

# Release 6 Installation Document of IEC Type 3: Android cloud native applications on Arm servers in edge

•

[Introduction](#)

[How to use this document](#)

[Deployment Architecture](#)

[Pre-Installation Requirements](#)

- [Software Perequisites](#)
- [Components Version](#)
- [Database Perequisites](#)
- [Installation High-Level Overview](#)
- [Bare Metal Deployment Guide](#)
- [Upstream Deployment Guide](#)

[Install Main Components](#)

- [Run Anbox On Ubuntu18.04](#)
  - [Build Android image for anbox](#)
  - [Required packages installation](#)
  - [Run robox android system](#)
- [Cluster deployment and access by K8S](#)
  - [1 Basic Architecture](#)
  - [2 K8s Cluster construction](#)
  - [3 Check cluster running status](#)
- [Cloud platform monitoring & Analyze](#)
  - [1 Basic Architecture](#)
  - [2 Bootup basic components](#)
  - [3 Analyze data by web browser](#)

[Developer Guide and Troubleshooting](#)

[Uninstall Guide](#)

[Troubleshooting](#)

[Maintenance](#)

[Frequently Asked Questions](#)

[License](#)

[References](#)

[Definitions, acronyms and abbreviations](#)

## Introduction

IEC Type3 mainly focus on Android Application running on edge ARM Cloud architecture with GPU/ vGPU Management. Also, ARM cloud games need to have the basic features of "cloud", such as flexibility ,

availability everywhere. Based on cloud infrastructure optimized for android application, providing ARM application services.

## How to use this document

This document describes the construction, compilation and use of robox Android container environment. Introduce environment configuration, code download, build and compile, and how to use it.

This document is mainly suitable for users who build and compile robox container Android emulation.

## Deployment Architecture

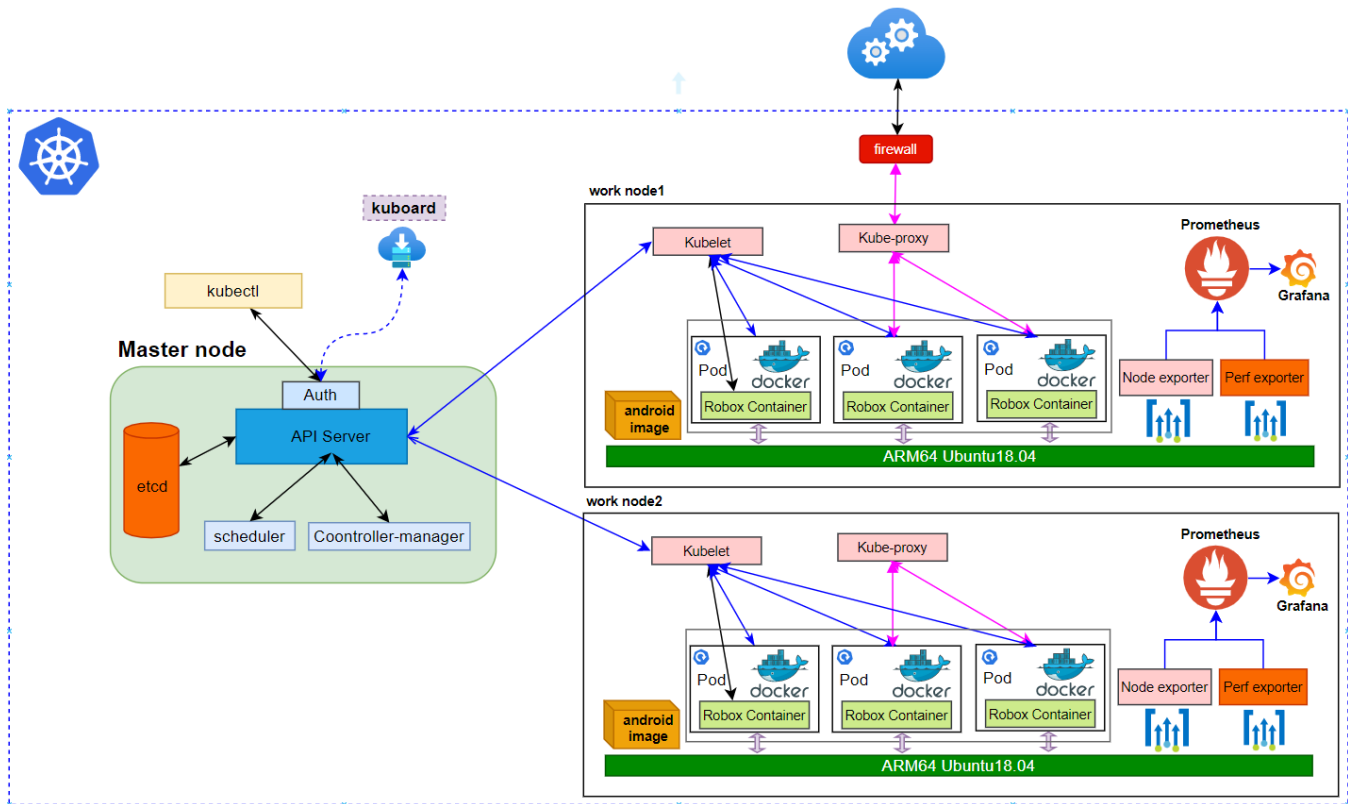


Figure1 Deployment Main Framework

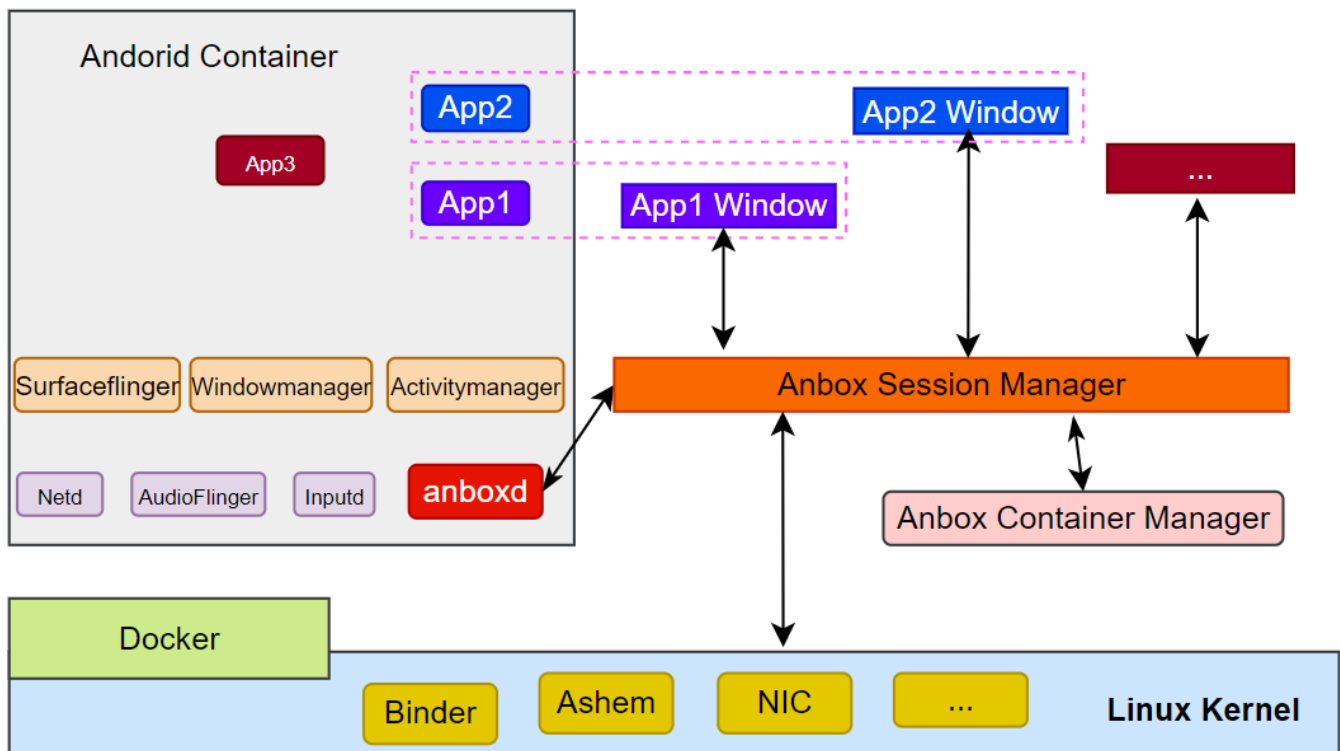


Figure2 Robox Framework

# Pre-Installation Requirements

- Hardware Requirements

2\*arm64 server:

Arch	AARCH64
Processor model	1*Aarch64 processor( A72 inside)
RAM	16*DDR4-2933
Storage	10*2.5 inch SAS/SATA/SSD or 8*2.5 inch NVMe SSD
Network	1 onboard network card, each card supports 4*GE port or 4*10GE port or 4*25GE port
Power Supply	Power 100~240V AC240V DC
Scale	447 mm x 490 mm x 86.1 mm

ARM Server satisfies the Arm Server Ready certified.

- Software Perequisites

item	comments	method
os	ubuntu 18.04.3/6( <a href="#">key</a> )	
robox	robox is an Android container.	<code>git clone <a href="https://github.com/kunpengcompute/robox.git">https://github.com/kunpengcompute/robox.git</a> -b release-phase2.3</code>
robox Compile and run dependent packages	support for robox compilation and operation.	<code>apt-get install build-essential cmake cmake-data debhelper dbus google-mock libboost-dev libboost-filesystem-dev libboost-log-dev libboost-iostreams-dev libboost-program-options-dev libboost-system-dev libboost-test-dev libboost-thread-dev libcap-dev libdbus-1-dev libdbus-cpp-dev libegl1-mesa-dev libgles2-mesa-dev libgl2.0-dev libglm-dev libgtest-dev liblxc1 libproperties-cpp-dev libprotobuf-dev libstdl2-dev libstdl2-image-dev lxc-dev pkg-config protobuf-compiler</code>
docker	needed by K8S/Robox	<code>apt-get install <a href="https://www.docker.io">docker.io</a></code>

- Components Version

Anbox	Run Android applications on any GNU/Linux operating system.	
Grafana	Compose and scale observability with one or all pieces of the stack	8.4.3
Prometheus	Cloud native system performance monitoring	2.34.0
K8s	container orchestration engine for automating deployment, scaling, and management of containerized applications	k8s: v1.23.5;  kube-apiserver:v1.21.11  kube-scheduler:v1.21.11  kube-proxy:v1.21.11  etcd:3.4.13-0  coredns:v1.8.0

- Database Perequisites

schema scripts: N/A

- Other Installation Requirements
  - Jump Host Requirements N/A
  - Network Requirements: The network connection is normal and can communicate with the external network.
  - Bare Metal Node Requirements N/A
  - Execution Requirements (Bare Metal Only) N/A

## Installation High-Level Overview

### Bare Metal Deployment Guide

- Install Bare Metal Jump Host : N/A
- Creating a Node Inventory File: N/A
- Creating the Settings Files: N/A
- Running: N/A

### Upstream Deployment Guide

- Upstream Deployment Key Features N/A
- Special Requirements for Upstream Deployments N/A
- Scenarios and Deploy Settings for Upstream Deployments N/A
- Including Upstream Patches with Deployment N/A
- Running N/A
- Interacting with Containerized Overcloud N/A

Verifying the Setup as defined the Akraino validation feature project plus any additional testing specific to the blue print

## Install Main Components

Since the components and images required by the project are relatively large, and the process of compilation takes time, we store the compiled images on the github repository.

The link is: <https://github.com/ysemi-computing/RoboxWidget.git>

```
git clone https://github.com/ysemi-computing/RoboxWidget.git
```

After this step, The code structure is as follows:

```
RoboxWidget/
  android.img
  components
  grafana-server
  node_exporter
  perf_exporter
  prometheus
  README.md
```

### Run Anbox On Ubuntu18.04

#### Build Android image for anbox

Code cloning and compilation is very time-consuming. If you just want to deploy and experience the Android system by iec, you can skip this section, because the project you just cloned already exists.

Firstly, you should install a ubuntu 18.04 system on Aarch64 processor, and swith to root user, then run the script "[build\\_android\\_image.sh](#)"

After about two hours, the Android image was compiled successfully, and the results are as follows:

```
ls out/target/product/arm64/  
  
android-info.txt  obj  previous_build_config.mk  recovery  symbols  system.img  build_fingerprint.txt  cache.img  data  gen  module-  
info.json  ramdisk.img  root  system  userdata.img
```

## Required packages installation

Before running anbox, we need to install some necessary packages, you can execute the fol script "[arm64\\_env\\_setup.sh](#)"

After this, the image of anbox has been completed, which can be viewed through docker related commandsType the command below:

```
sudo docker images | grep roblox  
  
android                                roblox                                e223a91c4b58  6 days ago  860MB
```

## Run roblox android system

After the above two scripts are executed, the roblox operating environment is ready, and the entire directory structure of roblox can be seen as follows

```
android  
binaryFiles  
cmake  
cmake_uninstall.cmake.in  
COPYING.GPL  
cross-compile-chroot.sh  
data  
docs  
external  
kernel  
patch  
products  
scripts  
src
```

Step1: You can copy the binaryFiles in the roblox code to the some place, then run roblox

start the first continer instance

```
./roblox -v start 1
```

1 is the id, used by session manager and docker container

Step2: Check the docker instance process and session manager process.

```
sudo docker ps | grep instance  
ps -aux | grep session
```

step3: Log in the container and confirm whether the robox instance started successfully.

container name format: **instance + id**

```
docker exec -it instance1 sh
```

step4: get android property sys.boot\_completed

```
getprop | grep sys.boot.completed  
[sys.boot_completed]: [1] //1 which means start successfully
```

## Cluster deployment and access by K8S

It is troublesome to start robox through commands to deploy in a real environment. Here, the well-known K8S system is used to complete the deployment of robox instances. On this basis,

Robox can be much more effectively deployed, run, monitored, and analyzed for multiple host nodes. Here is k8s cluster setup and container orchestration.

### 1 Basic Architecture

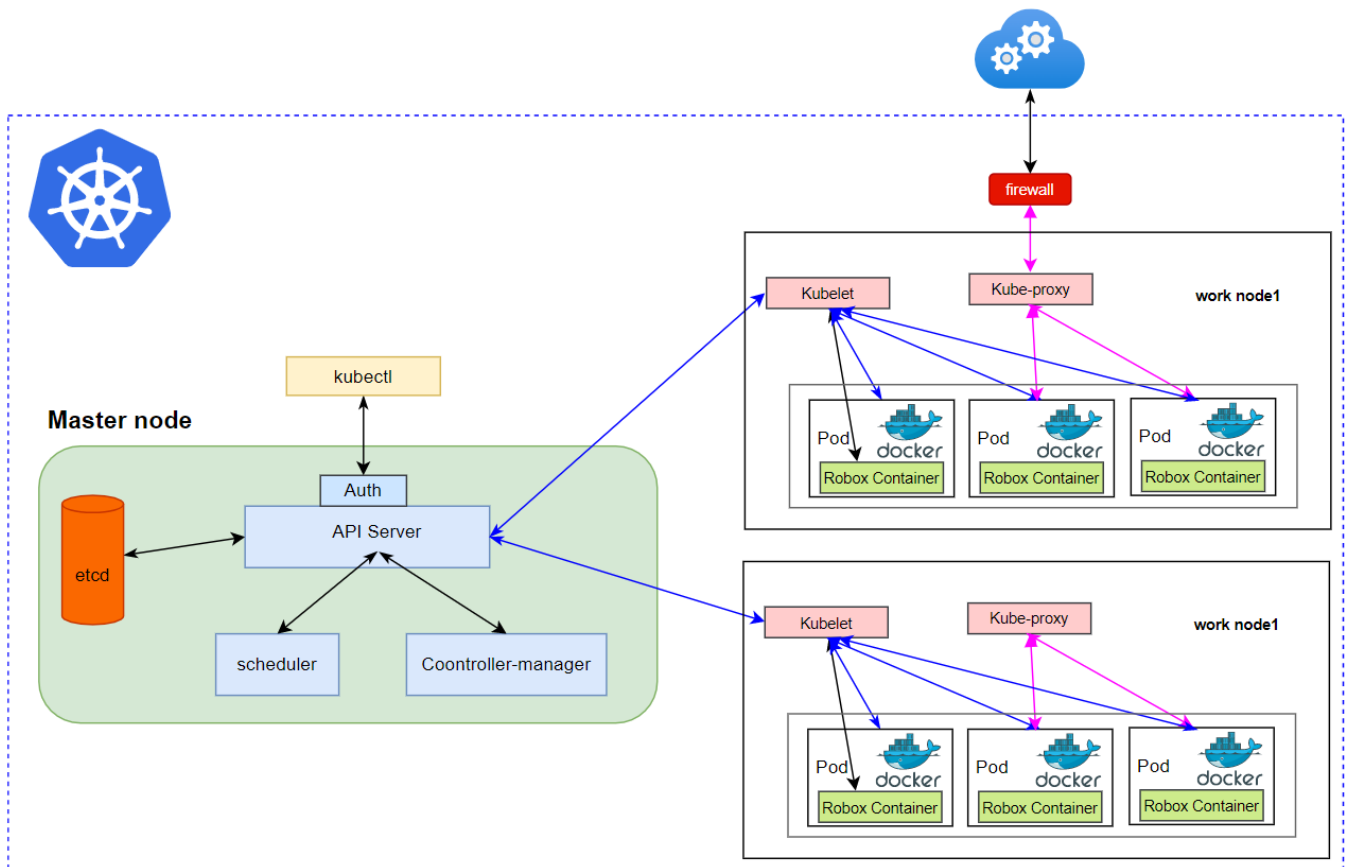


Figure3 View Of Robox In Cluster

## 2 K8s Cluster construction

The k8s cluster construction can be completed through the warehouse code, you can do as follows

```
git clone https://gerrit.akraino.org/r/iec.git
cd iec/deploy/compass && bash deployIEC.sh
```

current configuration:

- Ubuntu Version18.04
- Docker: 20+
- k8s: 1.21.3

Host Network:

- master: 192.168.10.66
- work: 192.168.10.62

## 3 Check cluster running status

kubectI apply -f <https://addons.kuboard.cn/kuboard/kuboard-v3-swr.yaml>

```
kubectI get cs
```

Warning: v1 ComponentStatus is deprecated in v1.19+

NAME	STATUS	MESSAGE	ERROR
------	--------	---------	-------

scheduler	Healthy	ok	
-----------	---------	----	--

controller-manager	Healthy	ok	
--------------------	---------	----	--

etcd-0	Healthy	{"health": "true"}	
--------	---------	--------------------	--

**kubectl get node**

NAME	STATUS	ROLES	AGE	VERSION
------	--------	-------	-----	---------

master	Ready	control-plane,master	22h	v1.21.3
--------	-------	----------------------	-----	---------

work	Ready	<none>	22h	v1.21.5
------	-------	--------	-----	---------

At the same time, we can present the status of the cluster through the k8s visual component. We use kuboard, it is very convenient to install and use. then start kuboard with web browser

`urlmaster-ip-address:30080`

`useradmin`

`passwordKuboard123`

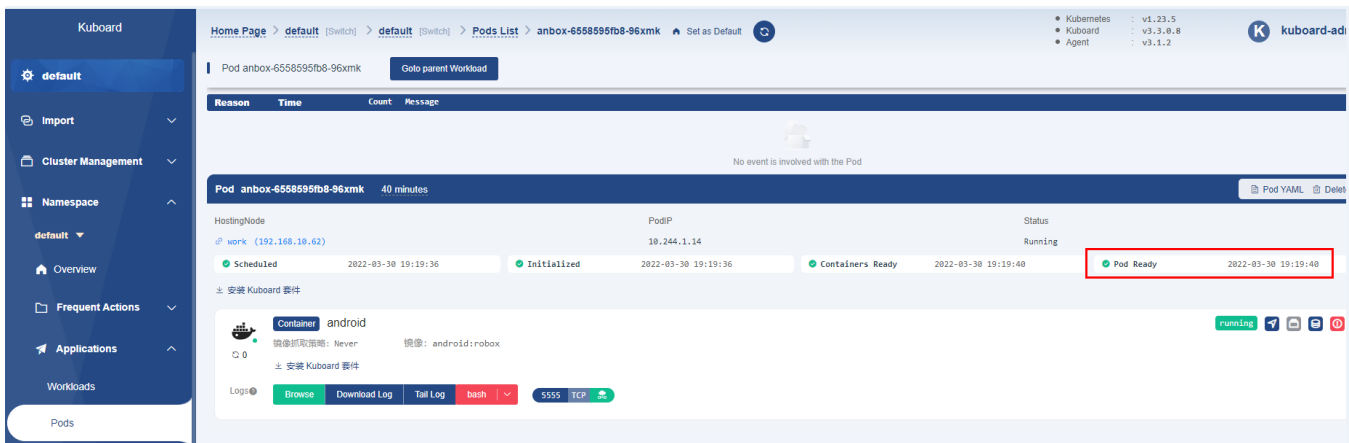
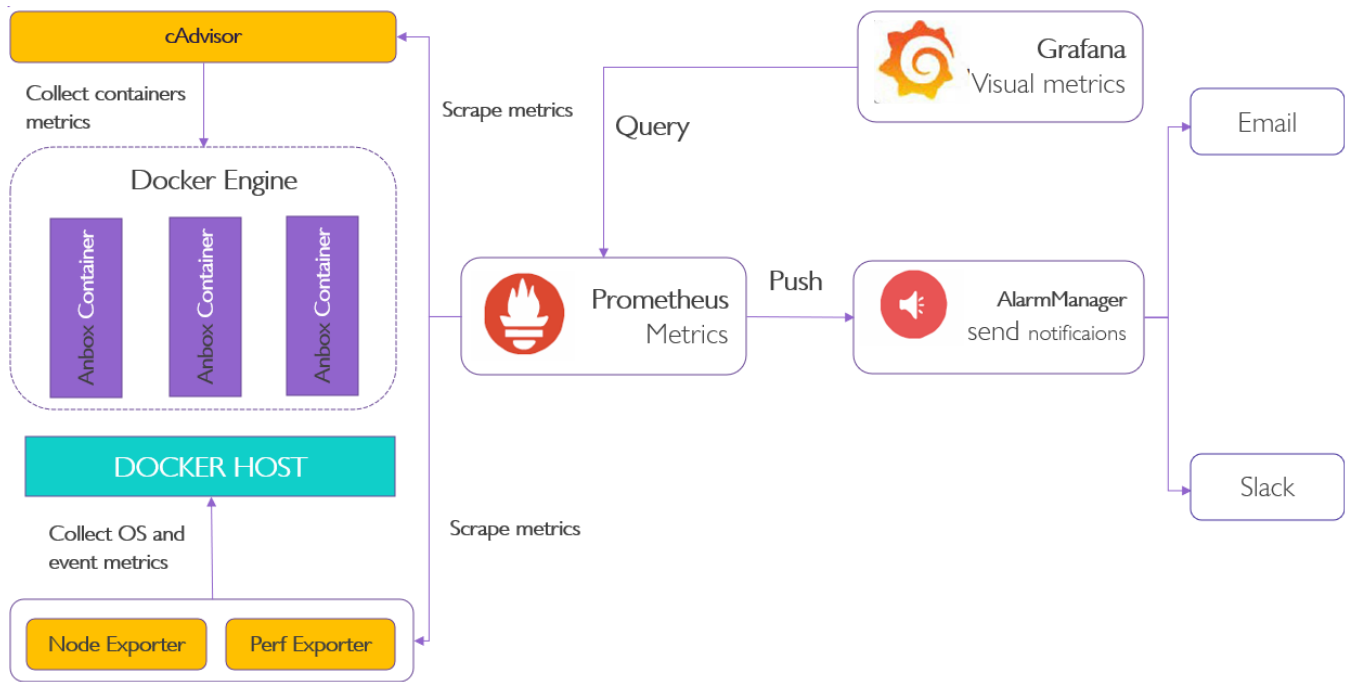


Figure4 View of Robox On Kuboard

## Cloud platform monitoring & Analyze

### 1 Basic Architecture





**Figure5 Prometheus and Grafana Mix**

## 2 Bootup basic components

Prometheus is only used for monitoring data. The real data needs an exporter. Different exporters export different data, These data are finally presented by prometheus.

Only node exporter and perf exporter are used in this project, these two can be directly downloaded and installed with binary files. Prometheus is used for data monitoring,

and then a better visual interface is grafana, the data it needs can just be obtained through Prometheus.

This project only needs to run the script below

[https://github.com/ysemi-computing/RoboxWidget/blob/main/scripts/load\\_components.sh](https://github.com/ysemi-computing/RoboxWidget/blob/main/scripts/load_components.sh)

After the script is executed, the required Prometheus and grafana are downloaded and run in the background

## 3 Analyze data by web browser

Let us explore data that Prometheus has collected about itself. To use Prometheus's built-in expression browser, navigate to <http://localhost:9090/graph> and choose the "Table" view

within the "Graph" tab. Enter the below into the expression console and then click "Execute":

```
prometheus_target_interval_length_seconds
```



Figure6 First Startup View Of Prometheus

In addition, we can log in to grafana through a web browser and see the effect as shown below

url: <http://localhost:3000>  
 user: admin  
 password: admin

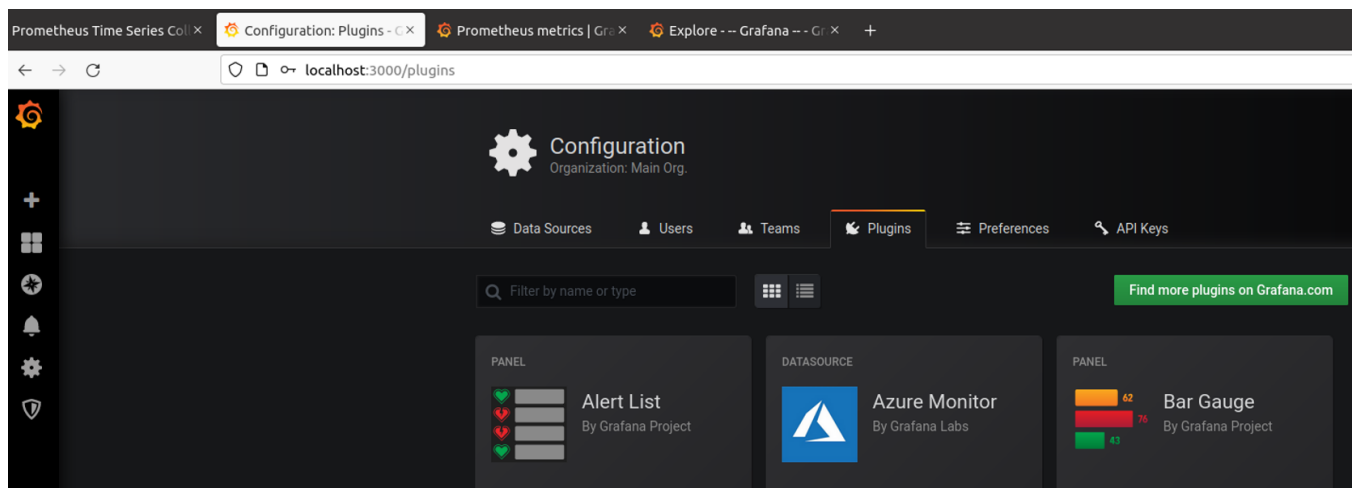


Figure7 First Startup View Of Grafana

## Developer Guide and Troubleshooting

- Utilization of Images

N/A

- Post-deployment Configuration

N/A

- Debugging Failures

N/A

- Reporting a Bug

N/A

## Uninstall Guide

N/A

## Troubleshooting

When the server restarts, the binder\_linux module needs to be loaded and installed as follows

```
sudo modprobe ashmem_linux  
  
sudo rmmod binder_linux || true; sudo modprobe binder_linux num_devices=254  
  
sudo chmod 777 /dev/ashmem /dev/binder*
```

If there is an external graphics card, you need to manually disable the built-in graphics card after each server restart. for example:

```
lspci | grep -in VGA  
  
6:0007:41:00.0 VGA compatible controller: Huawei Technologies Co., Ltd. Hi1710 [iBMC Intelligent Management system chip w  
/VGA support] (rev 01)  
  
echo 1 > /sys/bus/pci/devices/0007:41:00.0/remove
```

## Maintenance

- Blue Print Package Maintenance
  - Software maintenance N/A
  - Hardware maintenanceN/A
- Blue Print Deployment Maintenance (N/A)

## Frequently Asked Questions

N/A

## License

N/A

## References

N/A

## Definitions, acronyms and abbreviations

N/A

When porting an Android app to Anbox Cloud (usually in the form of an APK), there are a few issues that might cause your app to not function properly: