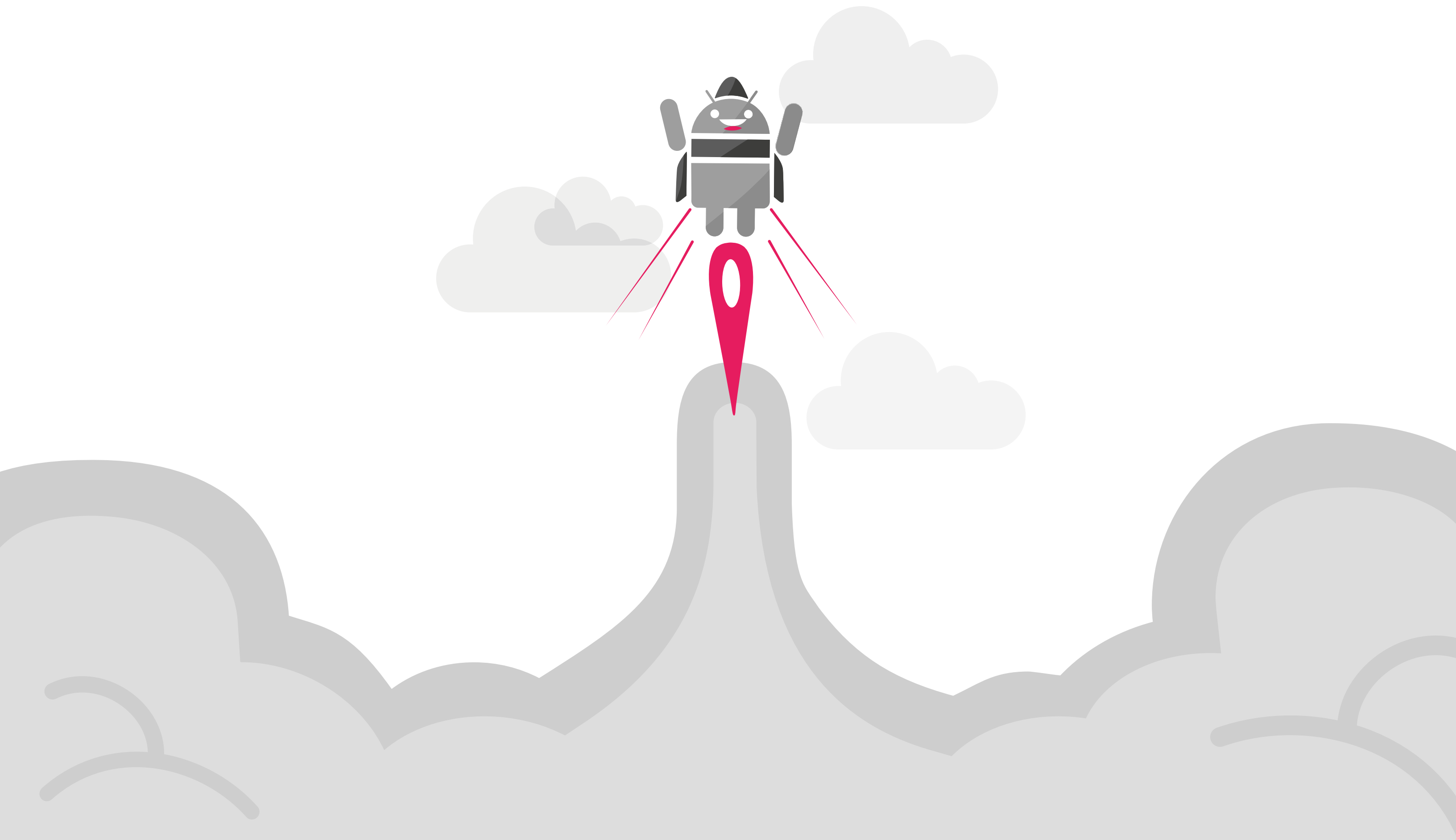
GENYMOTION cloud

# PaaS User Guide

Version 5.0



# Table of contents

# Overview

Genymotion Cloud "Platform as a Service" (PaaS) offers an Android environment featuring the power of Genymotion on the cloud providers Amazon Web Services and Google Cloud Platform. Genymotion Cloud PaaS is a "pay-as-you-use" solution that allows you to play with Android virtual devices within a web browser and size your environment according to your exact needs. You can for example integrate test automation into your continuous integration workflow but you can also use those on-demand virtual devices to address any kinds of mobile needs you may have, be they marketing, sales, support, monitoring-related and so on.

This user guide will help you get access to Genymotion Cloud virtual devices in order to make full use of its resources.

In this guide, the following instructional icons are used:

***Notes, tips or additional information.***

***Situations that could cause performance issues or data losses.***

# Requirements

This chapter lists the prerequisites for an optimal use of Genymotion Cloud on Amazon Web Services or Google Cloud Platform.

## Web browser

To use Genymotion Cloud and view your instances in a web browser, we recommend using Google Chrome. Genymotion Cloud is also known to work fine on Opera and Mozilla Firefox but may not provide an optimal level of performance.

## Communication

To enable communication with your cloud provider, you must allow ports listed in this table.

| Provider's firewall rules | | Client side | |
|---|---|---|---|
| Inbound SSH | TCP port 22 | Outbound SSH | TCP port 22 |
| Inbound HTTP | TCP port 80 | Outbound HTTP | TCP port 80 |
| Inbound HTTPS/WSS | TCP port 443 | Outbound HTTPS/WSS | TCP port 443 |
| Inbound WebRTC (WebRTC is used to create a peer-to-peer connection. This peer connection uses a dynamic port). | TCP and UDP ports 51000-51100 | Outbound WebRTC | TCP and UDP ports 51000-51100 |

⚠ ***We do not recommend opening the ADB port. The ADB connection is neither secured nor authenticated. Instead we recommend creating an SSH tunnel, as explained in section Enabling ADB. If you do decide to open the ADB connection, allow this port: Inbound ADB on TCP port 5555.***

If you are behind a very restrictive web proxy, then you can install a TURN server to use it as a relay. Please refer to *Accessing virtual devices behind a web proxy* for more information on how to install and configure a TURN server.

## Amazon Web Services

Ports are configured in a security group that contains all the firewall rules. In case you need to edit those rules:

1. Go to **Network & Security > Security groups**.
2. Select the security group associated with your virtual device.
3. From tabs **Inbound** or **Outbound**, click **Edit**.
4. Edit rules according to your needs.

## Google Cloud Platform

Ports are configured in network tags that are specific to a firewall rule. Your instance is bound to various network tags. To edit a firewall rule:

1. Go to **VPC network > Firewall rules**.
2. Edit an existing rule by clicking its name and **Edit**.

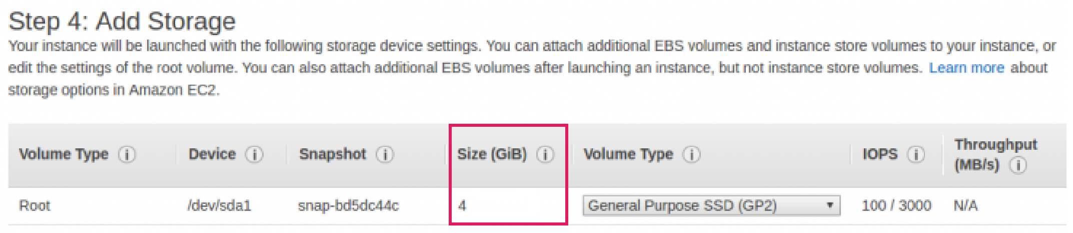
# Getting started

In this section, you will be guided through the very first actions to quickly have a virtual device running in your web browser. Virtual devices are based on machine images.

## Starting Genymotion Cloud on AWS

To use Genymotion Cloud on AWS:

1. Go to *Amazon Web Services*.
2. From *AWS Marketplace*, search for Genymotion.
3. Select the version of Genymotion Cloud corresponding to the Android version of virtual devices you wish to run. You can run Genymotion virtual devices based on a Lollipop AMI (Android 5.1), a Marshmallow AMI (Android 6.0) or a Nougat AMI (Android 7.0).
4. Click **Continue**.
5. In **1-Click Launch** section, select a region.
6. From **EC2 Instance Type** section, select an instance corresponding to your needs. You can refer to the table below to know which instance to select depending on your use case.
7. Select your security group to define the access rules of your Genymotion instance.
8. Accept software terms and launch in 1 click.
   By default, the created instance has 4GB. This is enough to run Google apps and a 1GB application. If you need more space, go through the **ManualLaunch** section instead of the **1-Click Launch** section. During step **Add Storage**, adjust the size of the storage disk by changing default values in the **Size** field.



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

| Volume Type | Device | Snapshot | Size (GiB) | Volume Type | IOPS | Throughput (MB/s) |
|---|---|---|---|---|---|---|
| Root | /dev/sda1 | snap-bd5dc44c | 4 | General Purpose SSD (GP2) | 100 / 3000 | N/A |

   ***Only the /data partition will be resized accordingly.***

   Once the initial setup is done, the instance automatically starts. It can take up to two minutes before the instance can be accessed through the web browser.
9. From the EC2 management console, in **Instances**, select the instance you want to access.
10. From the **Description** tab, copy the hostname or public IP and paste it to your web browser.
    A connection warning pops up. This happens because the certificate is autosigned. Proceed and authenticate.
    By default, the username is "genymotion" and the password is the ID of the instance you can

retrieve from tab **Description**.
To avoid this security warning, you must install a trusted certificate, as explained in section *Installing a certificate*.

Below is a table with the recommended instances for the most frequent use cases.

| Description | Recommended instance |
|---|---|
| Basic Android environment<br>Limited graphical interface<br>(automation, monitoring, testing) | t2.small |
| Standard Android environment<br>Standard graphical interface<br>(excluding video streaming and gaming) | t2.medium |
| Professional Android environment<br>Good visual performance<br>(Video streaming, 2D gaming) | m4.large |
| Professional Android environment<br>High-end graphic rendering<br>(Better performance for video streaming and 2D gaming) | m4.xlarge |
| Turbocharged Android environment<br>(Video streaming, 3D gaming) | m4.2xlarge |
| Turbocharged Android environment, hardware-accelerated GPU display<br>(Better performance for video streaming, 3D gaming) | g2.2xlarge |

A wider range of compatible EC2 instances is available. You can view all of them on *Genymotion Cloud on AWS marketplace*. For more details on instance types, please refer to *Amazon EC2 instance types* web page.

## Starting Genymotion Cloud on GCP

To use Genymotion Cloud on GCP:

1. Go to *Google Cloud Platform*.
2. Select the version of Genymotion Cloud corresponding to the Android version of virtual devices you wish to run. You can run Genymotion virtual devices based on a Lollipop image (Android 5.1), a Marshmallow image (Android 6.0) or a Nougat image (Android 7.0).
3. Click **Launch on Compute Engine**.

4. Configure your virtual device:
    1. In **Deployment name**, enter a name for the virtual device.
    2. In **Zone**, select your geographical area. The zone determines what computing resources are available and where your data is stored and used.
    3. In **Machine type**, select the specifications of your machine from the dropdown list or by clicking **Customize**. Machine types determine the specifications of your machines, such as the amount of memory, virtual cores, and persistent disk limits of your instance. For more information about instances, please refer to *GCP machine types* web page.
    4. From the **Boot Disk** section, select the boot disk type and size. Storage space is much less expensive for a standard persistent disk. An SSD persistent disk is better for random IOPS or streaming throughput with low latency. Persistent disk performance is tied to the size of the persistent disk volume. You are charged for the actual amount of provisioned disk space.
    5. From the **Networking** section, select the network to be used by the instance and a subnetwork. This assigns the instance an IPv4 address from the subnetwork's range. Instances in different subnetworks can communicate with each other using their internal IPs as long as they belong to the same network.
    6. In **Firewall**, select the type of network traffic you want to allow. By default all incoming traffic from outside a network is blocked.
    7. In **External IP**, select a type of IP address associated with this instance. Selecting "None" will result in the instance having no external internet access.
    8. In **Source IP ranges for HTTPS/HTTP/Webrtc/Adb traffic**, define the IP address ranges on which traffic is allowed. Use CIDR notation when entering ranges.

        ***We do not recommend opening the ADB port. The ADB connection is neither secured nor authenticated. Instead we recommend creating an SSH tunnel, as explained in section Enabling ADB. If you do decide to open the ADB connection, allow this port: Inbound ADB on TCP port 5555.***

    9. Enable or disable **IP forwarding**. Forwarding allows the instance to help route packets.
5. Click **Deploy**.
6. Copy the temporary **Web UI password**.
7. Click **Log into Genymotion**.
8. Enter username "genymotion" and paste the password.

To stop your virtual device, close the tab where it is running and click **Delete** on the GCP interface.

## Installing a certificate

To avoid the security warning which displays when accessing an instance, you must install a trusted certificate. Follow one of the methods below whether you already own a certificate or not.

### You have no certificate

If you do not have any certificates, we developed a service that generates a certificate from Let's encrypt servers and installs it directly in the instance.

#### On AWS

1. From **Network & Security/Elastic IPs** , create an Elastic IP address by clicking **Allocate new address**.
   This ensures that the IP address will not change.
2. Set up your domain name according to the Elastic IP address.
3. Make sure that the security groups configured on your instance allow 80 and 443 port connection from all IP addresses.
4. From the **Elastic IPs** tab, associate your instance with the Elastic IP address:
   1. Click **Actions** > **Associate address**.
   2. In the **Instance** field, select your instance and click **Associate**.
5. From the **Instances** tab, modify the user data of the instance:
   1. Click **Actions** > **Instance Settings** > **View/Change User Data**.
   2. Add:
      `{ "user_dns" : "your.domain.name" }`, or
      `{ "user_dns" : "your.domain.name1,your.domain.name2" }` for multiple domain names.
6. Start the instance.

You can now access your server using its domain name from your web browser.

To clear previous settings or generate a new certificate, you must run the following commands using ADB:

```
adb shell am startservice -a genymotionacme.clear -n
com.genymobile.genymotionacme/.AcmeService
```

```
adb shell am startservice -a genymotionacme.generate -n
com.genymobile.genymotionacme/.AcmeService
```

### On GCP

1. From **Compute Engine > VM instances** , select an instance.
2. Click **Edit**.
3. Click **Network interface > External IP > Create IP address**.
4. Enter a name for the new static IP address.
   The name must start with a lowercase letter followed by up to 63 lowercase letters, numbers, or hyphens, and cannot end with a hyphen.
5. (Optional) Enter a description.
6. Click **Reserve** and **Done**.
7. Modify the user data of the instance: in **Custom metadata**, click **Add item**.
8. In **Key**, enter `"user_dns"`.
9. In **Value**, enter `"your.domain.name"` or `"your.domain.name1,your.domain.name2"` for multiple domain names.
10. Click **Save**.

You can now access your server using its domain name from your web browser.

To clear previous settings or generate a new certificate, you must run the following commands using ADB:

```
adb shell am startservice -a genymotionacme.clear -n
com.genymobile.genymotionacme/.AcmeService
```

```
adb shell am startservice -a genymotionacme.generate -n
com.genymobile.genymotionacme/.AcmeService
```

## You already have a certificate

To install your own SSL certificate, please refer to this repository:
*github.com/Genymobile/genymotion-cloud-ssl-tool*

## Accessing a virtual device from the command line

You can access your virtual device via SSH. Your public key is retrieved from your provider's metadata at boot time. Use your corresponding private key to get SSH access using:

```
ssh -i key.pem shell@instance_ip
```

If you want to avoid adding `-i key.pem` for every SSH command, make sure you have an SSH agent configured and add the key using:

```
ssh-add key.pem
```

## Accessing virtual devices behind a web proxy

With Genymotion Cloud, users give or get access to virtual devices through the Internet. Thus, they must be able to connect to them from their corporate network and configure them to make them reachable.

For every asset and webpage, the Genymotion virtual devices are accessible through the HTTPS port 443. However, WebRTC connections, needed to display the virtual devices in the web browser, use an alternative protocol relying on the non-standard port range 51003 to 51100 (UDP or TCP). This might cause problems to users who are behind a web proxy to access the Internet. Their proxy must allow CONNECT requests on that port range (eg. Squid). If users cannot modify their corporate network configurations to allow those outgoing connections, then an external TURN server must be configured.

TURN is a server used as a relay for the media part of WebRTC communications. It is used to relay UDP or TCP when one of the peers cannot be reached or cannot contact the other peer because of port restriction.

This section explains how to install and configure the TURN server. The server can be installed on a remote machine that needs to be accessible publicly.

### Installing a TURN server on Linux

We recommend installing a CoTURN server which is available in the latest stable Ubuntu Linux distribution (16.04+).

To install a CoTURN server:

1. Add the Universe repository using `sudo apt-add-repository universe`
2. Install the server using `sudo apt-get install coturn`
3. Make sure the server restarts on boot by uncommenting `TURNSERVER_ENABLED=1 in /etc/default/coturn file`

4. Modify the configuration file `/etc/turnserver.conf` to have the CoTURN server listen on port 443:
    1. Make sure the server listens to port 443 by uncommenting `listening-port=3478` and changing to `listening-port=443`.
    2. Add a user and password for your Genymotion virtual device by uncommenting `user=your-username:your-password`.
5. Restart your computer.
6. Make sure the CoTURN server started correctly and that it is listening on port 443 by checking the most recent log file (`/var/log/turn_xxxx_2017-02-01.log` where "xxxx" changes at each server boot).
7. Make sure no other servers are running and listening on port 443, or CoTURN won't be able to use it.

## Configuring the Genymotion virtual device to use the TURN server

First, you need to authorize the TURN server public IP in AWS or GCP firewall rules so that it can access your virtual device.

Next, to make sure the web browser uses the TURN server, you must add its configuration to the web page that is served by the Genymotion device internal web server. To do so:

1. Connect to the device using SSH. For more information, you can refer to section *Accessing a Genymotion virtual device from the command line*.
2. From the `/data/www/` directory of the `index.htm` page, add a TURN element in the options element:

```
var options = {
    template: "god_default",
    god: true,
    token: (typeof token !== 'undefined') ? token :
'genymotion',
    turn: {
         urls: [
              'turn:TURNServerPublicIP:443?transport=tcp',
              'turn:TURNServerPublicIP:443?transport=udp'
         ],
         username:'username1',
         credential:'password1'
    }
};
```

where:

- `TURNServerPublicIP` is your TURN server public IP;
- `username1` is the username set in `turnserver.conf` file;
- `password1` is the password set in `turnserver.conf` file.

***Don't forget to add a comma at the end of the previous line in the options structure.***

Your web browser now uses the TURN server on port 443 when you cannot reach Genymotion Cloud directly for WebRTC connections (on port 51003 -> 51100).

## Setting up ADB

ADB is the communication protocol specific to Android that must be enabled to allow communication between a computer and Android devices.

### Enabling ADB

To establish the communication between your computer and Android virtual devices, follow the steps below:

1. Log in with SSH using `ssh -i key.pem shell@instance_ip`
2. Enable ADB using `setprop persist.sys.usb.config adb`

   ***ADB is now accessible using the standard 5555 TCP port. Since the ADB connection is neither secured nor authenticated, we do not recommend opening the ADB port 5555 in AWS Security Group, nor allowing ADB traffic in your GCP configuration. Instead, we recommend creating an SSH tunnel.***

3. Create an SSH tunnel for this connection using:
   `ssh -i key.pem -NL 5555:localhost:5555 shell@instance_ip`
   Do not close this shell.
4. Open another shell to run other commands.
5. (Optional) Connect your virtual device using:
   `adb connect locahost:5555`
   This command is optional as the first virtual device might have been automatically connected to ADB.
6. To connect other virtual devices to ADB, run commands below. Make sure you increment the port number for every new virtual device (5556, 5557, 5558, etc.):

```
ssh -i key.pem -NL 5556:localhost:5555 shell@instance_ip2
adb connect localhost:5556
```

### Disabling ADB

To stop the communication between your computer and Android virtual devices, follow the steps below:

1. Log in with SSH using `ssh -i key.pem shell@instance_ip`
2. Run the shell command `setprop persist.sys.usb.config none`

## Installing Open GApps

***Genymobile Inc. assumes no liability whatsoever resulting from the download, install and use of Google Play Services within your virtual devices. You are solely responsible for the use and assume all liability related thereto. Moreover, Genymobile Inc. disclaims any warranties of any kind, either express or implied, including, without limitation, implied warranties of merchantability, or fitness for a particular purpose regarding the compatibility of the Open GApps packages with any version of Genymotion. In no event shall Genymobile Inc. or its affiliates, or their respective officers, directors, employees, or agents be liable with respect to your download or use of the Google Play Services and you release Genymobile Inc. from any liability related thereto. You agree to defend, indemnify and hold harmless Genymobile Inc. for any claims or costs related to your use or download of the Google Play Services.***

The application you are developing or testing may require an interaction with Google Play Services (e.g.: in-app purchasing, advertising, etc.).

If you really need them, you can use the packages provided by Open GApps. This section details how to install Open GApps from their website or using the command line.

### From the Open GApps website

You can install Open GApps directly from their website.

1. Visit *opengapps.org*.
2. Select platform **x86_64**.
3. Select the Android version corresponding to your virtual device.
4. Select variant **nano** or **pico**.
5. Download the selected Open GApps package.
6. Drag and drop the installer in the new Genymotion virtual device.
7. Follow the installation instructions.

### From the command line

You can install Open GApps using SSH or ADB. Both methods are detailed below.

#### SSH method

To install Open GApps using SSH:

1. Copy the archive using `scp -i key.pem archive.zip shell@instance_ip:/sdcard/Download/archive.zip`
2. Log in with SSH `ssh -i key.pem shell@instance_ip`.
3. Run `su` to switch to root user. For more information about root access, please refer to section *Using root access*.
4. Flash the archive using `/system/bin/flash-archive.sh /sdcard/Download/archive.zip`
5. Reboot your instance.

#### ADB method

To install Open GApps using ADB:

1. Copy the archive using `adb push archive.zip /sdcard/Download/archive.zip`
2. Flash the archive using `adb shell /system/bin/flash-archive.sh /sdcard/Download/archive.zip`
3. Reboot your instance using `adb reboot`

## Deploying an application

To deploy an application to a virtual device, use either of the following methods:

- Drag and drop the application `APK` file into the virtual device window, or
- Run the following command: `adb install <application name>.apk`.

## Emulating sensors and features

To simulate various behaviors of your application according to specific use cases, Genymotion provides easy-to-use widgets which emulate the following sensors and features: Sound volume, Rotate screen, Fullscreen, File upload, Camera, Battery, GPS, Identifers, Network and Baseband, Phone, Resolution, Disk I/O.

### Sound volume

You can control the volume of the sound emitted by your virtual device by clicking or .

### Rotate screen

You can rotate the screen of your virtual device by clicking .

### Full-screen

You can turn the full-screen mode on or off by clicking .

### File upload

You can upload files from your computer to the virtual device. To do so, click and browse for the file you wish to upload.

### Camera

The Camera widget allows you to send a video stream from a virtual device to the Android system. With this widget, you can test an Android application that uses an Android built-in camera.

The video stream comes from a real physical webcam connected or integrated into your computer.

To use the Camera widget, click .

***Make sure your camera is allowed to access the virtual device.***

***From , select a resolution with a ratio that matches your webcam orientation (for example 1024x800 for a portrait image or 800x1024 for a landscape orientation source).***

## Battery

The Battery widget allows you to test how your application reacts with different battery charge levels and states of charge.

To use the Battery widget:

1. Click .
2. Modify the charge level using the slider or enter a value in the **Charge level** field.
3. Modify the state of charge by checking the **State of charge** box:
   - Check to activate the **Charging** state.
     This simulates that the power supply is plugged in and the battery is charging.
   - Uncheck to activate the **Discharging** state.
     This simulates that the power supply is unplugged and the battery is discharging.

## GPS

The GPS widget allows real-time activation and modifications of a position, accuracy and bearing.

To use the GPS widget:

1. Click .
2. Set the latitude value you wish to simulate using the **Latitude** field.
   The latitude value must range from -90° to 90°.
3. Set the longitude value you wish to simulate using the **Longitude** field.
   The longitude value must range from -180° to 180°.
4. Set the altitude value you wish to simulate using the **Altitude** field.
   The altitude value must range from -10000m to 10000m.
5. Set an accuracy value using the slider or by entering a value in the **Accuracy** field.
   The accuracy value must range from 0m to 200m.
6. Set a bearing value using the compass or by entering a value in the **Bearing** field.
   The bearing value must range from 0° to 359.99°.

   ***Many applications do not rely on the GPS orientation, but use the device accelerometer or gyroscope to determine the bearing of the device, which are not yet supported.***

You can also define a location using the **Map** button. Once a location selected, click **Capture** to retrieve its coordinates.

## Capture

The Capture widget allows you to take a screenshot or screencast of virtual devices. This way, you can broadcast images or videos of your applications.

To take a screenshot of your virtual device, click and .

To take a screencast of your virtual device:

1. Click .
2. Click to start recording.
3. Record your sequence.
4. Click to stop recording.

Screenshots and screencasts are stored in your **Downloads** folder.

## Identifiers

The Identifiers widget shows **Device ID** and **Android ID** numbers. You can view and edit these values at any time, without having to reboot your virtual device.

To use the Identifiers widget, click ID.

- **Android ID**
  An Android ID is a 64-bit number randomly generated when the user first sets up the device. It remains the same for the whole lifetime of the user's device. Android 4.2.2 and greater versions support multiple user accounts, each one having a unique Android ID.
  When clicking **GENERATE**, a random Android identifier is generated. Valid Android ID numbers are 16-hexadecimal digits long.

  ***You are not allowed to set an empty Android ID.***

- **Device ID / IMEI / MEID**
  By default, a new virtual device is deployed with the default device ID number 00000000000000 0. When clicking **GENERATE**, a random identifier is generated. As IMEI or MEID numbers are used as device ID, Genymotion generates numbers compliant with the GSM 02.16 standard and the 3GPP2 specification (14 digits or hexadecimal digits + a checksum digit).
  Valid characters for setting device ID/IMEI/MEID are: lower-case and upper-case letters **[a-z, A-Z]**, digits **[0-9]**, dots **[.]**, dashes **[-]** and underscores **[_]**.

## Network and Baseband

The Network and Baseband widget allows you to test how your application reacts with different network quality and performance types. You can also test different mobile network operators with different SIM operators.

To test network quality and performance types:

1. Click .
2. Select a network profile from the **Network Speed** drop-down list.

   Network profiles and their corresponding values are listed in the table below.

| | Upload speed | Download speed | Upload delay | Download delay | Upload packet loss | Download packet loss | DNS delay |
|---|---|---|---|---|---|---|---|
| No data | 0Kb/s | 0Kb/s | 0ms | 0ms | 100% | 100% | 0ms |
| GPRS | 40Kb/s | 40Kb/s | 500ms | 500ms | 0.01% | 0.01% | 1000ms |
| Edge | 200Kb/s | 240Kb/s | 400ms | 400ms | 0.01% | 0.01% | 800ms |
| 3G | 1.5Mb/s | 7.2Mb/s | 100ms | 100ms | 0.01% | 0.01% | 200ms |
| 4G | 5.5Mb/s | 17.9Mb/s | 50ms | 50ms | 0.01% | 0.01% | 100ms |
| 4G (high DNS delay) | 5.5Mb/s | 17.9Mb/s | 50ms | 50ms | 0.01% | 0.01% | 3000ms |
| 4G (high packet losses) | 5.5Mb/s | 17.9Mb/s | 50ms | 50ms | 10% | 10% | 100ms |
| Wifi | 33.0Mb/s | 40.0Mb/s | 0ms | 0ms | 0% | 0% | 0ms |

To use the Baseband feature, you can define SIM operator information:

1. Click .
2. In **MCC/MNC**, enter a Mobile Country Code for the SIM operator.
3. In **MSIN**, enter a mobile subscription identification number used by the SIM operator.
4. In **Name**, enter a name for the SIM operator.
5. In **Phone Number**, enter the phone number corresponding to the SIM.

You can then specify the mobile network information:

1. In **MCC/MNC**, enter the Mobile Country Code or Mobile Network Code you wish to test.
2. Enter the name of the operator.

## Phone

The Phone widget allows you to test applications relying on telephony features and observe their behavior when receiving a call or a text message.

To use the Phone widget, click .

To simulate an incoming call:

1. Enter an incoming phone number.
2. Click **Call**.

To simulate an incoming message:

1. Enter an incoming phone number.
2. Enter a text message.
3. Click **Send message**.
   The text message is displayed in the virtual device via a notification and can also be read in the **Messaging** application.

## Resolution

To change the resolution of your virtual device, follow one of the methods detailed below.

### From a web browser

1. From the right-hand side toolbar, click .
2. Select a resolution from the dropdown list.
   The virtual device reboots.
3. Refresh the web page.
   This can take several seconds.

### From the command line

Android geometry (size and density) can be changed with the following properties. You must be in the virtual device shell to run the following commands.

```
setprop persist.graph_mode WidthxHeight-Depth (e.g. 720x1280-32)
setprop persist.dpi dpi (e.g. 240)
```

***Only resolutions available in the side toolbar (see previous section) are guaranteed to work, others may cause image corruptions.***

### Disk I/O

The Disk I/O widget allows you to emulate devices with slow internal storage. It can be very handy if your app requires reading large amount of data from the disk such as gallery apps that load locally stored images or game apps that load large files.

***As the speed limit provided by this feature comes on top of the one already defined in the AWS or GCP instance, make sure the profile you wish to apply does not exceed the disk performance already provided in the instance. If so, the widget will not be able to emulate the expected disk performance. For more information, refer to the AWS documentation or GCP documentation.***

To simulate high or poor disk performance:

1. Click .
2. From **Profile**, select a type of device:
   - **High-end device:** Read speed limit: 200MiB per second;
   - **Mid-range device:** Read speed limit: 100MiB per second;
   - **Low-end device:** Read speed limit: 50MiB per second;
   - **Custom device:** enter the read speed limit you wish to emulate in MiB.
3. Click **Update**.

***When switching from a device profile to another, the disk cache is automatically cleared. You can force clearing the cache by clicking Clear Cache.***

## Setting up the Kiosk mode

If you want a user to interact with your application and nothing else, you can configure the virtual device so that it turns into a kiosk mode, making it a single-purpose tool.

### Locking an application

1. Connect to your virtual device via SSH, as explained in section *Accessing a Genymotion virtual device from the command line*.
2. Start the application you want to lock in kiosk mode.
3. Once the application is started, run `cmd activity kiosk start`.

The status bar is hidden and cannot be scrolled down. The user cannot use the **Back** or **Home** buttons.

## Unlocking an application

To unlock an application and disable the kiosk mode, run `cmd activity kiosk stop`. You can now exit the application. If you reboot your device, your application will still be locked.

***If within your application you start another application, this might not work depending on how this application is started. For more information, please refer to the LockTask mode from Android DeviceOwner APIs.***

# Disabling the toolbar

Once in the virtual device shell, you can enable or disable the toolbar:

1. To change the shell ownership from ordinary to root user, run `su`. For more information about root access, please refer to section *Using root access*.
2. To disable the toolbar, run `/sbin/busybox sed -i -e 's/default/simple/g' /data/www/index.html`
3. To enable the toolbar, run `/sbin/busybox sed -i -e 's/simple/default/g' /data/www/index.html`

# Changing credentials

1. Connect to the virtual device using ADB.
2. Change the username using `adb shell "setprop persist.webrtcd.username \$(echo -n "new_username" | sha1sum | cut -d \" \" -f1)"`
3. Change the password using `adb shell "setprop persist.webrtcd.password \$(echo -n "new_password" | sha1sum | cut -d \" \" -f1)"`
4. Enter the new credentials.

# Disabling authentication

1. Log in with SSH using `ssh -i key.pem shell@instance_ip`
2. Change the property using `setprop persist.webrtcd.authent off`

   ***For security reasons, make sure that the virtual device is only accessible from authorized IP addresses.***

3. Close your instance using `exit`
4. Refresh the web page running the virtual device.

If you want to change or disable the credentials, please refer to this *tutorial*.

# Using root access

Rooting allows Android mobile operating system users to reach privileged control (known as root access) over various Android subsystems. As Android uses the Linux kernel, rooting an Android device gives similar access to administrative (superuser) permissions as on Linux or any other Unix-like operating systems such as FreeBSD or macOS.

***All Genymotion Cloud devices are rooted and cannot be unrooted.***

## From an application

Superuser is already installed on Genymotion Cloud devices. When an application requests root access, it prompts a pop-up asking whether root access should be authorized or denied. The default policy can be changed using the Superuser application.

## From the command line

You can become root using SSH or ADB. Both methods are detailed below.

### SSH method

1. Log in with SSH as "shell" user: `ssh -i yourprivatekey.pem shell@instance_ip`.
2. Run `su` to switch to root user.

### ADB method

When logging in using ADB, you are already connected as root.

# Getting the list of open-source licenses

Genymotion Cloud uses some open-source components. Licenses of those components are listed in the virtual device. To access this list, go to **Settings > About phone > Legal Information > Open source licenses**.

For components under GPL and LGPL, we provide an archive containing all their source code in a TAR file: `/system/src/gpl-source.tgz`

# Genymotion Java API

Sometimes, in your Android tests (formerly known as instrumented tests), you need to test what happens in your application when sensors return specific values. As Android real devices cannot fake sensor values, you need to modify your source code to mock sensors and create proxy objects. This adds unwanted noise into your project source code only useful for testing, making your code less readable and harder to maintain.

All sensors being already mocked inside Genymotion, the Genymotion Java API allows you to directly manipulate sensor values from your Android tests.

This explains how to install and use Genymotion Java API.

## Installing Genymotion Java API

To install Genymotion Java API, follow the steps corresponding to your build engine:

- Maven
- Gradle
- Other build systems

### Maven

1. Add the Genymotion Java API repository to the `pom.xml` file:

```
<repository>
  <id>genymotion</id>
  <name>Genymotion repo</name>
  <url>http://api.genymotion.com/repositories/releases/</url>
</repository>
```

2. Add the dependency:

```
<dependency>
  <groupId>com.genymotion.api</groupId>
  <artifactId>genymotion-api</artifactId>
  <version>1.0.4</version>
</dependency>
```

### Gradle

1. Add the Genymotion Java API repository to the `build.gradle` file:

```
repositories {
  maven{
    url "http://api.genymotion.com/repositories/releases/"
  }
}
```

2. Add the dependency:

```
androidTestCompile 'com.genymotion.api:genymotion-api:1.0.4'
```

### Other build systems

Add *genymotion-api-1.0.4.jar* file to the `libs` folder.

## Using Genymotion Java API

To use the Genymotion Java API in your Android test project, follow the steps below:

1. Inside your Android test project, get a reference to the Genymotion object using:

```
GenymotionManager genymotion = Genymotion.getGenymotionManager
(getInstrumentation().getContext())
```

2. Access sensors from the GenymotionManager.
   For example, to set the battery charge level, use the command below:

```
genymotion.getBattery().setLevel(10);
```

This call freezes the application (10 seconds maximum) until the change is really effective.

You can find all available methods in *Genymotion Java API Javadoc*.

### Tips

Most of the time, your application listens to sensor changes using a listener, then updates the application interface. Genymotion Java API only freezes until sensor values are retrieved.

To make sure your interface has had enough time to perform the update before testing it, you can add the following code snippet:

```
getInstrumentation().waitForIdleSync();
```

To make sure your Android test is only run inside Genymotion and not on a real device, you can exit the test by running the following command:

```
if (!GenymotionManager.isGenymotionDevice()) {
  return; //don't execute this test
}
```

## Examples

An application called Binocle showcases Genymotion Java API use. In this application, you can find activities for which the behavior depends on sensor values.

Below are some Android test examples built with Genymotion Java API to manipulate sensor values and check activity behaviors.

### Battery

An application must display a warning message if the device is not plugged to a power source and has less than 10% of charge left.

- Click this link to see the fragment showing it: *BatterySampleFragment.java*
- Click this link to see an Android test of the behavior: *TestBattery.java*

### GPS

An application must display a message if the device is localized near a specific place.

- Click this link to see the fragment showing it: *GpsSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestGps.java*

### Radio

An application must display a message if the device is a Nexus 4, as recognized by its IMEI number.

- Click this link to see the fragment showing it: *RadioSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestRadio.java*

### ID

An application must encrypt data using ANDROID_ID to avoid the backed up data to be moved to another Android device.

- Click this link to see the fragment showing it: *IdSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestId.java*

### Phone

An application must display a green check mark when the virtual device receives a text message containing the text "666".

- Click this link to see the fragment showing it: *PhoneSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestPhone.java*

For more information about the Binocle application, you can refer to *Binocle source code*.