



REMOTE CONTROL & TELEMETRY

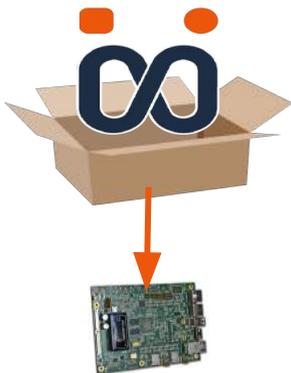
Technical description

TomorrowData

How does it work?

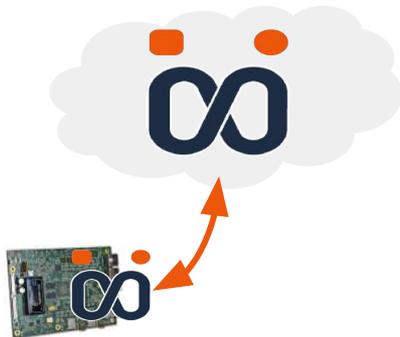


1



Instantly install and launch iottlyAgent on devices

2



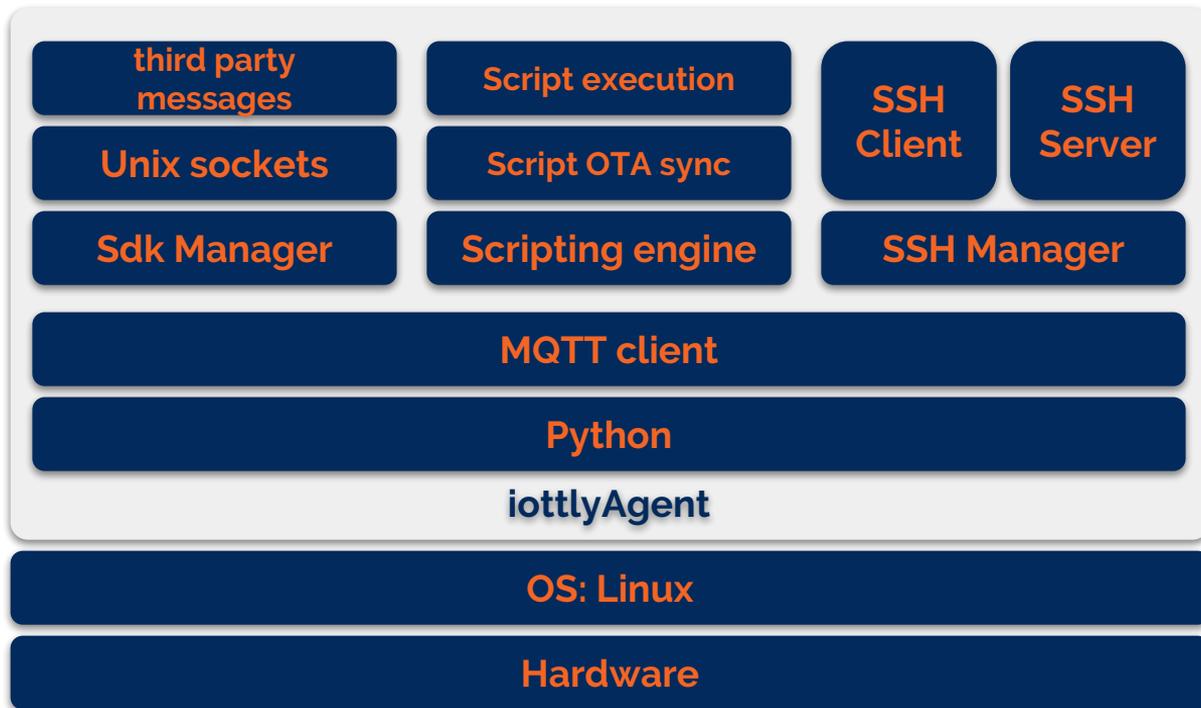
Devices **autonomously connect** to iottlyCloud (MQTT/TLS)

3

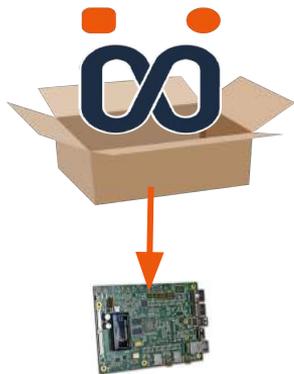


Technicians operate on the devices **from everywhere** via the web app

iottlyAgent architecture



iottlyAgent: Technical Specs



Easy device setup

iottlyAgent coexists without interfering with third-party firmware

Footprint: 30Mb flash, 6Mb RAM



Available for Linux
ARMv5 ARMv6+ AMD64 i386

Based on static python tailored for embedded devices

Rapid portability on any platform



Outbound
Connections Only

Outbound: iottlyCloud 8883 (MQTT/TLS)

Outbound: iottlyCloud 2200 (SSH)

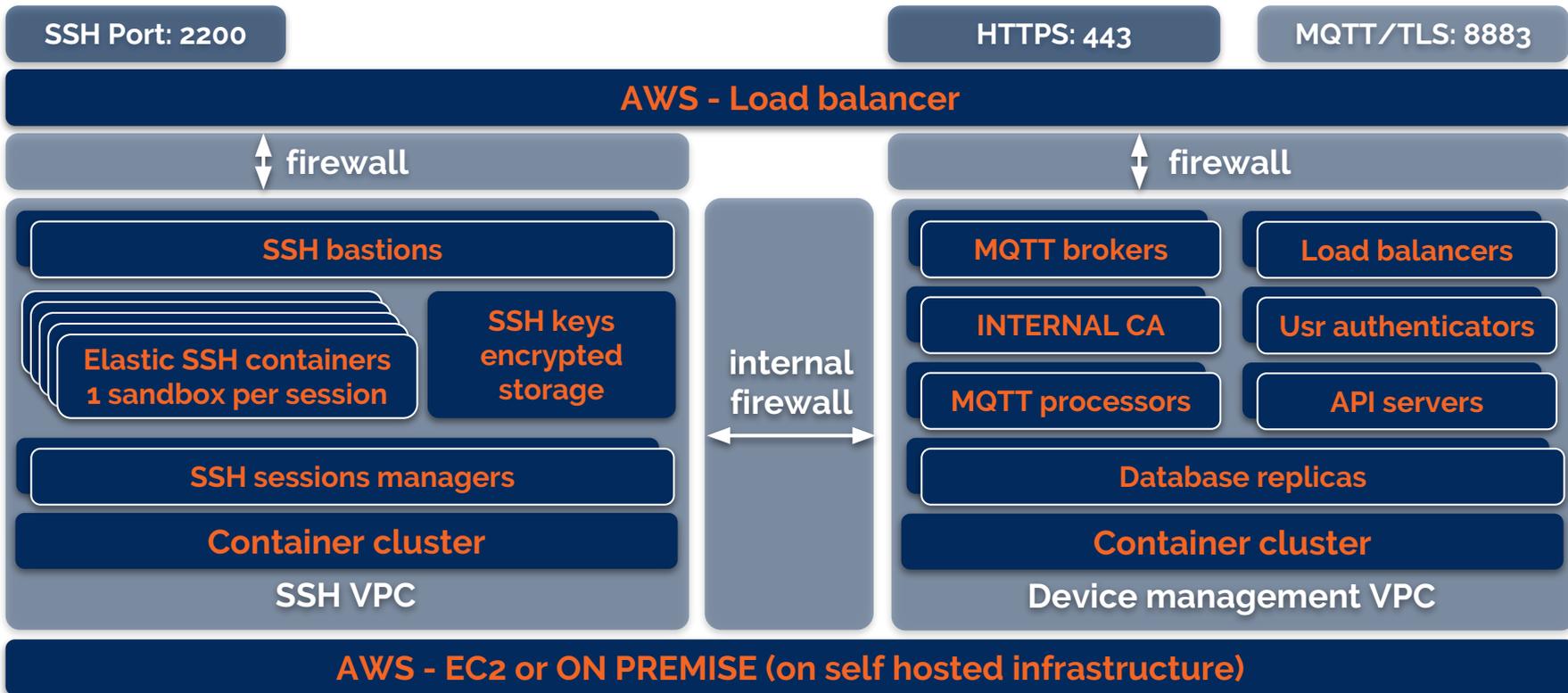


Connectivity:
LAN, WiFi,
LTE, GPRS

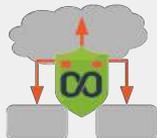
Lightweight MQTT protocol

Secure communications: TLS 2048

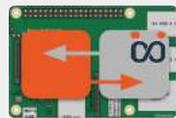
iottlyCloud architecture



Core features



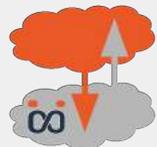
off the shelf **security**



on-device **SDK**



on-device python **scripting engine**



Telemetry middleware via cloud APIs

Management features



everywhere **web SSH**



real-time **control panel**



 Flash over the air

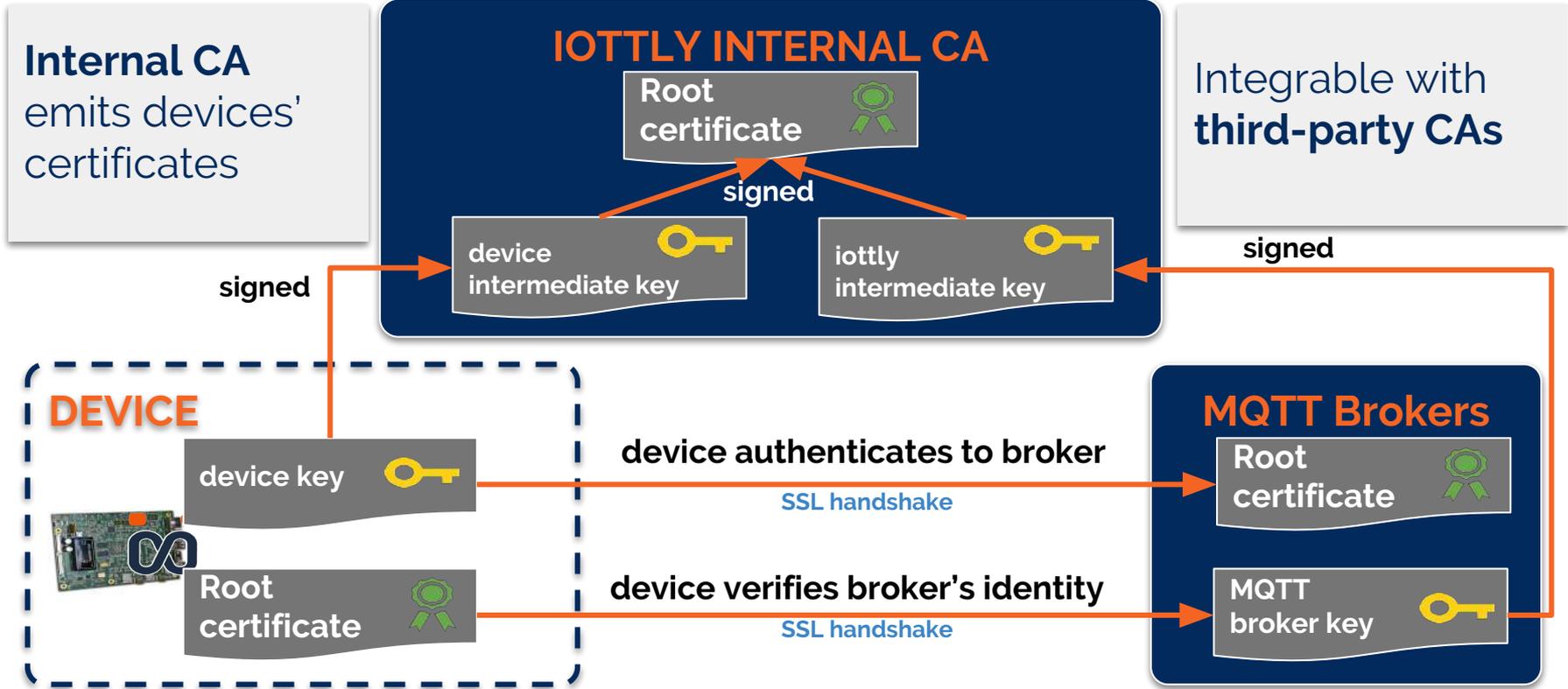
third party **firmware upgrades**



fleet management with **groups**



Off the shelf security - MQTT X.509





Off the shelf security - pairing



Device identification and registration phase.

Unique and unambiguous **device identification**.

Integrable with third-party bulk flashing systems.



Pair new device
HTTPS / authenticated



Ephemeral unique device token
HTTPS / authenticated



Ephemeral unique device token

SSH or USB



Request credentials

HTTPS / token authenticated



MQTT certificates

SSH keys

HTTPS / token authenticated





on-device agent architecture - SDK





on-device scripting engine - web



TEST BANCO SOFTWARE Python 3.4 2 Devices OPEN

DEVICES-COP DEPLOYMENT-GROUPS MESSAGES CODING-FIRMWARE CONSOLE TERMINAL

CODING FIRMWARE

```

dev
# Save
83 HEADERS.update({'Content-MD5': content_md5})
84 if content_type:
85     HEADERS.update({'Content-Type': content_type})
86
87
88     return {
89         'method': verb,
90         'url': resource,
91         'body': content,
92         'headers': HEADERS
93     }
94
95 # REBOOT
96 def _reboot(option = None):
97     command = ['reboot']
98     if not option is None:
99         command.append(option)
100
101     return check_output(command, shell=False)
102
103 #KILLALL
104 def _killall(signal, processname):
105     try:
106         check_output(['killall', "%s" % (signal), processname], shell=False)
107         return "success"
108     except Exception as e:
109         raise e
110
111 #PS TO DICT
112 ps_headers = {'pid': 1, 'cmd': 2, 'cmdargs': 3}
113 def _ps_to_dict():
114
115     ps = check_output(['ps'], shell=False, decode=True).strip().split("\n")
116     return {'processes': ps}
117
118 # GET SACT VERSION
119 def _get_version():
120     fw1_version = check_output(['myfw/myfw1', '-v'], shell=False, decode=True)
121     fw2_version = check_output(['myfw/myfw2', '-v'], shell=False, decode=True)
122     return {'version': [fw1_version, fw2_version]}
123
124
125 #IFCONFIG
126 def _remove_empty():
127     return [e for e in l if e]
128
129 def _ifconfig():
130
131     ifcfg = _remove_empty(check_output(['ifconfig'], shell=False, decode=True))
132
133     with open('/etc/network/network.conf', 'r') as f:
134         netconf = _remove_empty(list(map(lambda l: l.rstrip(), list(f))))
135         netconf0 = netconf[0].split("-")
136         netconf1 = netconf[1].split("-")
137         netconf2 = netconf[2].split("-")
138         netconf3 = netconf[3].split("-")
139
140

```

CONSOLE

SCHEDA LAN - UDPI3 - BANCO 10.115

Commands

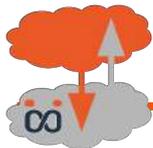
- upload_file
- download_file
- killall
- reboot
- start_tcpdump
- stop_tcpdump
- ps
- ifconfig

get the list running process

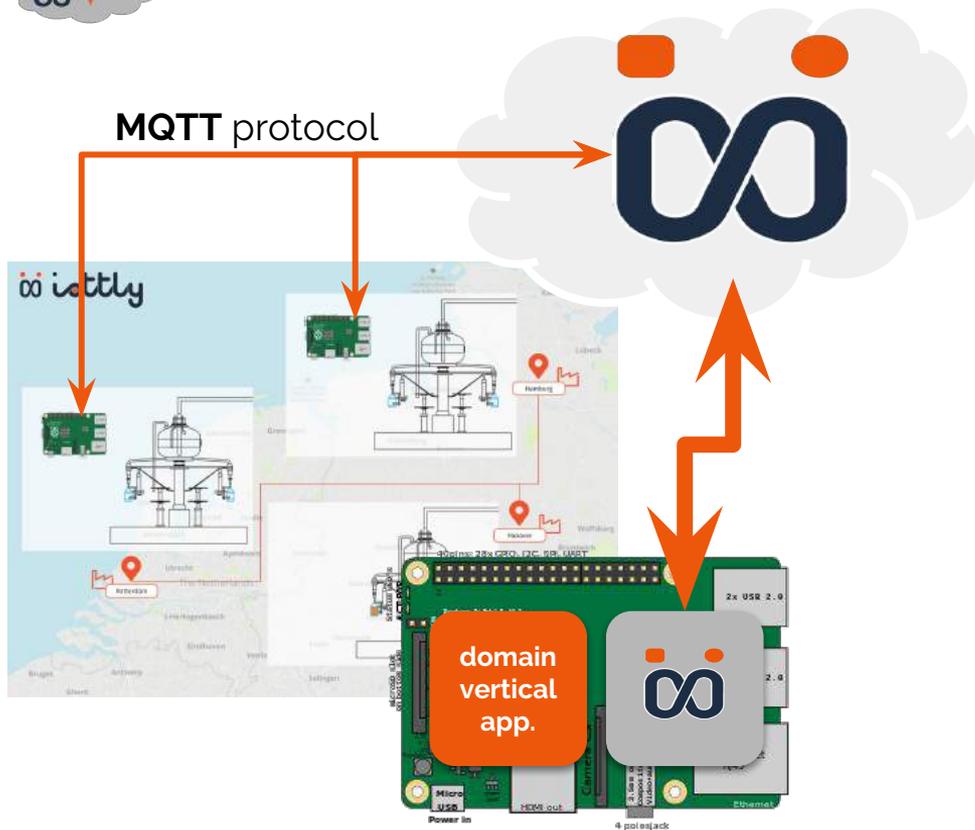
```

{"ps": [{"pid": 1, "cmd": "sh", "cmdargs": ""}, {"pid": 2, "cmd": "python", "cmdargs": ""}, {"pid": 3, "cmd": "python", "cmdargs": ""}, {"pid": 4, "cmd": "python", "cmdargs": ""}, {"pid": 5, "cmd": "python", "cmdargs": ""}, {"pid": 6, "cmd": "python", "cmdargs": ""}, {"pid": 7, "cmd": "python", "cmdargs": ""}, {"pid": 8, "cmd": "python", "cmdargs": ""}, {"pid": 9, "cmd": "python", "cmdargs": ""}, {"pid": 10, "cmd": "python", "cmdargs": ""}, {"pid": 11, "cmd": "python", "cmdargs": ""}, {"pid": 12, "cmd": "python", "cmdargs": ""}, {"pid": 13, "cmd": "python", "cmdargs": ""}, {"pid": 14, "cmd": "python", "cmdargs": ""}, {"pid": 15, "cmd": "python", "cmdargs": ""}, {"pid": 16, "cmd": "python", "cmdargs": ""}, {"pid": 17, "cmd": "python", "cmdargs": ""}, {"pid": 18, "cmd": "python", "cmdargs": ""}, {"pid": 19, "cmd": "python", "cmdargs": ""}, {"pid": 20, "cmd": "python", "cmdargs": ""}, {"pid": 21, "cmd": "python", "cmdargs": ""}, {"pid": 22, "cmd": "python", "cmdargs": ""}, {"pid": 23, "cmd": "python", "cmdargs": ""}, {"pid": 24, "cmd": "python", "cmdargs": ""}, {"pid": 25, "cmd": "python", "cmdargs": ""}, {"pid": 26, "cmd": "python", "cmdargs": ""}, {"pid": 27, "cmd": "python", "cmdargs": ""}, {"pid": 28, "cmd": "python", "cmdargs": ""}, {"pid": 29, "cmd": "python", "cmdargs": ""}, {"pid": 30, "cmd": "python", "cmdargs": ""}, {"pid": 31, "cmd": "python", "cmdargs": ""}, {"pid": 32, "cmd": "python", "cmdargs": ""}, {"pid": 33, "cmd": "python", "cmdargs": ""}, {"pid": 34, "cmd": "python", "cmdargs": ""}, {"pid": 35, "cmd": "python", "cmdargs": ""}, {"pid": 36, "cmd": "python", "cmdargs": ""}, {"pid": 37, "cmd": "python", "cmdargs": ""}, {"pid": 38, "cmd": "python", "cmdargs": ""}, {"pid": 39, "cmd": "python", "cmdargs": ""}, {"pid": 40, "cmd": "python", "cmdargs": ""}, {"pid": 41, "cmd": "python", "cmdargs": ""}, {"pid": 42, "cmd": "python", "cmdargs": ""}, {"pid": 43, "cmd": "python", "cmdargs": ""}, {"pid": 44, "cmd": "python", "cmdargs": ""}, {"pid": 45, "cmd": "python", "cmdargs": ""}, {"pid": 46, "cmd": "python", "cmdargs": ""}, {"pid": 47, "cmd": "python", "cmdargs": ""}, {"pid": 48, "cmd": "python", "cmdargs": ""}, {"pid": 49, "cmd": "python", "cmdargs": ""}, {"pid": 50, "cmd": "python", "cmdargs": ""}, {"pid": 51, "cmd": "python", "cmdargs": ""}, {"pid": 52, "cmd": "python", "cmdargs": ""}, {"pid": 53, "cmd": "python", "cmdargs": ""}, {"pid": 54, "cmd": "python", "cmdargs": ""}, {"pid": 55, "cmd": "python", "cmdargs": ""}, {"pid": 56, "cmd": "python", "cmdargs": ""}, {"pid": 57, "cmd": "python", "cmdargs": ""}, {"pid": 58, "cmd": "python", "cmdargs": ""}, {"pid": 59, "cmd": "python", "cmdargs": ""}, {"pid": 60, "cmd": "python", "cmdargs": ""}, {"pid": 61, "cmd": "python", "cmdargs": ""}, {"pid": 62, "cmd": "python", "cmdargs": ""}, {"pid": 63, "cmd": "python", "cmdargs": ""}, {"pid": 64, "cmd": "python", "cmdargs": ""}, {"pid": 65, "cmd": "python", "cmdargs": ""}, {"pid": 66, "cmd": "python", "cmdargs": ""}, {"pid": 67, "cmd": "python", "cmdargs": ""}, {"pid": 68, "cmd": "python", "cmdargs": ""}, {"pid": 69, "cmd": "python", "cmdargs": ""}, {"pid": 70, "cmd": "python", "cmdargs": ""}, {"pid": 71, "cmd": "python", "cmdargs": ""}, {"pid": 72, "cmd": "python", "cmdargs": ""}, {"pid": 73, "cmd": "python", "cmdargs": ""}, {"pid": 74, "cmd": "python", "cmdargs": ""}, {"pid": 75, "cmd": "python", "cmdargs": ""}, {"pid": 76, "cmd": "python", "cmdargs": ""}, {"pid": 77, "cmd": "python", "cmdargs": ""}, {"pid": 78, "cmd": "python", "cmdargs": ""}, {"pid": 79, "cmd": "python", "cmdargs": ""}, {"pid": 80, "cmd": "python", "cmdargs": ""}, {"pid": 81, "cmd": "python", "cmdargs": ""}, {"pid": 82, "cmd": "python", "cmdargs": ""}, {"pid": 83, "cmd": "python", "cmdargs": ""}, {"pid": 84, "cmd": "python", "cmdargs": ""}, {"pid": 85, "cmd": "python", "cmdargs": ""}, {"pid": 86, "cmd": "python", "cmdargs": ""}, {"pid": 87, "cmd": "python", "cmdargs": ""}, {"pid": 88, "cmd": "python", "cmdargs": ""}, {"pid": 89, "cmd": "python", "cmdargs": ""}, {"pid": 90, "cmd": "python", "cmdargs": ""}, {"pid": 91, "cmd": "python", "cmdargs": ""}, {"pid": 92, "cmd": "python", "cmdargs": ""}, {"pid": 93, "cmd": "python", "cmdargs": ""}, {"pid": 94, "cmd": "python", "cmdargs": ""}, {"pid": 95, "cmd": "python", "cmdargs": ""}, {"pid": 96, "cmd": "python", "cmdargs": ""}, {"pid": 97, "cmd": "python", "cmdargs": ""}, {"pid": 98, "cmd": "python", "cmdargs": ""}, {"pid": 99, "cmd": "python", "cmdargs": ""}, {"pid": 100, "cmd": "python", "cmdargs": ""}]}

```



Telemetry middleware via cloud APIs



The embedded devices are connected to iottly over the internet

both cloud or on-premise

secure connections

- iottly applies enterprise grade security standards (TLS 2048 / X.509)

iottly management software running on the devices

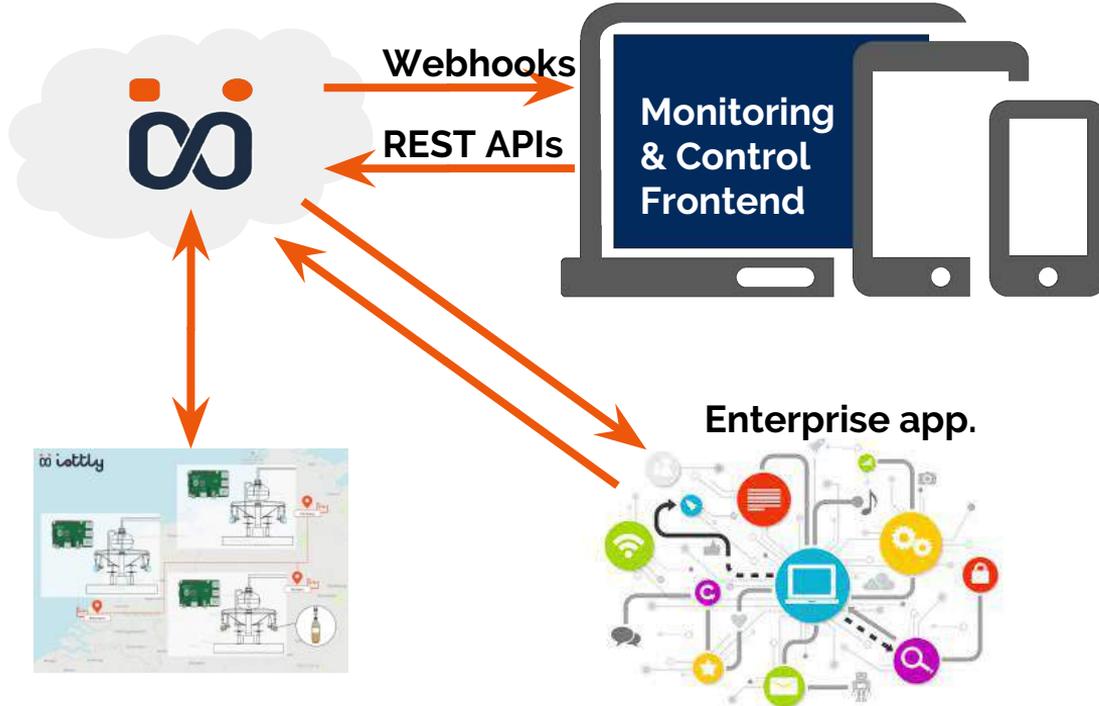
- iottly provides a standard software to be installed on any kind of device

edge: domain vertical running on the devices

- test, debug, upgrade, maintain the custom application on the device



Telemetry middleware via cloud



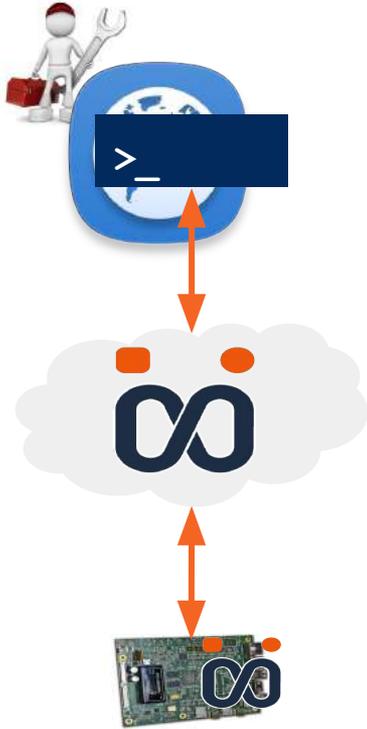
Third-party applications communicate with the devices through iottly

Real-time notifications

- iottly pushes messages from the devices to third-party applications, through webhooks

Trusted active controls

- third-party applications can securely interact with the devices through iottly rest apis



 **Unique access and management console**

 **Zero inbound connections**
no open ports on the device

 **Customizable access control**
role based policies

 **Works behind NATs**
No need to maintain long IP addresses lists

 **100% web based**
no need to install client software

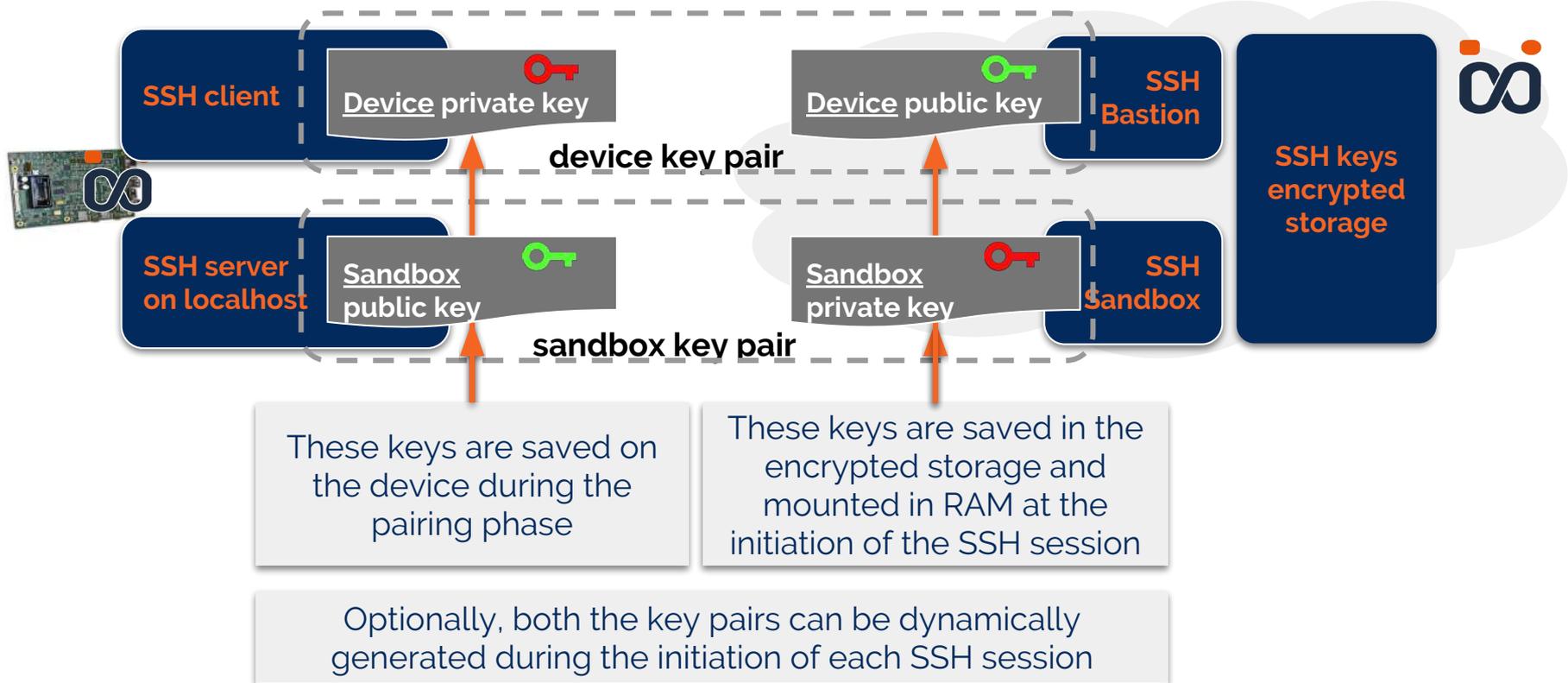
 **Access enabled by the device** via MQTT

 **Automatic activity auditing and reporting**

 **Works with devices connected to mobile networks** GPRS, 3G, 4G



everywhere web SSH - keys



TomorrowData



Tomorrowdata Srl

Strada Basse di Dora 42 - 10146 Torino

tomorrowdata.io

info@tomorrowdata.io

iottly@tomorrowdata.io

Keep in touch!

 @stefanoterna

 stefano.terna@tomorrowdata.io

 stefanoterna

 stefanoterna

 @danielegrieco

 daniele.grieco@tomorrowdata.io

 danielegrieco