corresponding script, refer to the next page.

## docker-compose.yaml

```yaml
version: '3'
services:
  prebid-server:
    build: .
    depends_on:
      - database
    ports:
      - 8000:8000
    volumes:
      - .:/usr/src/prebid-server
    container_name: prebid-server
    networks:
      - prebidnet
  prebid-cache:
    build: /root/prebid/prebid-cache/.
    ports:
      - 2424:2424
    volumes:
      - .:/usr/src/prebid-cache
    container_name: prebid-cache
    networks:
      - prebidnet
  database:
    image: postgres
    ports:
      - 5432:5432
    volumes:
      - ./database-seed.sql:/docker-entrypoint-initdb.d/database-seed.sql
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=prebid
    container_name: prebid-server-database
    networks:
      - prebidnet
    hostname: postgres
  pgadmin:
    image: dpage/pgadmin4
    logging:
      driver: none
    environment:
      PGADMIN_DEFAULT_EMAIL: "test@huawei.com"
      PGADMIN_DEFAULT_PASSWORD: "@@Huawei2023!!"
    ports:
      - "16543:80"
    depends_on:
      - database
    container_name: prebid-server-pgadmin
    networks:
      - prebidnet
volumes:
  pgdata:
networks:
  prebidnet:
```

Prebid server container, listening at port 8000, container built based on the Dockerfile in `prebid/prebid-server/Dockerfile`

Prebid cache container, listening at port 2424, container built based on the Dockerfile in `prebid/prebid-cache/Dockerfile`

Database container, listening at port 5432, container built based on the `postgres` Docker image, initialized with the script database-seed.sql to create the necessary tables and insert sample records into the tables

Pgadmin container, listening at port 16543, container built based on the `pgadmin4` Docker image. This container is not needed in the actual production environment. Included for debug purposes.

## pbs.yaml

```
gdpr:
  enabled: true
  default_value: "0"
  timeouts_ms:
    init_vendorlist_fetches: 5000
    active_vendorlist_fetch: 5000
  tcf2:
    enabled: true
stored_requests:
  postgres:
    connection:
      host: database
      port: 5432
      user: postgres
      password: postgres
      dbname: prebid
    fetcher:
      query: SELECT id, requestData, 'request' as type FROM stored_requests WHERE id in (%REQUEST_ID_LIST%) UNION ALL SELECT id,
impData, 'imp' as type FROM stored_imps WHERE id in (%IMP_ID_LIST%);
stored_responses:
  postgres:
    connection:
      host: database
      port: 5432
      user: postgres
      password: postgres
      dbname: prebid
    fetcher:
      query: SELECT id, responseData as data, 'response' as dataType FROM stored_responses WHERE id in (%ID_LIST%);
adapters:
  huaweiads:
    disabled: false
cache:
  host: 'prebid-cache:2424'
  scheme: 'http'
```

# database-seed.sql

```sql
CREATE TABLE stored_requests
(
    id uuid DEFAULT gen_random_uuid() PRIMARY KEY,
    requestdata text,
    created_on timestamp with time zone NOT NULL DEFAULT now()
);

insert into stored_requests(id, requestdata) values ('06489cdd-f544-4a61-b377-be6e9b5050f5','{ "cur": ["USD"], "ext": { "prebid": { "cache": { "bids": {} }, "targeting": { "includewinners": true, "pricegranularity": {
"ranges": [{ "max": 20, "increment": 0.01 }], "precision": 2 }, "includebidderkeys": true } } }, "events": { "enabled": true }, "disabled": false, "cache_ttl": { "audio": 3600, "video": 3600, "banner": 600, "native": 3600 }
}');

CREATE TABLE stored_imps
(
    id uuid DEFAULT gen_random_uuid() PRIMARY KEY,
    impdata text,
    created_on timestamp with time zone NOT NULL DEFAULT now()
);
insert into stored_imps(id, impdata) values ('cbbd553d-179e-430b-9d0a-f5bd4e416822','{ "banner": { "format": [ { "w": 300, "h": 250 }, { "w": 300, "h": 600 } ] }, "ext": { "prebid": { "bidder": { "appnexus": { "placement_id":
12883451 } } } } }');

CREATE TABLE stored_responses
(
    id uuid DEFAULT gen_random_uuid() PRIMARY KEY,
    responsedata text,
    created_on timestamp with time zone NOT NULL DEFAULT now()
);
insert into stored_responses(id, responsedata) values ('532042f9-46de-4478-8408-836d9fe4627f','[{
                 "bid": [{
                                   "id": "06489cdd-f544-4a61-b377-be6e9b5050f5",
                                   "impid": "06489cdd-f544-4a61-b377-be6e9b5050f5",
                                   "price": 5,
                                   "adm": "<style> html, body  { margin: 0; padding: 0; width: 100%; height: 100%; vertical-align: middle; }  html  { display: table; }  body { display: table-cell; vertical-align: middle; text-
align: center; -webkit-text-size-adjust: none; }  </style> <span class=\"title-link advertiser_label\"></span> <a href=\"https://appgallery.huawei.com/#/app_simple/C105624843\" style=\"text-decoration:none\"
onclick=sendGetReq()> <img src=\"https://cs02-pps-dre.dbankcdn.com/dl/pps/20220721003915BDFBAF3609CB2CE3F5BA88BA7B4B0D1F.jpg\" width=\"320\" height=\"50\"/> </a> <img height=\"1\" width=\"1\" src=\"https://events-
dre.op.hicloud.com/contserver/tracker/action?ch=200002&etype=imp&kn=2&pfsa=FNe7ribdxb4Tujdm8SiapJXFdicPandGRiatia4OGHTe9jSqH2Aldvf9IYrNSGfe529EsBzPJypDDZOiannSO8FMlxL3bNQCMia0u96G4IcOqDwAREKE4svvg20YmdeWWp6p1kicUw5ZNtZsFjkkW1
SfGMrG0uGSyMeZWDRGKS8iaw3xicyy7qib7FsXCmFibzpkOvwoNNKdLrnOA7Mbv8ibZFtic8QPRjRvbIsr0SccsVOKnwstibO85UtEicFq6l0zRef5MsydVtwibyzDkoQkQctiaCo5icVib6rwJDlBACZgMMxofDGzuTnbEiby3wpiaQAZ2EgRY3CuCRJhkHUvqiaJSBTLD1PAdLtLY6bPohDyN6MaTkJ
n3KmRmicmBHPcC78DDmwgRybtrc1Mu27POhpeVYa1Eh3nB8PNSwAymGMe7GLsmfrZF1vJQjKRj4czl8CjElbdKxXvdesKwxvsWeE8UcXibfAFjdOtsUXH8Nmww7pp2zuwMBiaX3hX6WTpJVVj0rogEeJR48S9JQkCpGA9Bmgx1bQYsdoYjYAHc2GqRXPGDoXZfFqmicsfcZ0EicQSxiayjiblicS9d09b
8eSslG19A2ibOWeJCMjzN7hXHkfqaP2JwkM9HGTOdJgygXxvSn9kvnau8b5C3DYF8lnD49mTiaTiaJsAdyry9WF4AiccBRZHKkDUOP9sGHM7kichic82WAr1Mu5U3PUZ5LwJfkpQdzPJTLD00MrYEwiaZLSFicUah0QkyzapbMuicicL9vKy8HiavyHBslqvHZOquDdzt3uv2Qt7YniafwduA&uuid=_U
UID_\" >  <script type=\"text/javascript\">var dspClickTrackings = [\"https://events-
dre.op.hicloud.com/contserver/tracker/action?ch=200002&etype=click&kn=2&pfsa=FNe7ribdxb4Tujdm8SiapJXFdicPandGRiatia4OGHTe9jSqH2Aldvf9IYrNSGfe529EsBzPJypDDZOiannSO8FMlxL3bNQCMia0u96G4IcOqDwAREKE4svvg20YmdeWWp6p1kicUw5ZNtZsFjkk
W1SfGMrG0uGSyMeZWDRGKS8iaw3xicyy7qib7FsXCmFibzpkOvwoNNKdLrnOA7Mbv8ibZFtic8QPRjRvbIsr0SccsVOKnwstibO85UtEicFq6l0zRef5MsydVtwibyzDkoQkQctiaCo5icVib6rwJDlBACZgMMxofDGzuTnbEiby3wpiaQAZ2EgRY3CuCRJhkHUvqiaJSBTLD1PAdLtLY6bPohDyN6MaT
kJn3KmRmicmBHPcC78DDmwgRybtrc1Mu27POhpeVYa1Eh3nB8PNSwAymGMe7GLsmfrZF1vJQjKRj4czl8CjElbdKxXvdesKwxvsWeE8UcXibfAFjdOtsUXH8Nmww7pp2zuwMBiaX3hX6WTpJVVj0rogEeJR48S9JQkCpGA9Bmgx1bQYsdoYjYAHc2GqRXPGDoXZfFqmicsfcZ0EicQSxiayjiblicS9d0
9b8eSslG19A2ibOWeJCMjzN7hXHkfqaP2JwkM9HGTOdJgygXxvSn9kvnau8b5C3DYF8lnD49mTiaTiaJsAdyry9WF4AiccBRZHKkDUOP9sGHM7kichic82WAr1Mu5U3PUZ5LwJfkpQdzPJTLD00MrYEwiaZLSFicUah0QkyzapbMuicicL9vKy8HiavyHBslqvHZOquDdzt3uv2Qt7YniafwduA&uuid=
_UUID_&w=__HW_W__&h=__HW_H__&downx=__HW_DOWN_X__&downy=__HW_DOWN_Y__&upx=__HW_UP_X__&upy=__HW_UP_Y__\"];function sendGetReq() {sendSomeGetReq(dspClickTrackings)}function sendOneGetReq(url) {var req = new
XMLHttpRequest();req.open(\"GET\", url, true);req.send(null);}function sendSomeGetReq(urls) {for (var i = 0; i < urls.length; i++) {sendOneGetReq(urls[i]);}}</script>",
                                   "adomain": [
                                                    "huaweiads"
                                   ],
                                   "crid": "58025103",
                                   "w": 320,
                                   "h": 50,
                                   "mtype": 1
                 }],
                 "seat": "huaweiads"
}]');
```

## 8. Start the Prebid Server and Prebid Cache

- Make sure that you are in the `prebid/prebid-server` directory
- Run the following script

```
sudo docker-compose build
sudo docker-compose up -d
```

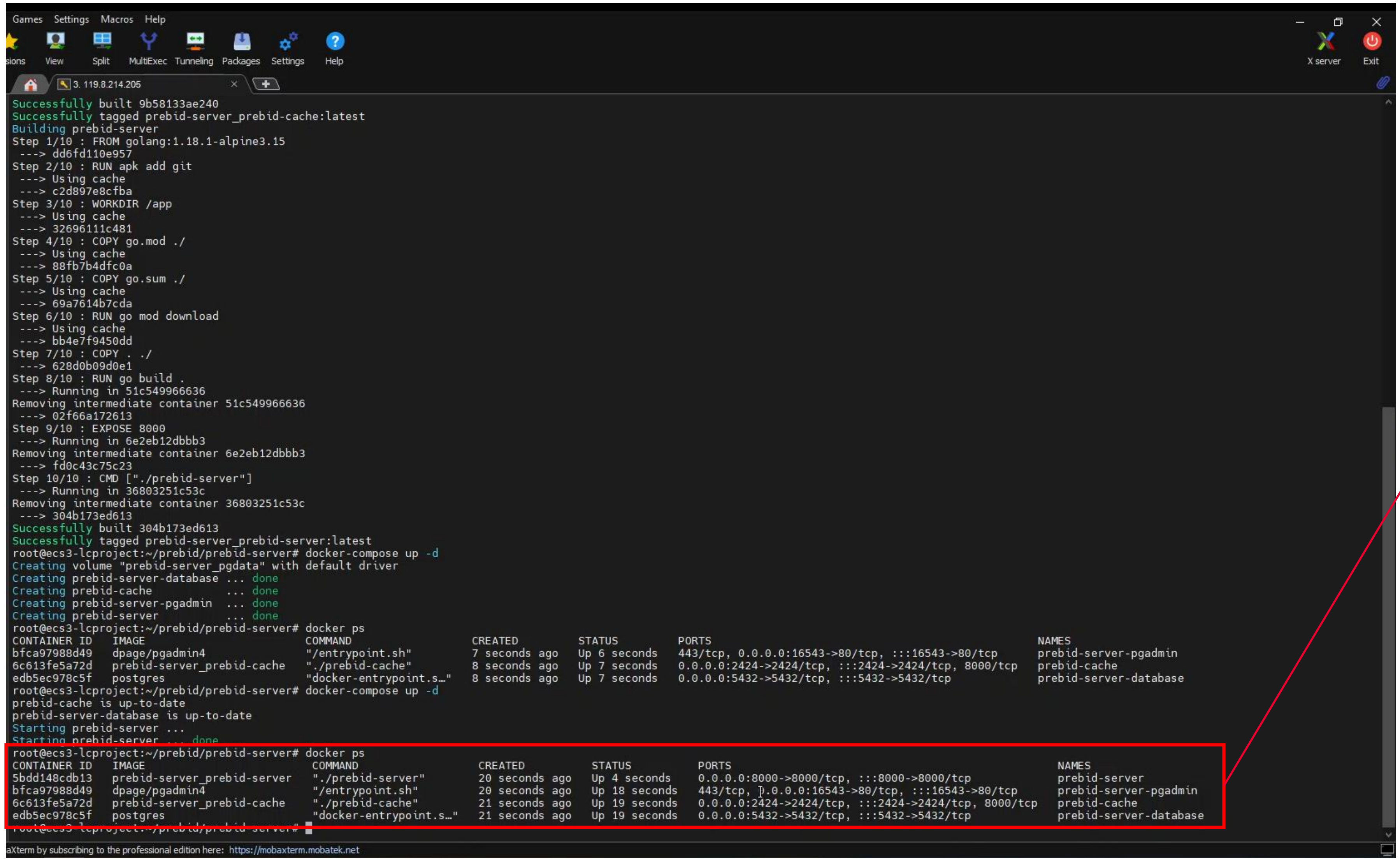
- Optionally, to check the logs, run

```
sudo docker logs --follow $(sudo docker ps -q --filter ancestor=prebid-server_prebid-server)
```

- Optionally, to stop and remove everything, run

```
sudo docker stop $(docker ps -a -q)
sudo docker rm $(docker ps -a -q)
sudo docker volume rm $(docker volume ls -q)
```

## 8. Start the Prebid Server and Prebid Cache

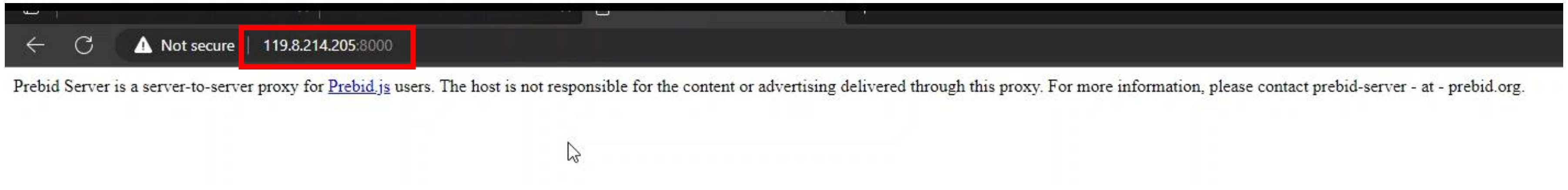


```
Successfully built 9b58133ae240
Successfully tagged prebid-server_prebid-cache:latest
Building prebid-server
Step 1/10 : FROM golang:1.18.1-alpine3.15
 ---> dd6fd110e957
Step 2/10 : RUN apk add git
 ---> Using cache
 ---> c2d897e8cfba
Step 3/10 : WORKDIR /app
 ---> Using cache
 ---> 32696111c481
Step 4/10 : COPY go.mod ./
 ---> Using cache
 ---> 88fb7b4dfc0a
Step 5/10 : COPY go.sum ./
 ---> Using cache
 ---> 69a7614b7cda
Step 6/10 : RUN go mod download
 ---> Using cache
 ---> bb4e7f9450dd
Step 7/10 : COPY . ./
 ---> 628d0b09d0e1
Step 8/10 : RUN go build .
 ---> Running in 51c549966636
Removing intermediate container 51c549966636
 ---> 02f66a172613
Step 9/10 : EXPOSE 8000
 ---> Running in 6e2eb12dbbb3
Removing intermediate container 6e2eb12dbbb3
 ---> fd0c43c75c23
Step 10/10 : CMD ["./prebid-server"]
 ---> Running in 36803251c53c
Removing intermediate container 36803251c53c
 ---> 304b173ed613
Successfully built 304b173ed613
Successfully tagged prebid-server_prebid-server:latest
root@ecs3-lcproject:~/prebid/prebid-server# docker-compose up -d
Creating volume "prebid-server_pgdata" with default driver
Creating prebid-server-database ... done
Creating prebid-cache           ... done
Creating prebid-server-pgadmin  ... done
Creating prebid-server          ... done
root@ecs3-lcproject:~/prebid/prebid-server# docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED         STATUS         PORTS                                                   NAMES
bfca97988d49   dpage/pgadmin4                "/entrypoint.sh"         7 seconds ago   Up 6 seconds   443/tcp, 0.0.0.0:16543->80/tcp, :::16543->80/tcp        prebid-server-pgadmin
6c613fe5a72d   prebid-server_prebid-cache    "./prebid-cache"         8 seconds ago   Up 7 seconds   0.0.0.0:2424->2424/tcp, :::2424->2424/tcp, 8000/tcp     prebid-cache
edb5ec978c5f   postgres                      "docker-entrypoint.s…"   8 seconds ago   Up 7 seconds   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp               prebid-server-database
root@ecs3-lcproject:~/prebid/prebid-server# docker-compose up -d
prebid-cache is up-to-date
prebid-server-database is up-to-date
Starting prebid-server ...
Starting prebid-server ... done
root@ecs3-lcproject:~/prebid/prebid-server# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                                                   NAMES
5bdd148cdb13   prebid-server_prebid-server    "./prebid-server"        20 seconds ago   Up 4 seconds    0.0.0.0:8000->8000/tcp, :::8000->8000/tcp               prebid-server
bfca97988d49   dpage/pgadmin4                 "/entrypoint.sh"         20 seconds ago   Up 18 seconds   443/tcp, 0.0.0.0:16543->80/tcp, :::16543->80/tcp        prebid-server-pgadmin
6c613fe5a72d   prebid-server_prebid-cache     "./prebid-cache"         21 seconds ago   Up 19 seconds   0.0.0.0:2424->2424/tcp, :::2424->2424/tcp, 8000/tcp     prebid-cache
edb5ec978c5f   postgres                       "docker-entrypoint.s…"   21 seconds ago   Up 19 seconds   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp               prebid-server-database
root@ecs3-lcproject:~/prebid/prebid-server#
```

aXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Use the command `docker ps` to check if all 4 containers are up and running.

## 9. Validation

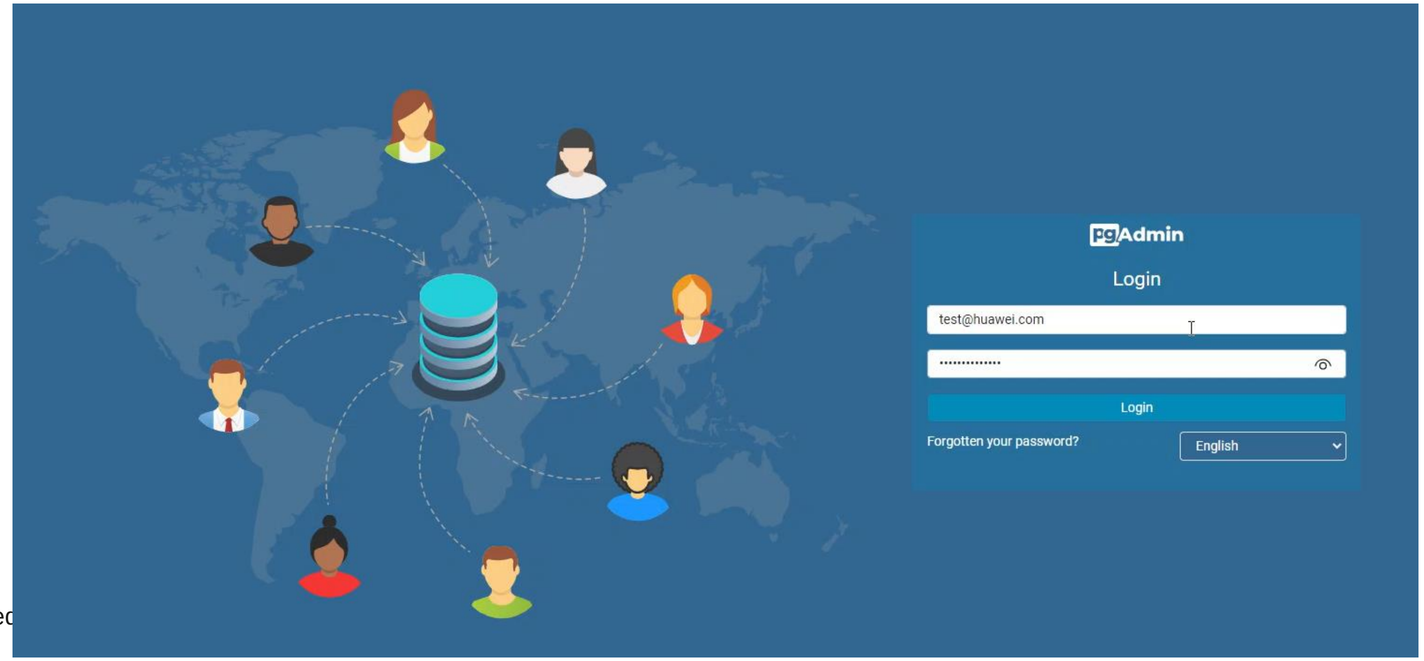


Prebid Server is a server-to-server proxy for Prebid.js users. The host is not responsible for the content or advertising delivered through this proxy. For more information, please contact prebid-server - at - prebid.org.

To check if your Prebid Server is running correctly, open your browser and enter **<server public ip>:8000** to ping the server. If the server is running correctly, you should see the message from above.

# 9. Validation

- Additionally, you can also use PgAdmin to validate if the Postgres Database, the tables and the data have been created (according to the database-seed.sql)
- On your browser, enter <server public ip>:16543 to open PgAdmin
- The default username and password is stated in the `docker-compose.yaml`
  - Username: test@huawei.com
  - Password: @@Huawei2023!!



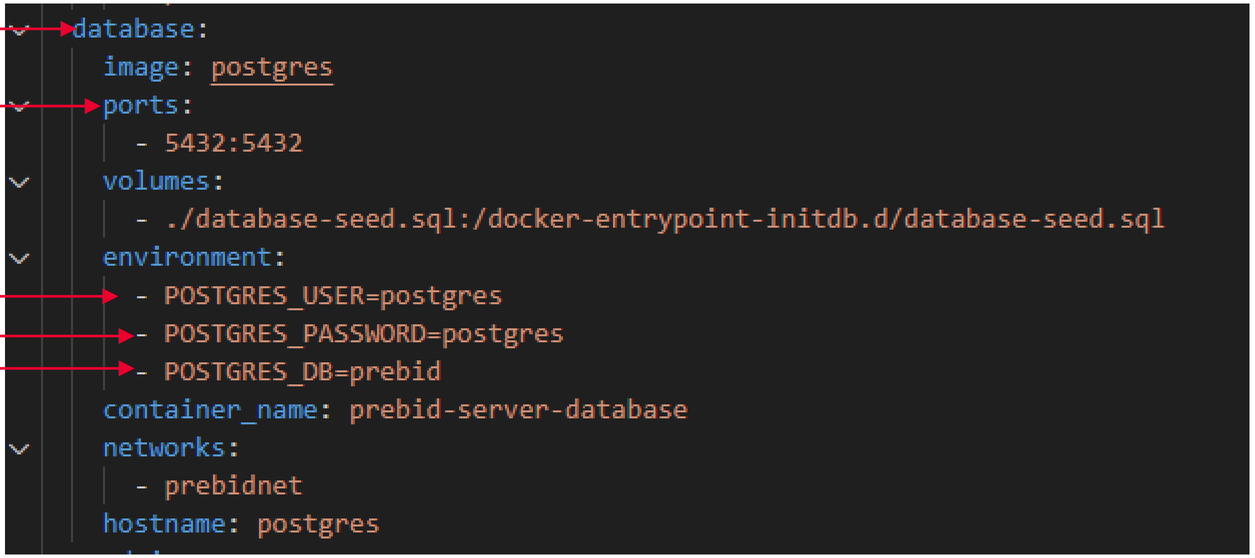
test@huawei.com

# 9. Validation

- Once you are in, **Register a new server** with the following credentials (They are all stated in the `docker-compose.yaml` file)

- General
  - > Name: <Any name you like>

- Connection
  - > Host name / address: database
  - > Port: 5432
  - > Username: postgres
  - > Password: postgres
  - > Maintenance database: prebid

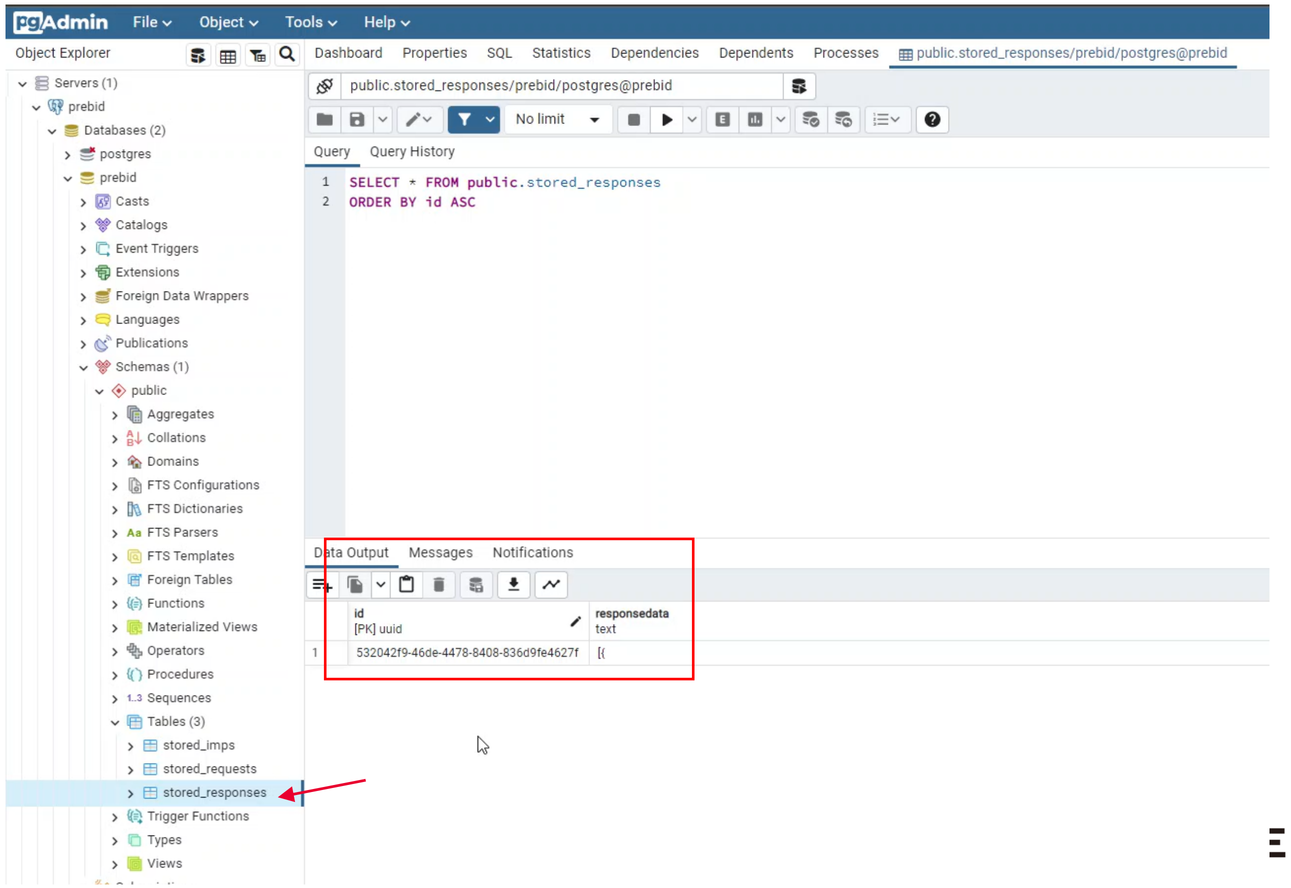


```
  database:
    image: postgres
    ports:
      - 5432:5432
    volumes:
      - ./database-seed.sql:/docker-entrypoint-initdb.d/database-seed.sql
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=prebid
    container_name: prebid-server-database
    networks:
      - prebidnet
    hostname: postgres
```

`docker-compose.yaml`

# 9. Validation

- Go to `Servers – prebid – Databases – prebid – Schemas – Tables` and see if the 3 tables are created
- View the record of each table to see if the values are successfully inserted, according to the `database-seed.sql`



```
SELECT * FROM public.stored_responses
ORDER BY id ASC
```

| | id [PK] uuid | responsedata text |
|---|---|---|
| 1 | 532042f9-46de-4478-8408-836d9fe4627f | [{ |

## 10. Remarks

- Please note that this setup guide is only for demo purposes, a quick kickstart. You have to do more if you wish to set up a real, production Prebid Server.
- Remember to close the unused ports after deploying. (eg: port 22, port 16543)
- Remember to look into the `docker-compose.yaml` file and understand what it does. Change the default credentials to something stronger if you want to deploy this publicly.
- Use the attached Android demo project to test the Prebid server.

## Thank you.

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI