TinyContainer

an abstraction and ressource control layer for containers on RIOT

Chrystel Gaber & Samuel Legouix

RIOT Summit 2023

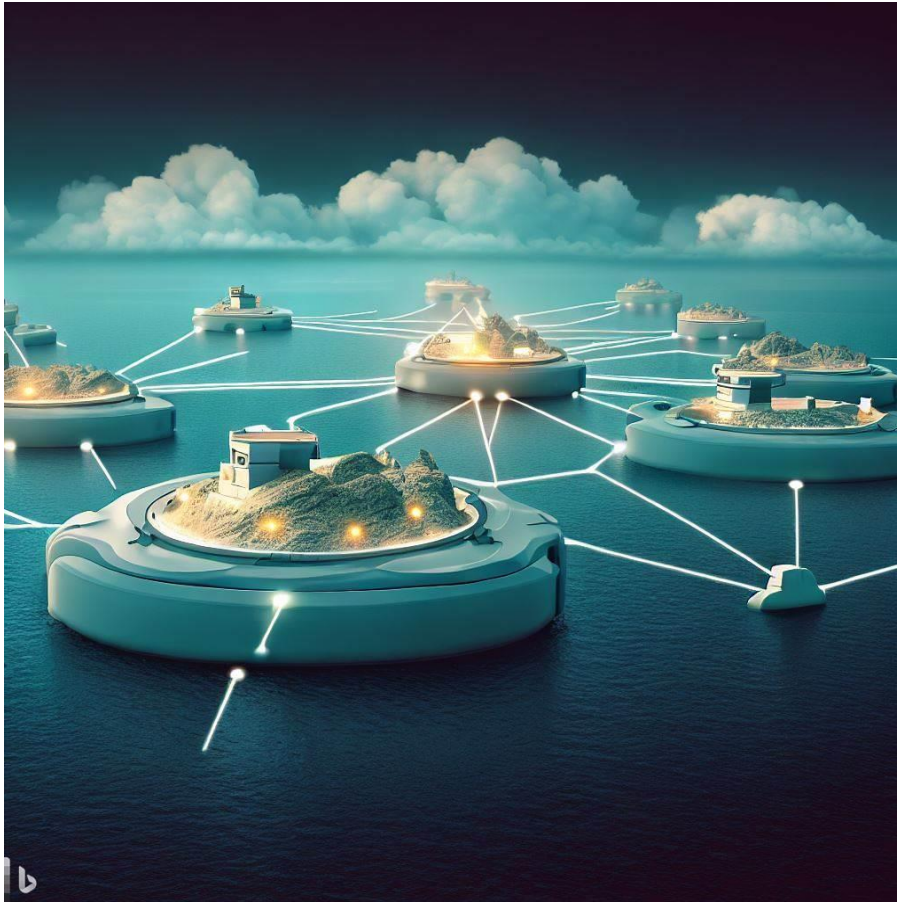
https://github.com/TinyPART/RIOT/tree/tinycontainer

TinyPART

TinyContainer in a nutshell

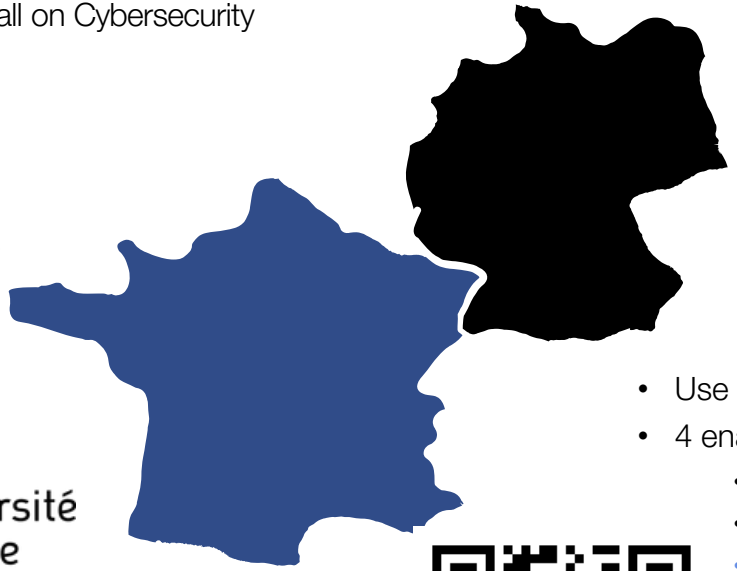
Think of containers on IoT devices as small, fortified self-contained islands in a vast digital archipelago. Each of them houses unique resources within its secure borders.

TinyContainer is a solution that builds bridges to connect these islands and enforces only authorized passage on them, enabling the controlled exchange of resources.



TinyPART Consortium

- German-French project in the joint Call on Cybersecurity
- 5 partners



Freie Universität Berlin

PHYSEC SECURITY FOR THINGS

Inria

orange

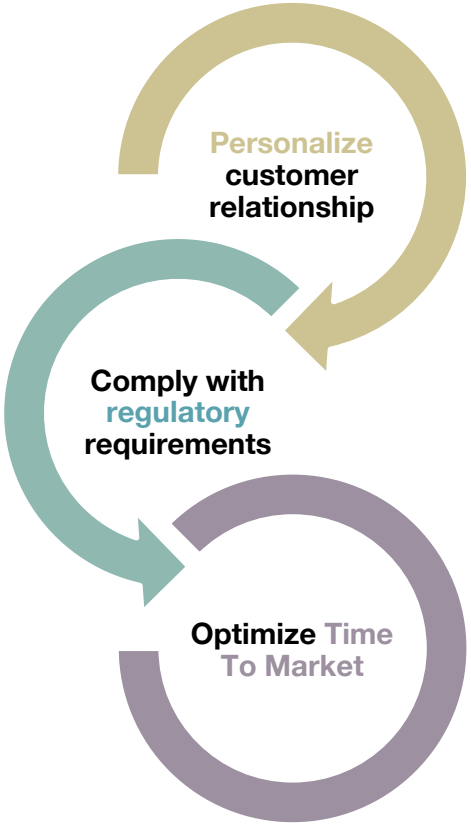
Université de Lille

- Use Case : Federated learning (FUB–PhySec)
- 4 enablers
  - PIP-MPU (Orange & UnivLille)
  - RIOT (INRIA)
  - TinyContainer (Orange & Inria & FUB)
  - Differential Privacy (FUB)

https://tinypart.github.io/TinyPART/

TinyPART : Industry needs

- EU GDPR requires to not expose user private data
- EU Cybersecurity Act requires certification of IoT devices
- EU CyberResilience Act reinforces Supply Chain Security
- Certification level depends on the risk of a specific usage context (e.g. risks on privacy or injuries)
- Contexts evolve and may require adaptation to new levels of risk (e.g. fire)



- Personalization of services make the customer feel unique and is a customer engagement driver,
- In post-Covid world, customers expectations blend human and digital,
- Privacy becomes a selling point.

Development and security require time and expertise

TinyPART : Our goals

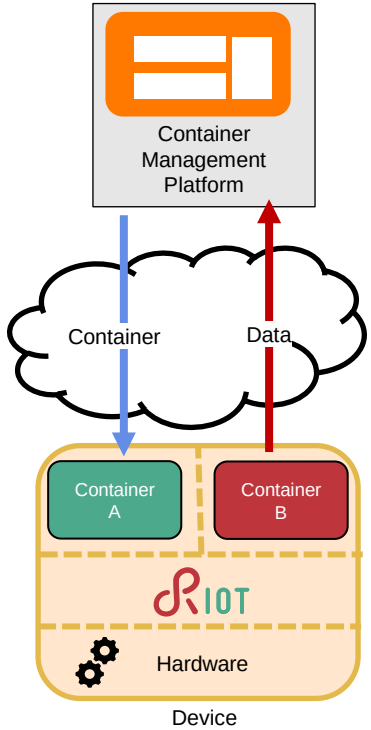
Investigate

- containers to adapt IoT logic to context
- capabilities to isolate untrusted IoT logic from multiple parties

Strategy

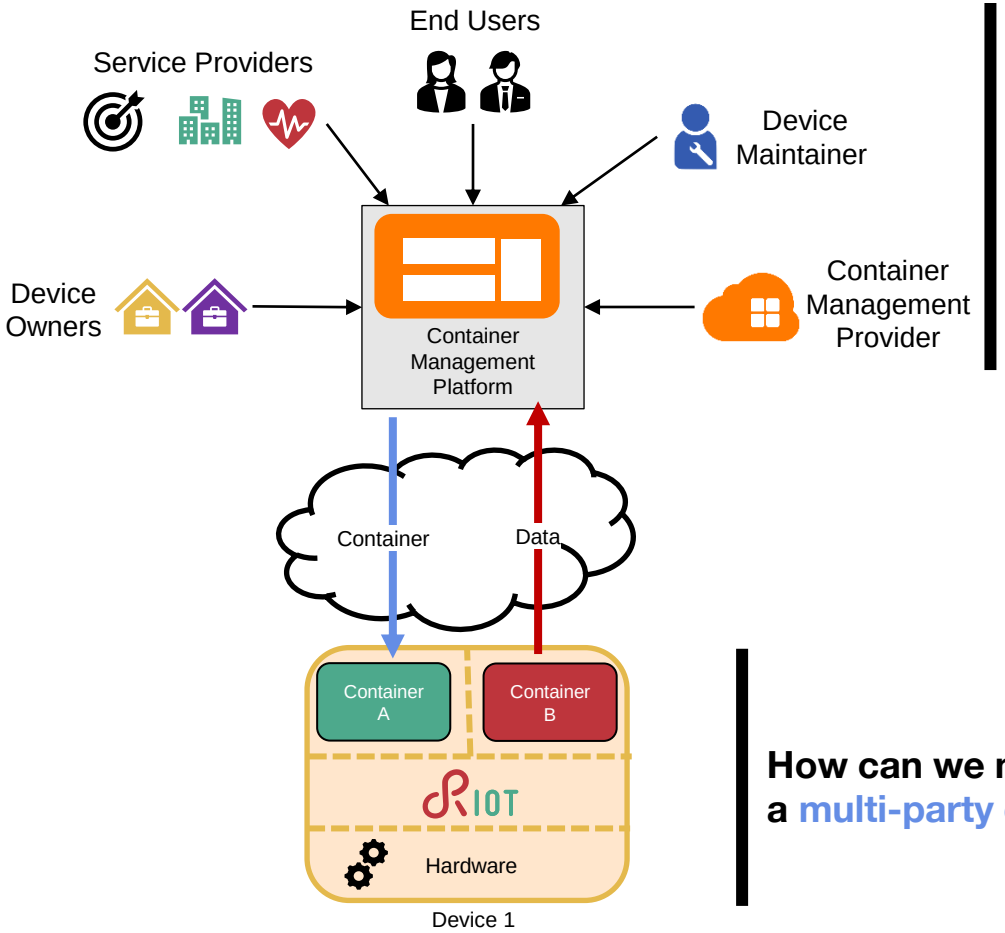
- secure & private-by-design architecture
- open-source and modular development bricks

TinyPART: Context



How can we manage isolation in a multi-party constrained device ?

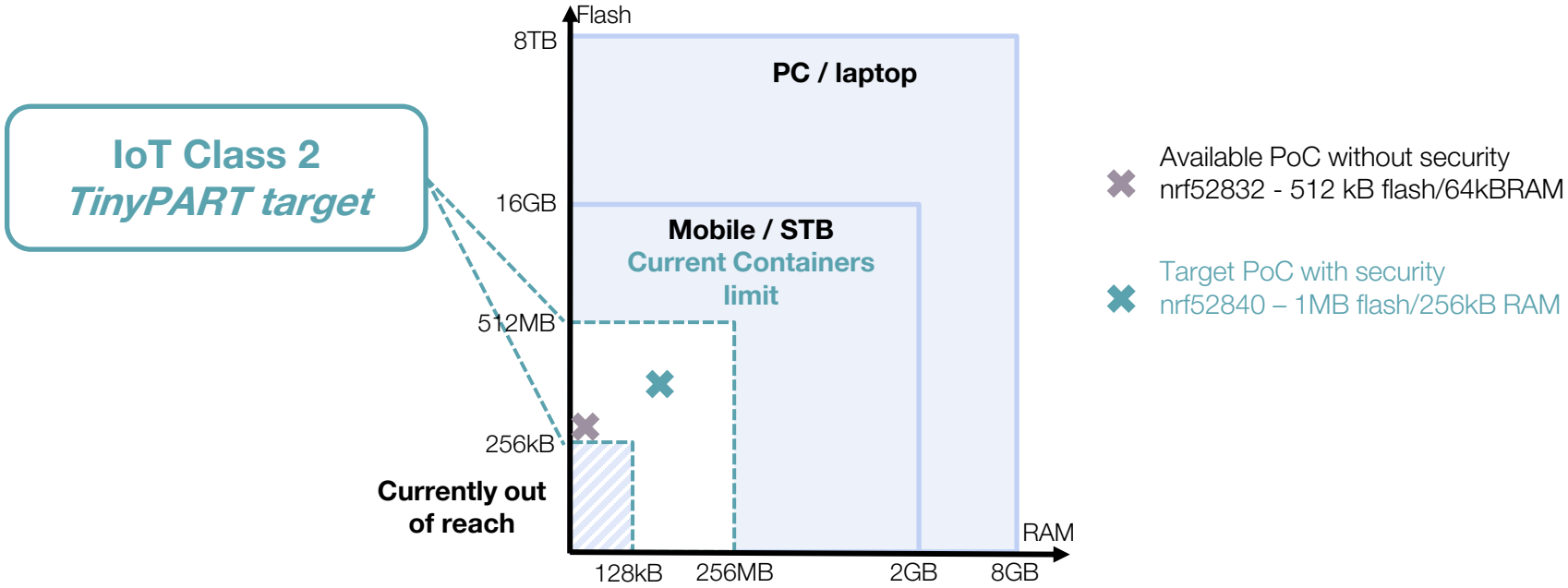
TinyPART: Context



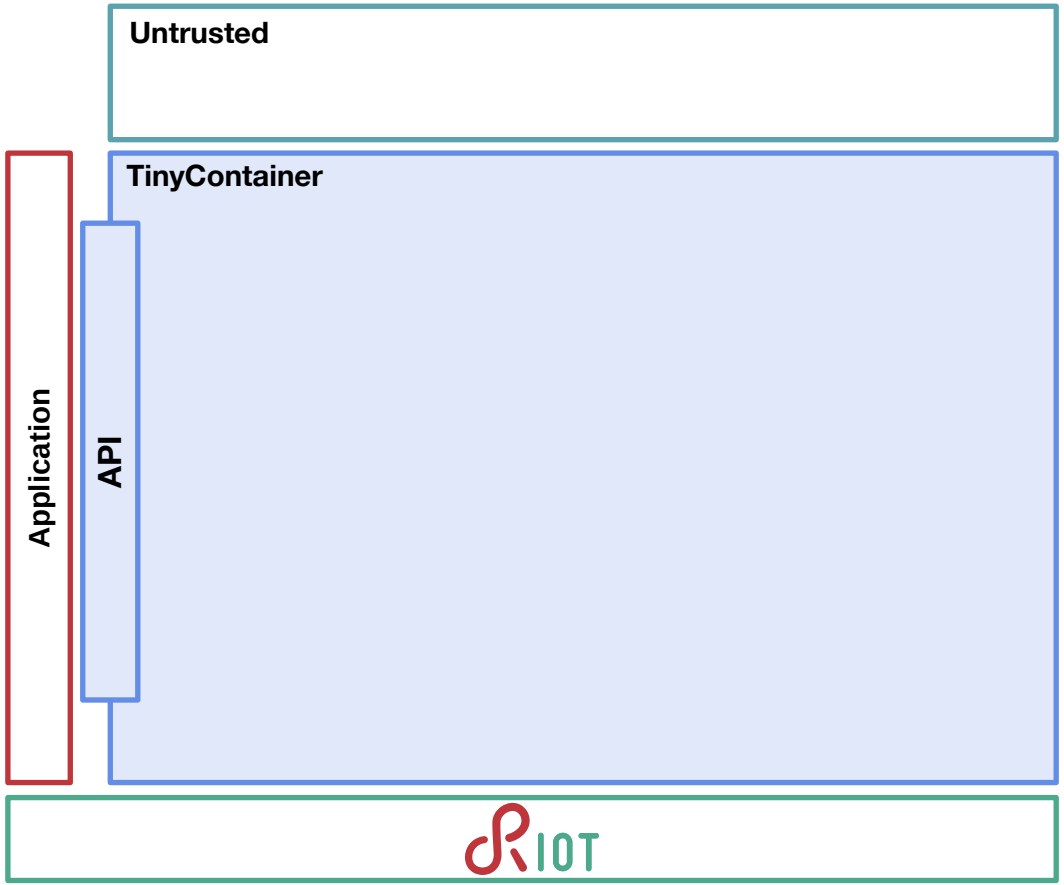
Roles

How can we manage isolation in a multi-party constrained device ?

Target: Low-power IoT



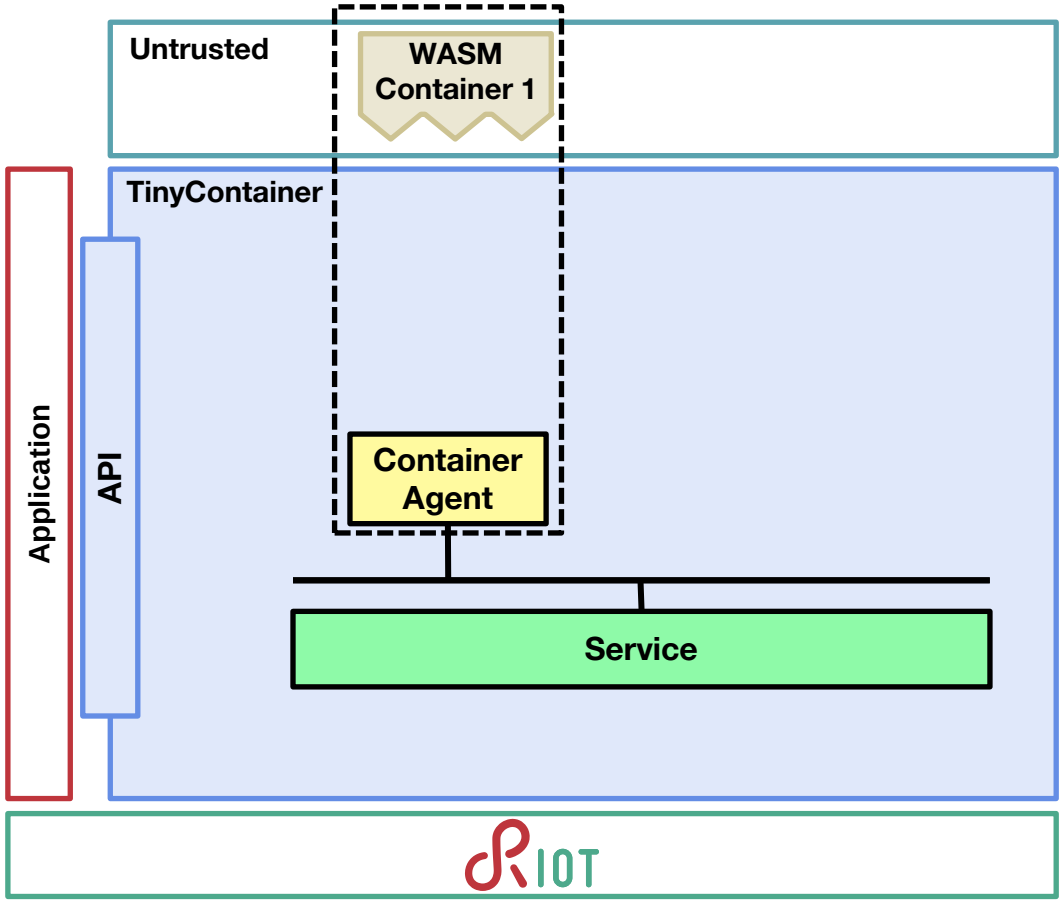
TinyContainer



API

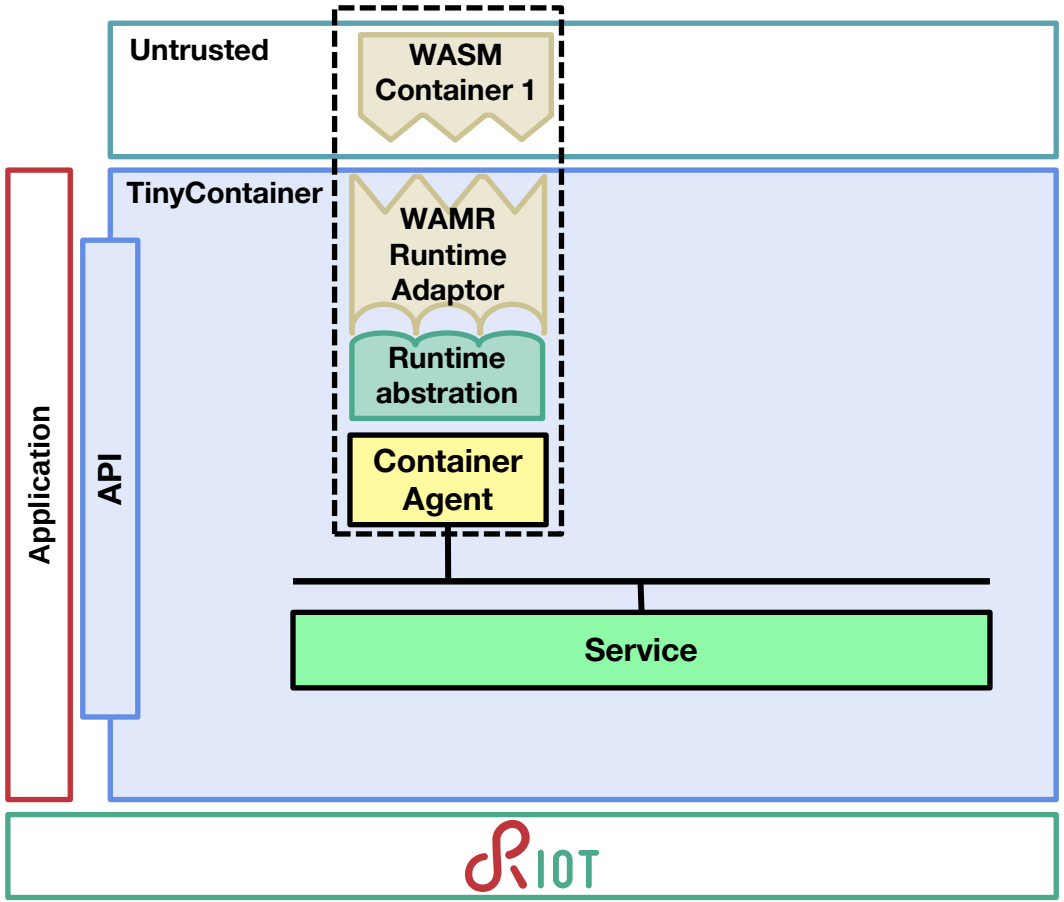
- init()
- loadcontainer()
- startcontainer()
- stopcontainer()
- iscontainerrunning()

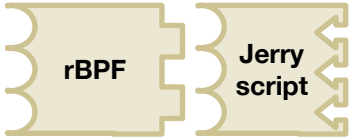
TinyContainer



1) Container lifecycle management

TinyContainer

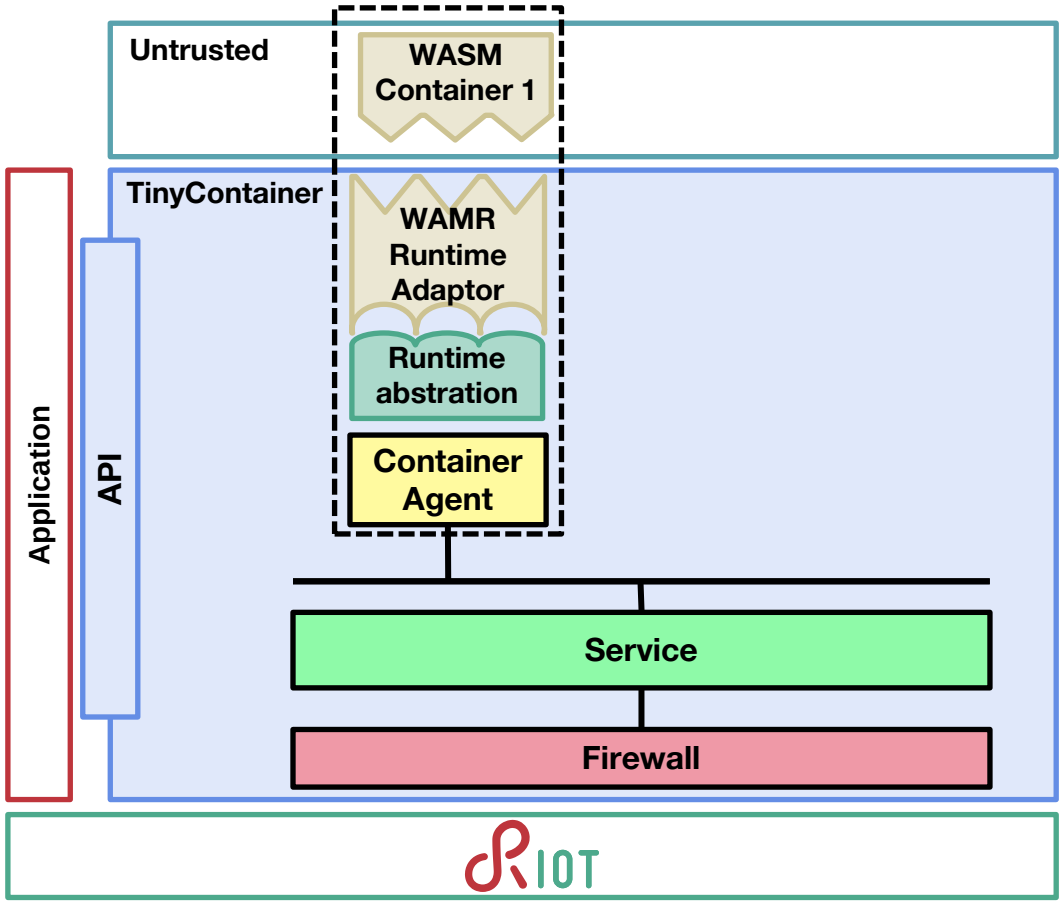


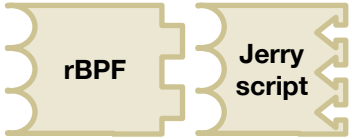


1) Container lifecycle management

2) Runtime Abstraction

TinyContainer



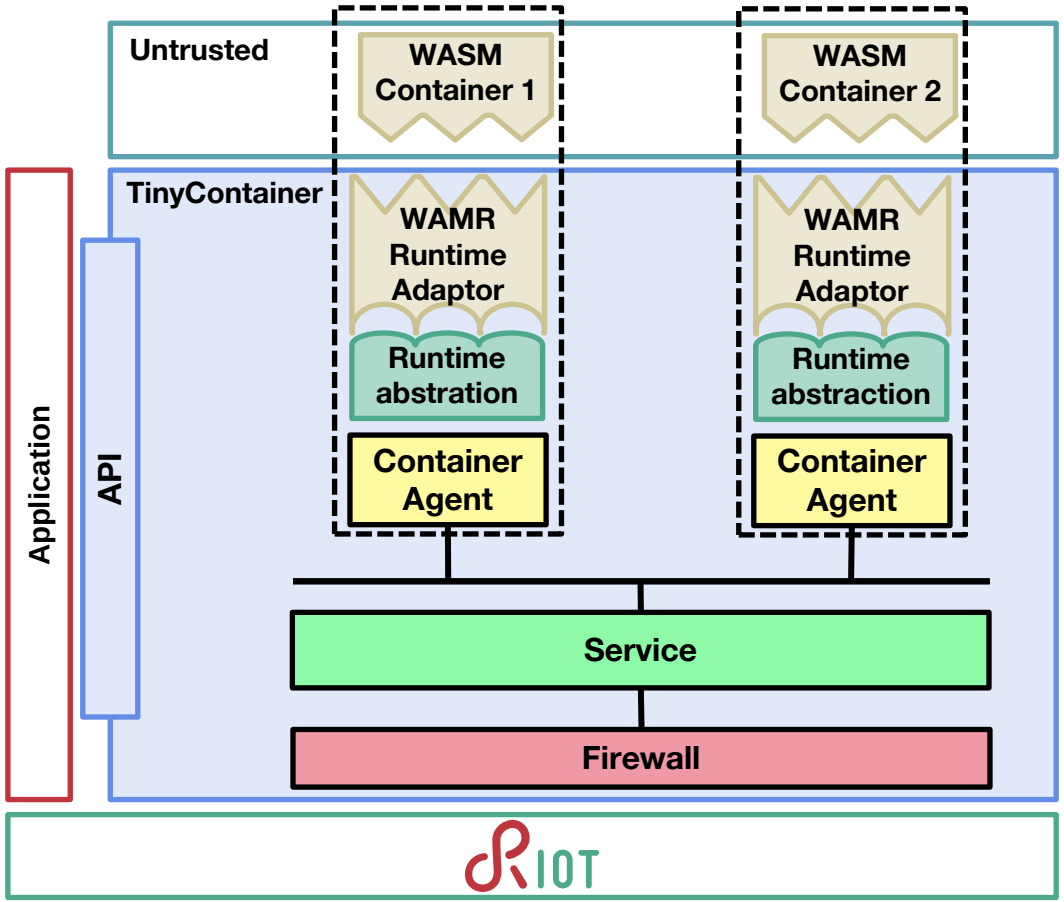


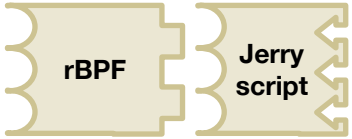
1) Container lifecycle management

2) Runtime Abstraction

3) Ressources access management

TinyContainer



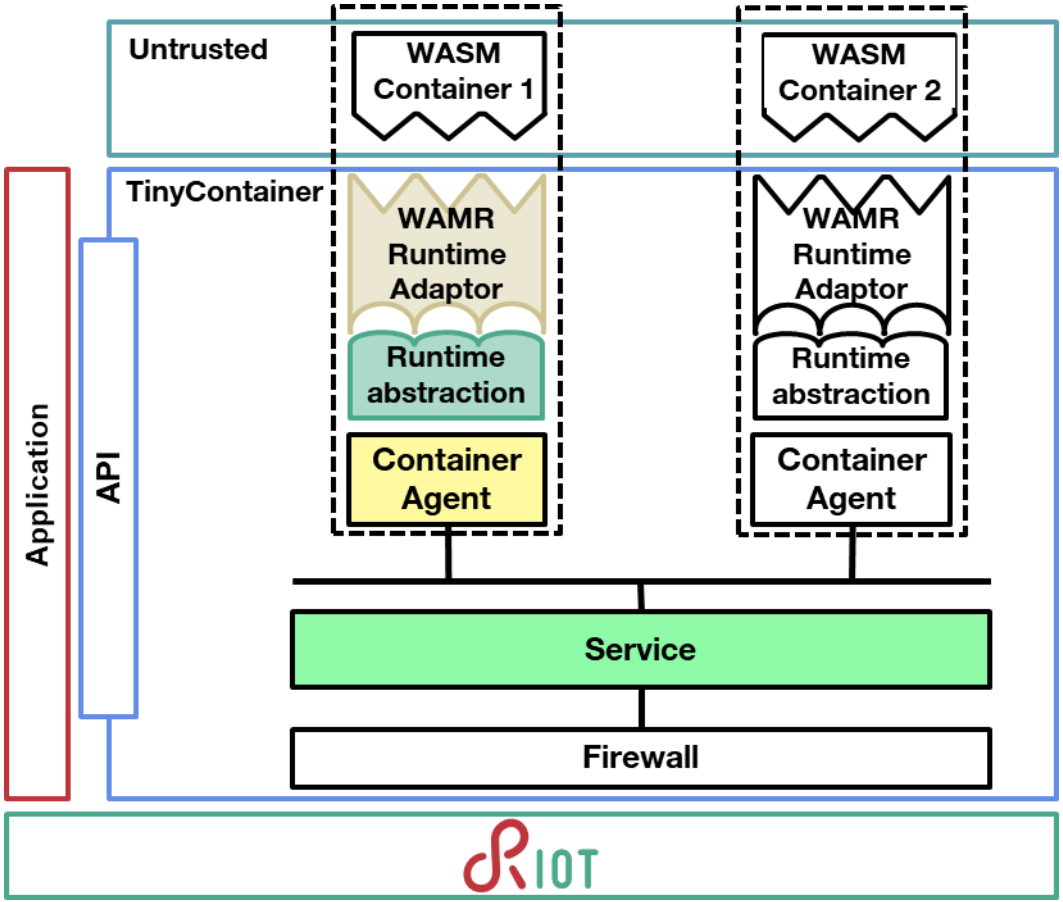


1) Container lifecycle management

2) Runtime Abstraction

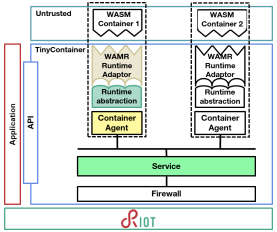
3) Ressources access management

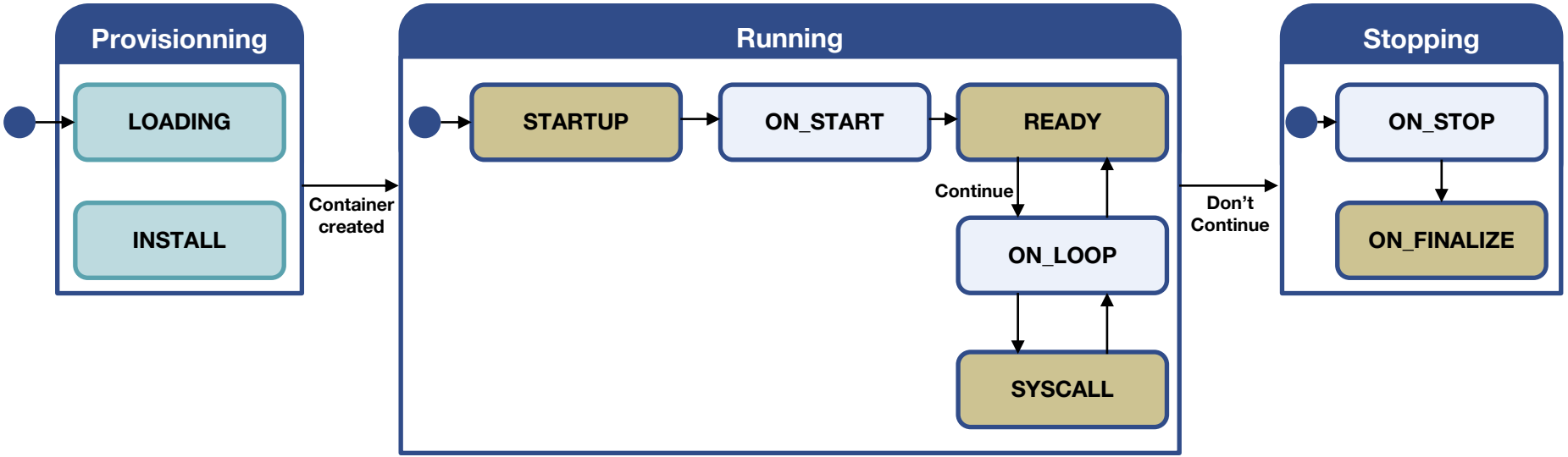
4) Multi-container support



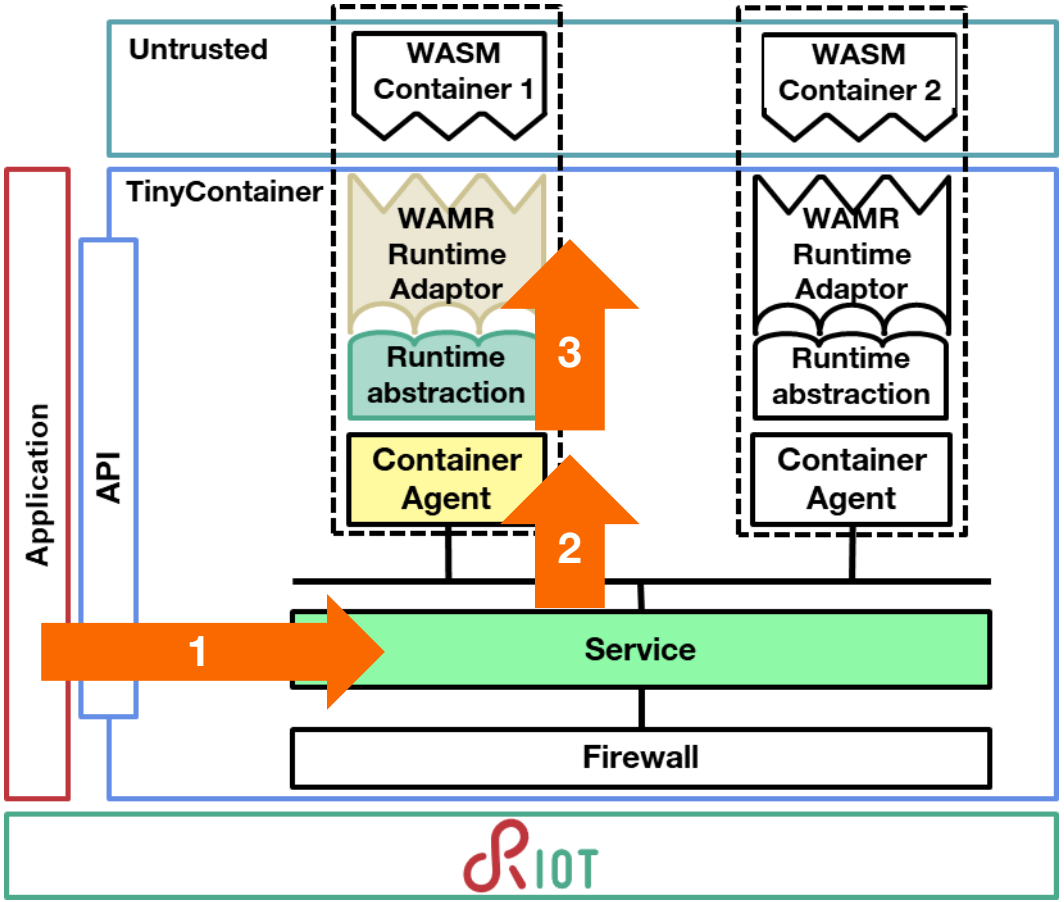
Using TinyContainer

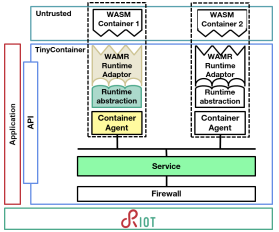
Container state machine





Create a container



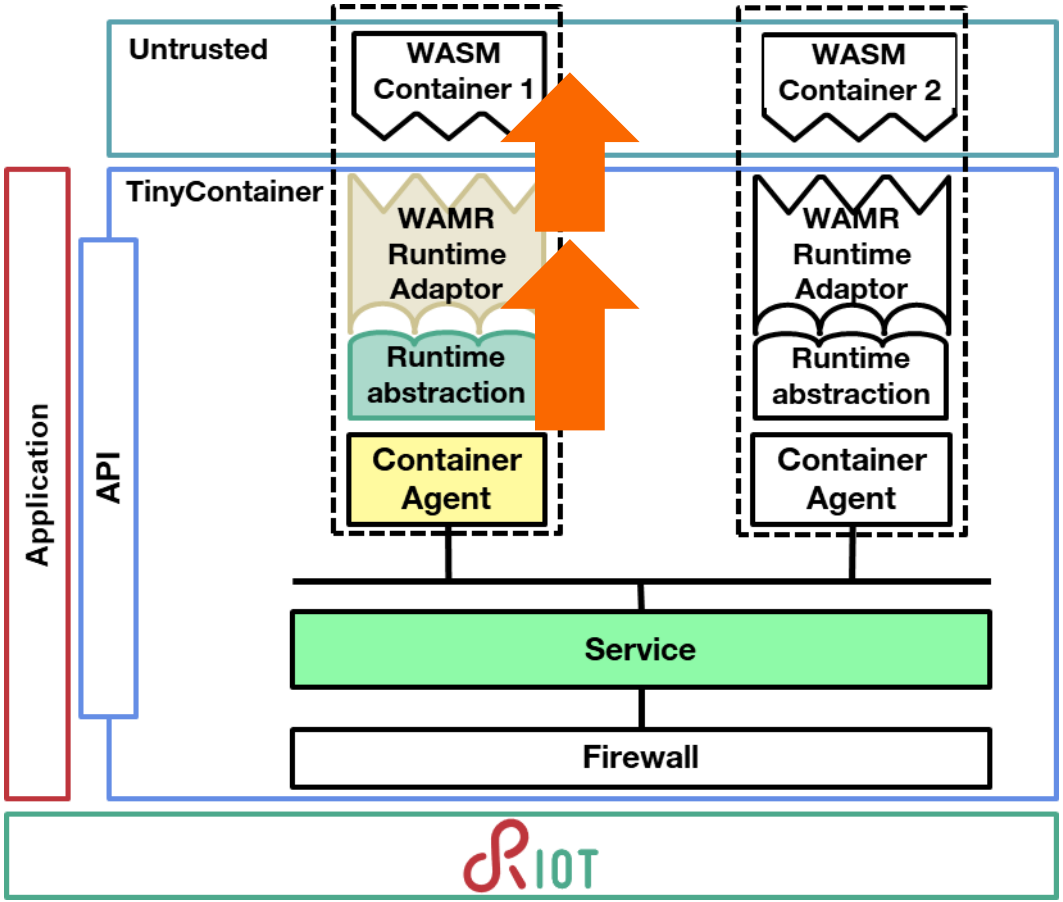


1) Provision container

2) Create thread & container agent

3) Load & Start Container

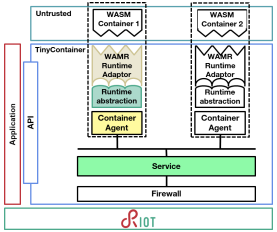
During the container life



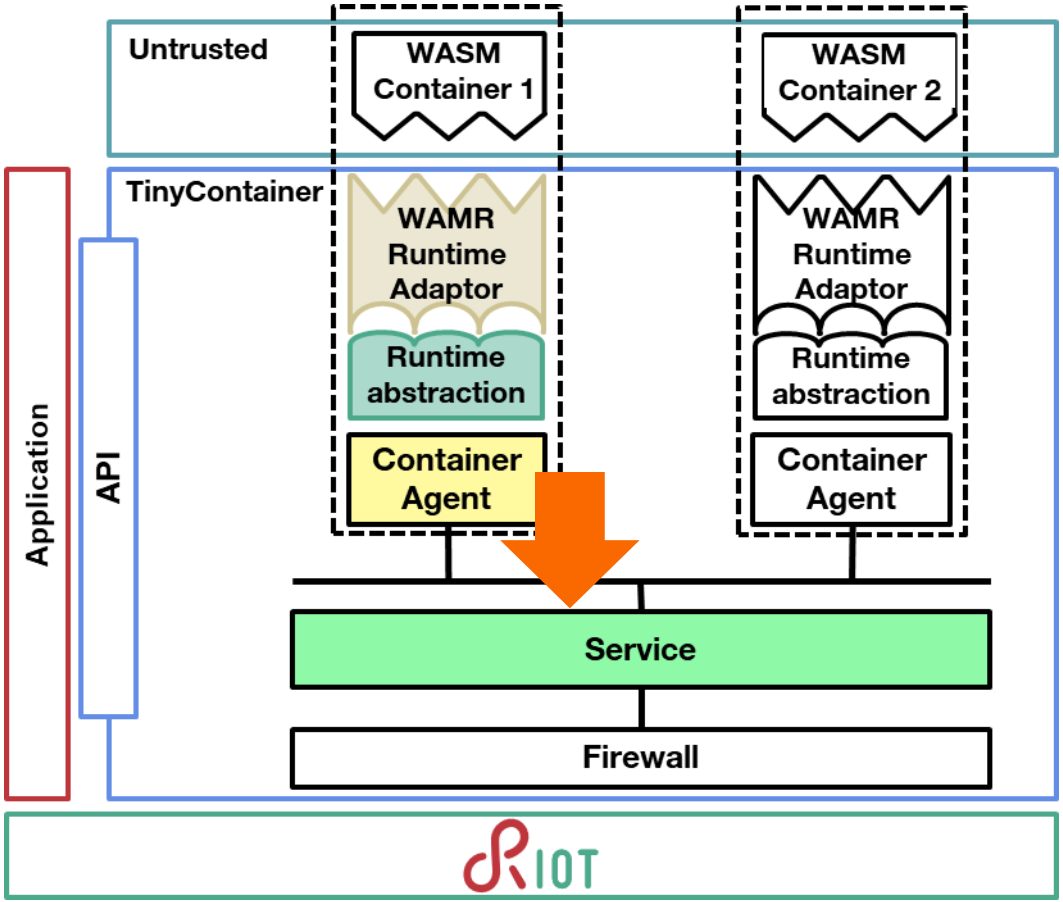
onstart()

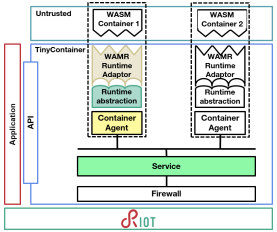
onstop()

onloop()



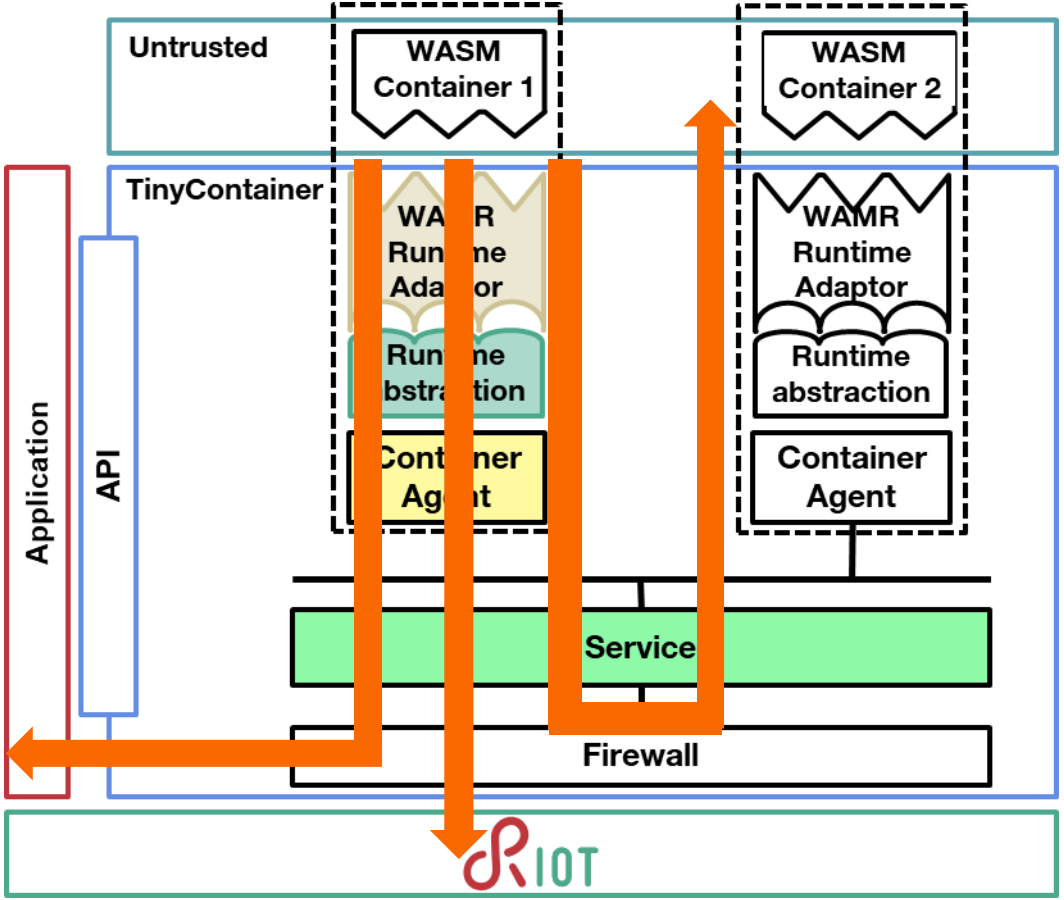
Periodic handover

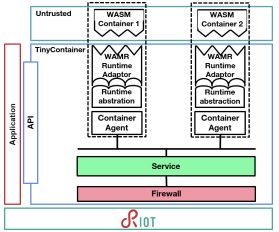




heartbeat()

Ressources exposed to containers





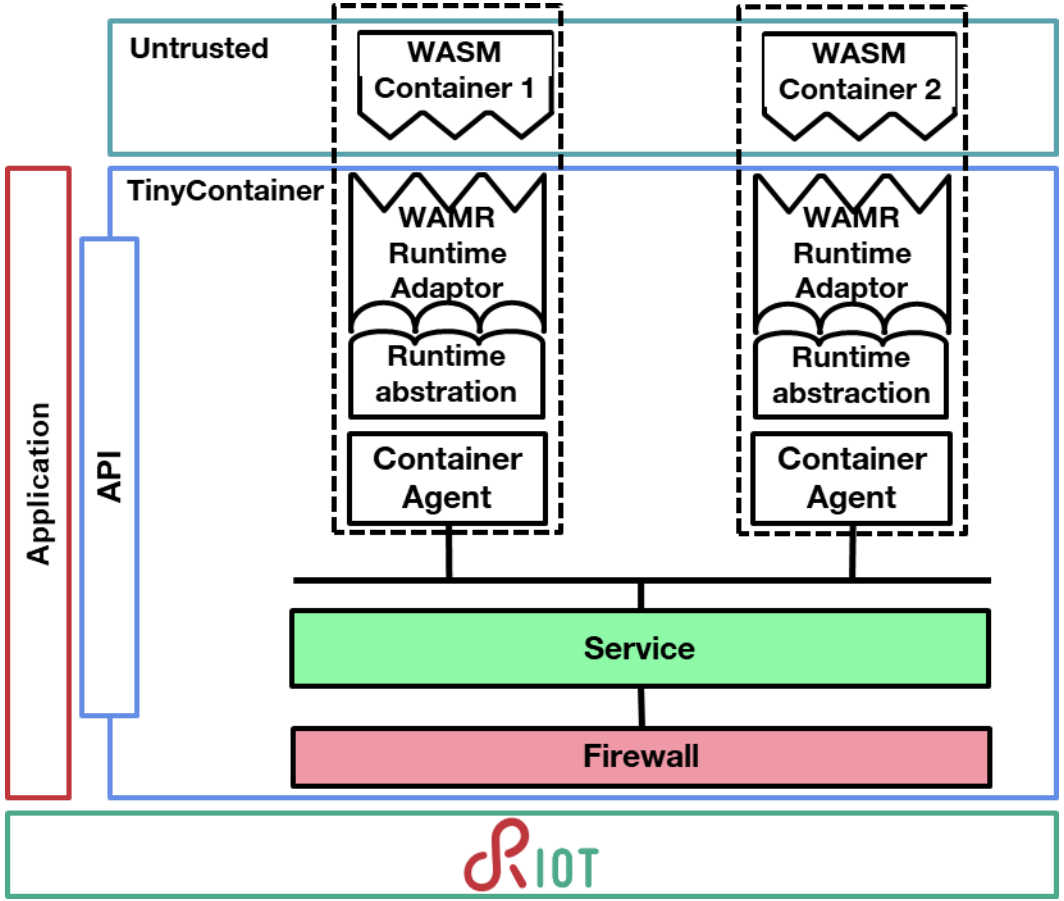
Ressources are syscalls or endpoints

- Local
- Container
- Remote

Service : Manage endpoints

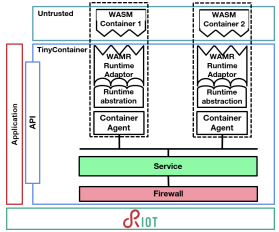
- Open
- Read
- Write
- Close

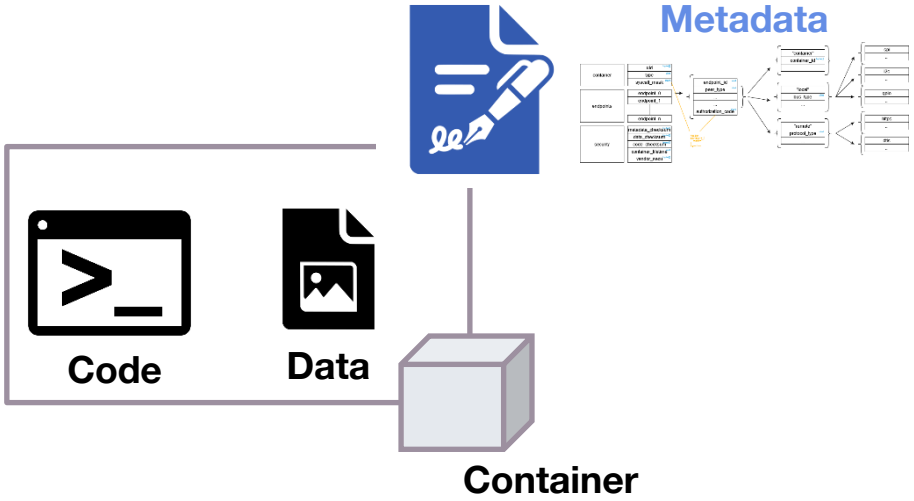
Firewall : Authorize calls



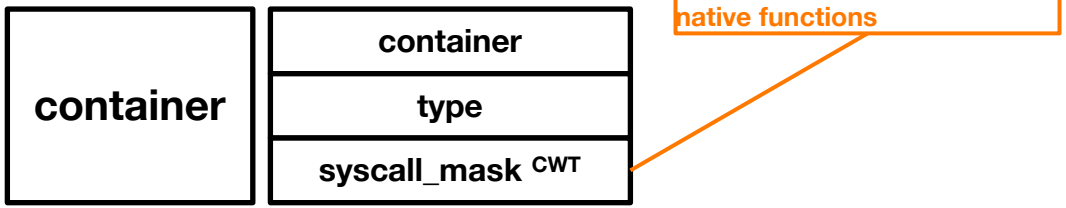
Endpoints & access rights management

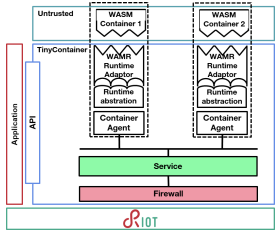
What is a container ?



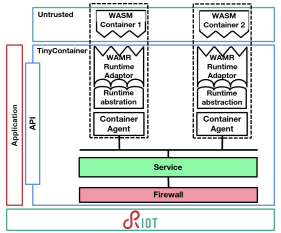


Metadata





Metadata



container

- container
- type
- syscall_mask CWT

endpoints

- endpoint_0
- endpoint_1
- endpoint_n

syscall_mask => rights to native functions

- endpoint_id
- peer_type
- ...
- ...
- authorization_code CWT

authorization_code =
Header
Payload=
{
endpoint_id
peer_type
...
},
can be provided by a 3rd party

- container
- container_id
- ...

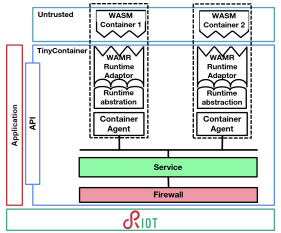
- local
- bus_type
- ...

- spi
- ...
- i2c
- ...
- gpio
- ...

- remote
- protocole_type
- ...

- https
- ...
- dtls
- ...

Metadata



- **container**
  - container
  - type
  - syscall_mask CWT
- **endpoints**
  - endpoint_0
  - endpoint_1
  - endpoint_n
- **security**
  - metadata_checksum
  - data_checksum
  - code_checksum
  - container_lifetime
  - vendor_secu

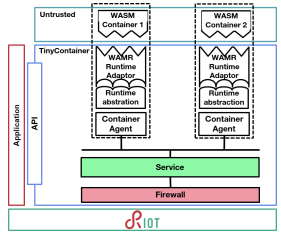
syscall_mask => rights to native functions

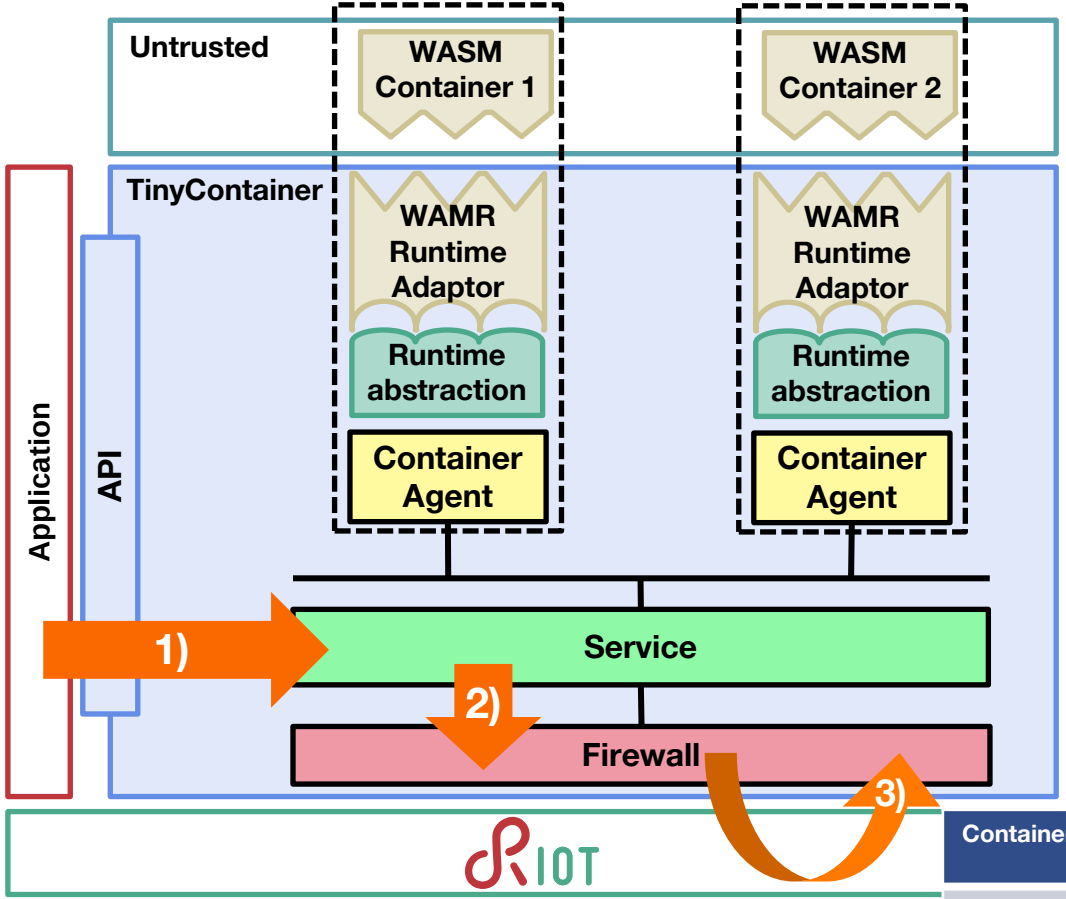
- endpoint_id
- peer_type
- ...
- ...
- authorization_code CWT

authorization_code =
Header
Payload=
{
  endpoint_id
  peer_type
  ...
},
can be provided by a 3rd party

- container
  - container_id
  - ...
- local
  - bus_type
  - ...
    - spi
      - ...
    - i2c
      - ...
    - gpio
      - ...
- remote
  - protocole_type
  - ...
    - https
      - ...
    - dtls
      - ...

Initialize container rights





1) Provision container

2) Hand over to Firewall

3) Configure rights
   a) Verify Checksums & tokens
   b) Assign ID to container
   c) Fill in container table

Container	ID	syscall mask	endpoints

Next steps

Performance & memory usage

Root of Trust

Integration with Pip-MPU

Food for discussion

Feedbacks & use cases

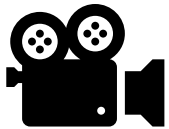
VM/interpreter syscall interface for bindings

Towards an application firewall ?

https://forum.riot-os.org/t/vm-interpreter-syscall-interface-for-bindings/3902

Live :
TinyContainer
repo & helloworld

Demos



Recorded:
MyCoach use
case

Demo : MyCoach

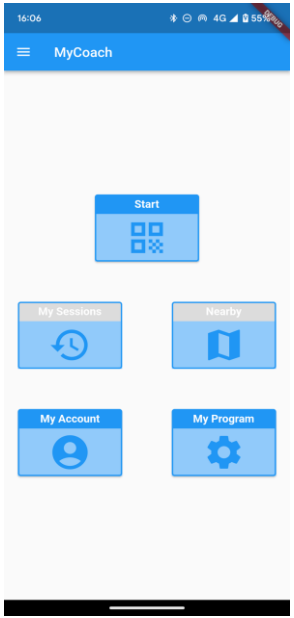
End User

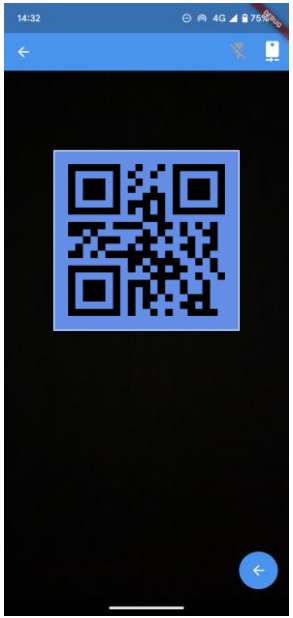
Device Owner

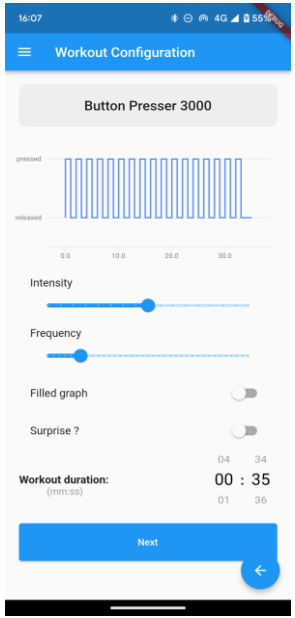
Service Provider

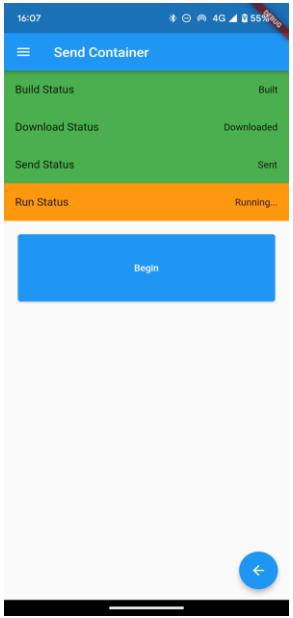
Container Management Provider

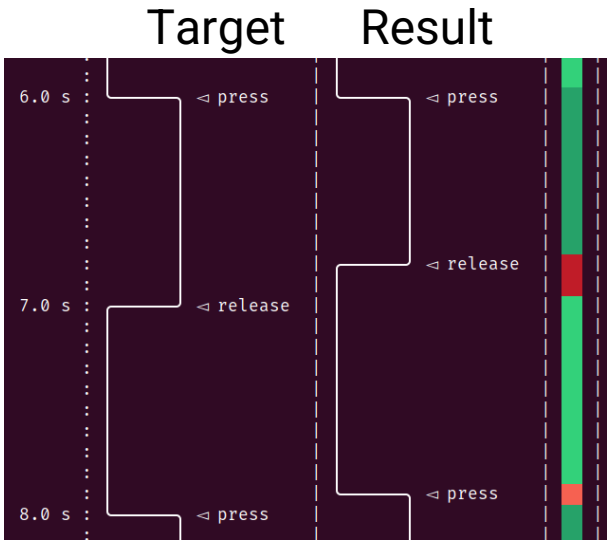
Demo : MyCoach

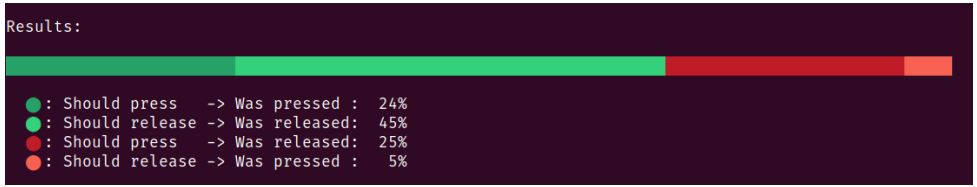




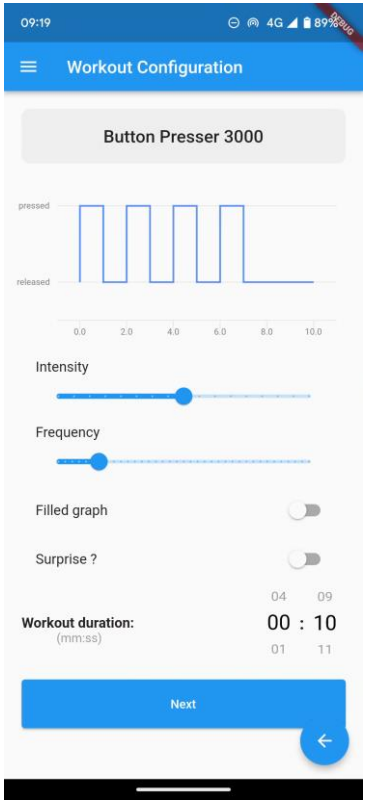




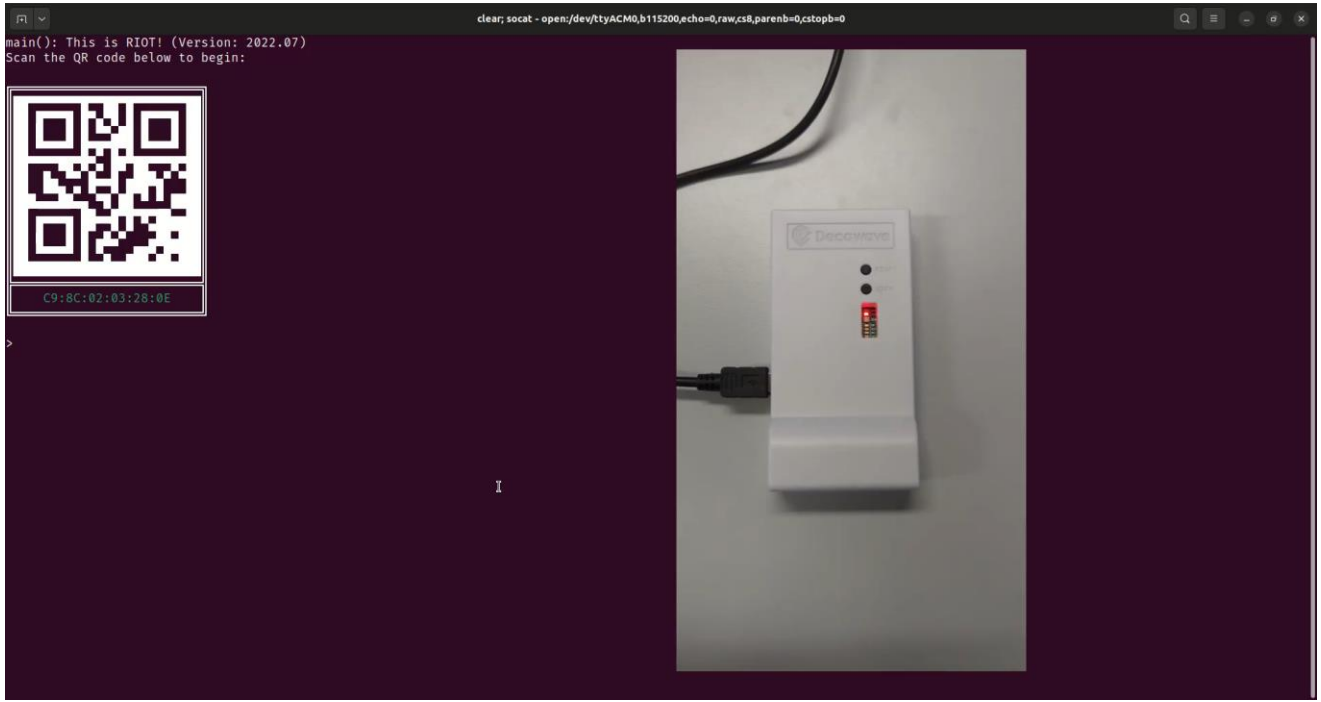




Demo : MyCoach



Before provisionning



After provisionning

Thank You

https://github.com/TinyPART/RIOT/tree/tinycontainer