Comparative presentation of eclipse frameworks for Fog/Edge computing:

February 2020

and

Speaker: Franck Roudet – Orange Labs Grenoble

Authors:

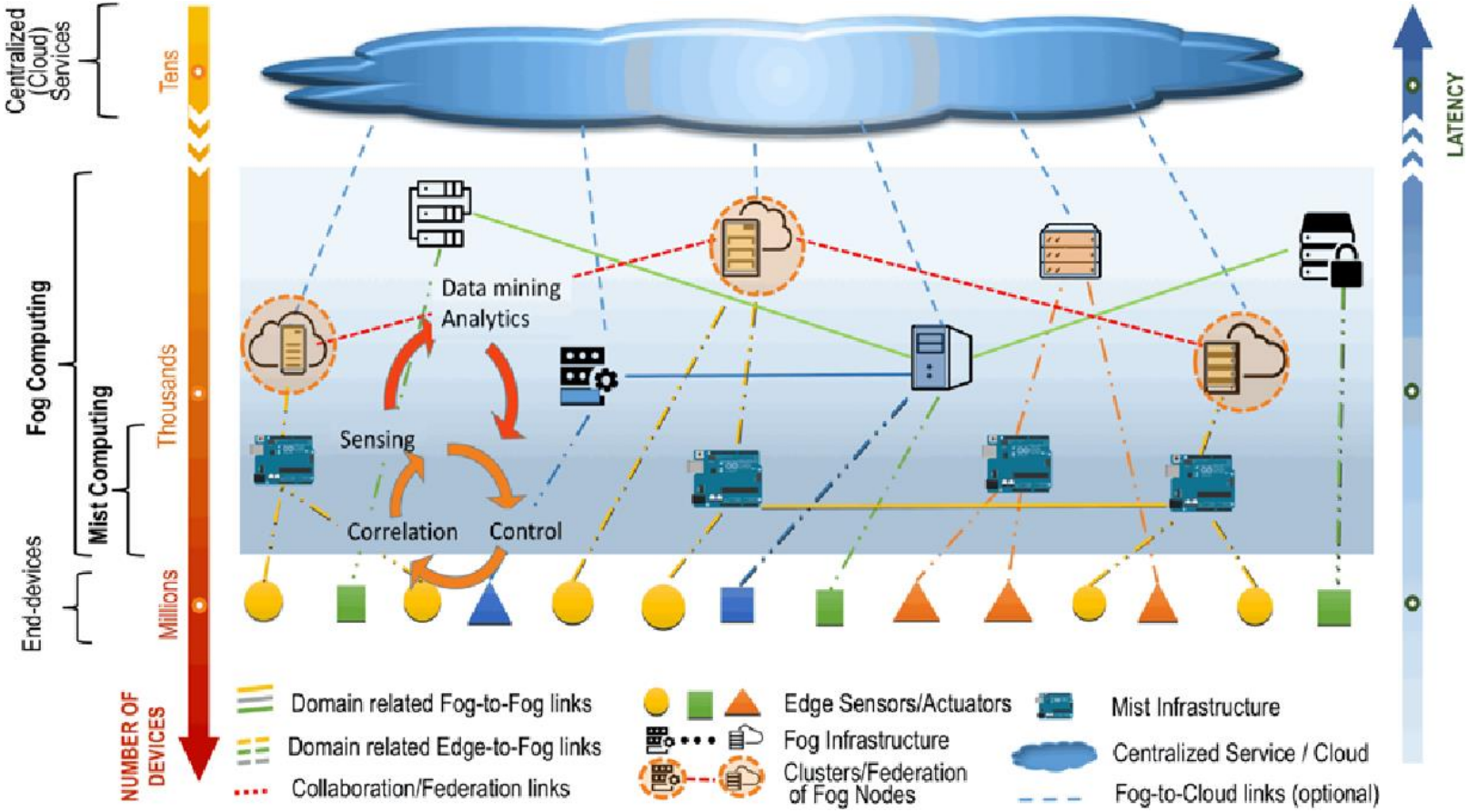
Yanjun CHEN – Orange Labs China Beijin

Rui ZHOU, Jingyang CHEN

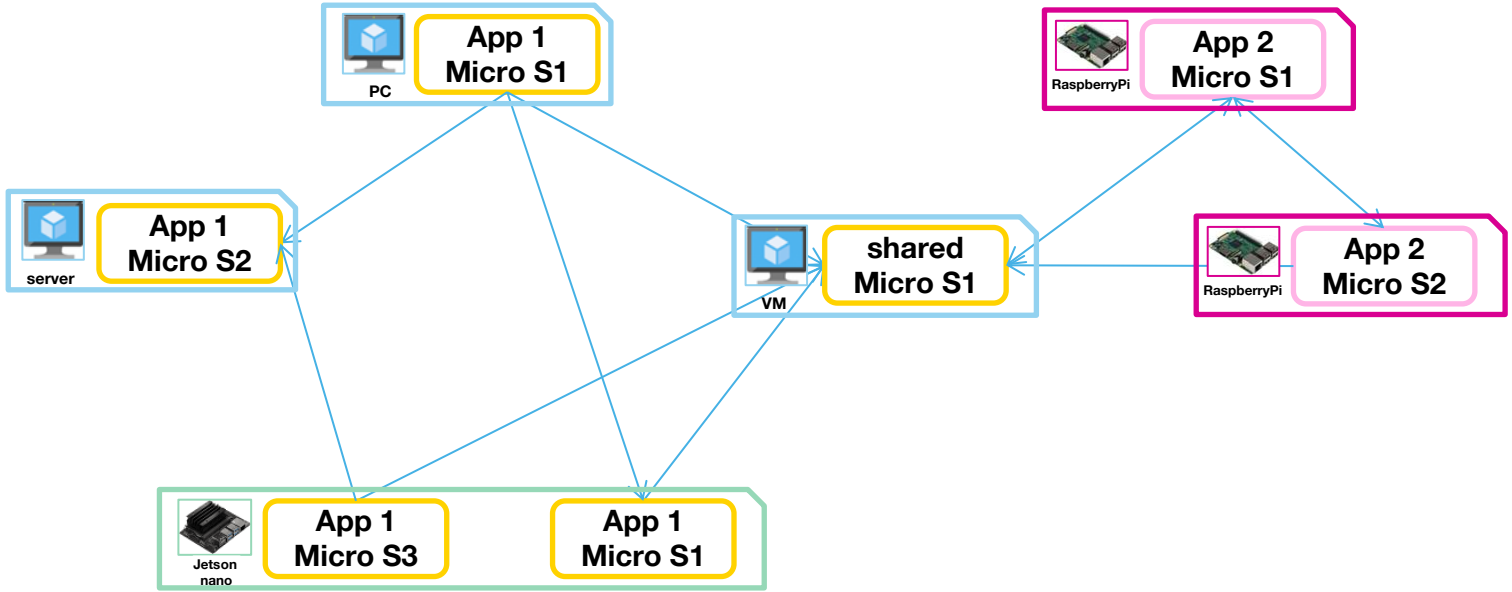
orange™

Orange

Fog Computing Conceptual Model: NiST SP 500-325 & 500-291



Generic fog application software architecture ➔ projection on HW/network



Outline

- Edge Computing in the Eclipse Foundation
- ioFog Introduction
- Fog05 Introduction
- Comparisons
- Summary

Edge Computing in the Eclipse Foundation

Eclipse Projects related with Edge Computing and belonging to Eclipse IoT

- Eclipse kura Eclipse Kura™ is an extensible open source IoT Edge Framework based on Java/OSGi.
- Eclipse ioFog Eclipse ioFog is a complete edge computing platform that provides all of the pieces needed to build and run applications at the edge at enterprise scale
- Eclipse fog05 The End-to-End Compute, Storage and Networking Virtualisation solution.

Eclipse edge native working group

- Public announcement on Dec.2019
- A separation from IoT initiatives, address particular challenges associated with edge computing in a very focused way
- ioFog and fog05 are two flagship projects
  - Collaboration or merging of the two projects are considered in the roadmap, but how and what to merge are not discussed no early than 2020 Jan

https://www.eclipse.org/community/eclipse_newsletter/2019/october/discover.php

Eclipse ioFog

ioFog - Project Overview

ioFog is an open source project in Eclipse foundation which parent project is eclipse IoT project.

Licenses: Eclipse Public License 2.0

The project source code repositories: https://github.com/eclipse-iofog

Latest Releases:

- **v1.3.0 2019-10-21**
- **v1.2.0 2019-08-08**
- **v1.1.0 2019-06-19**

Source code language: Java, Javascript, Go

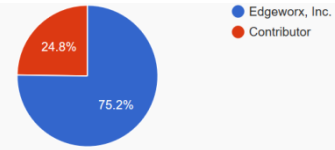
Project Leading Contributors: edgeworx

edgeworx

https://edgeworx.io/

ECLIPSE INCUBATION

ioFog



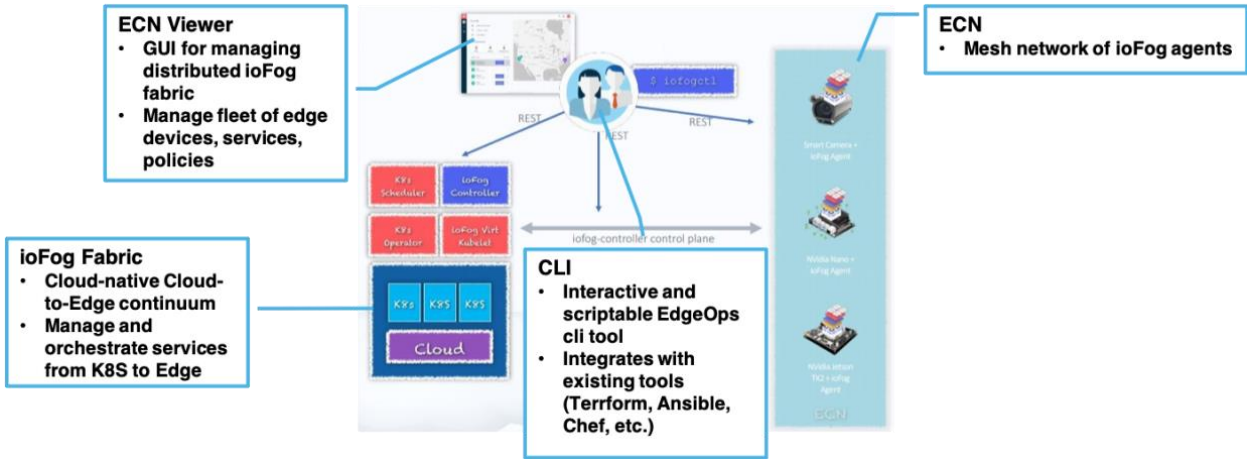
ioFog Overview

ioFog is an edge computing platform providing components to deploy and distribute microservices in Edge Compute Network (ECN) using container based approach.

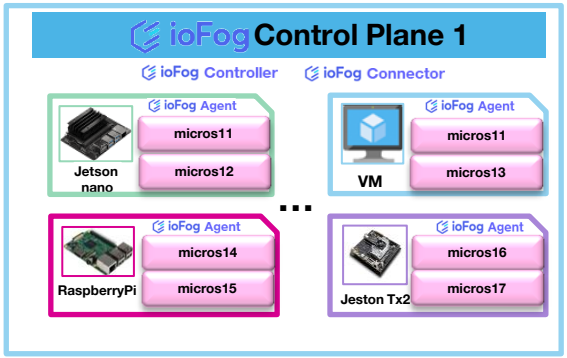
- **Controller: management and orchestration of different agents**
- **Connector: communications of microservices between different nodes**
- **Agent: a daemon service running in each (edge) node – monitoring & supervision**
- **Microservice: docker container doing business**

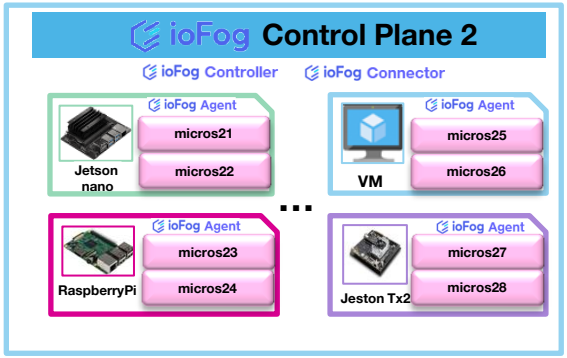
Other related tools or projects:

- **Command line tools: iofogctl**
- **ECN viewer**



ioFog Architecture





ioFog Components Main Features - Controller - Connector

Controller is used for orchestration of the different Agents.

- **Controller should be deployed in the node where network accessible for edge nodes (fixed IP address or DNS).**
- **One or multiple controllers could be used ➔ grouping agents**

Connector will enable the communication between different microservices.

- **The Connector assists in providing automatic discovery and NAT traversal, brokering communication when possible.**
- **Connector offers two connectivity types:**
  - public pipe punches through firewalls and NATed networks to perform automatic internetworking of the Fog.
  - private pipe, consumes bandwidth on the Connector but stabilizes connectivity

ioFog API swagger : https://iofog.org/docs/1.3.0/controllers/rest-api.html

Community: https://discuss.iofog.org/ Discourse

ioFog Components Main Features - Agent

Agent will handle the starting, stopping, and management of microservices running on the node.

- **a local API with REST-like endpoints as well as WebSockets, example interfaces:**
  - **Get container configuration, next unread message**
  - **Get messages from publishers within timeframe**
  - **Get control/message WebSocket connection**
  - **Get service status/info/version**
  - **Attach/detach ioFog agent to the configured ioFog controller**
  - **Change ioFog agent configuration**
- **All messages passing through the Local API must be in the ioMessage format (embodied in JSON, XML and Binary).**

Note: Agents could be remotely managed by controller after successfully connected which enable the deployment and maintaining of microservices without using SSH communication to contact each edge node.

ioFog Deployment Requirements

Agent	• Processor: x86-64 or ARM Dual Core or better • RAM: 256 MB minimum • Hard Disk: 100 MB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc) • Java Runtime v8.0.0 or higher • Docker v1.10 or higher
Controller	• Processor: x86-64 or ARM Dual Core or better • RAM: 1 GB minimum • Hard Disk: 5 GB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc), macOS 10.12+, or Windows 7+ • Node.js v8+ and NPM
Connector	• Processor: x86-64 or ARM Dual Core or better • RAM: 1 GB minimum • Hard Disk: 5 GB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc) • Java Runtime v8.0.0 or higher

Note: While ioFog fully support using Raspberry Pi's as workers on the edge environment, they are not meant to be used as the Controller and Connector infrastructure(specify using the Raspberry Pi as an agent in edge infrastructure).

ioFog Experimental Deployment

ioFog version: 1.3.0

Deployment environment:

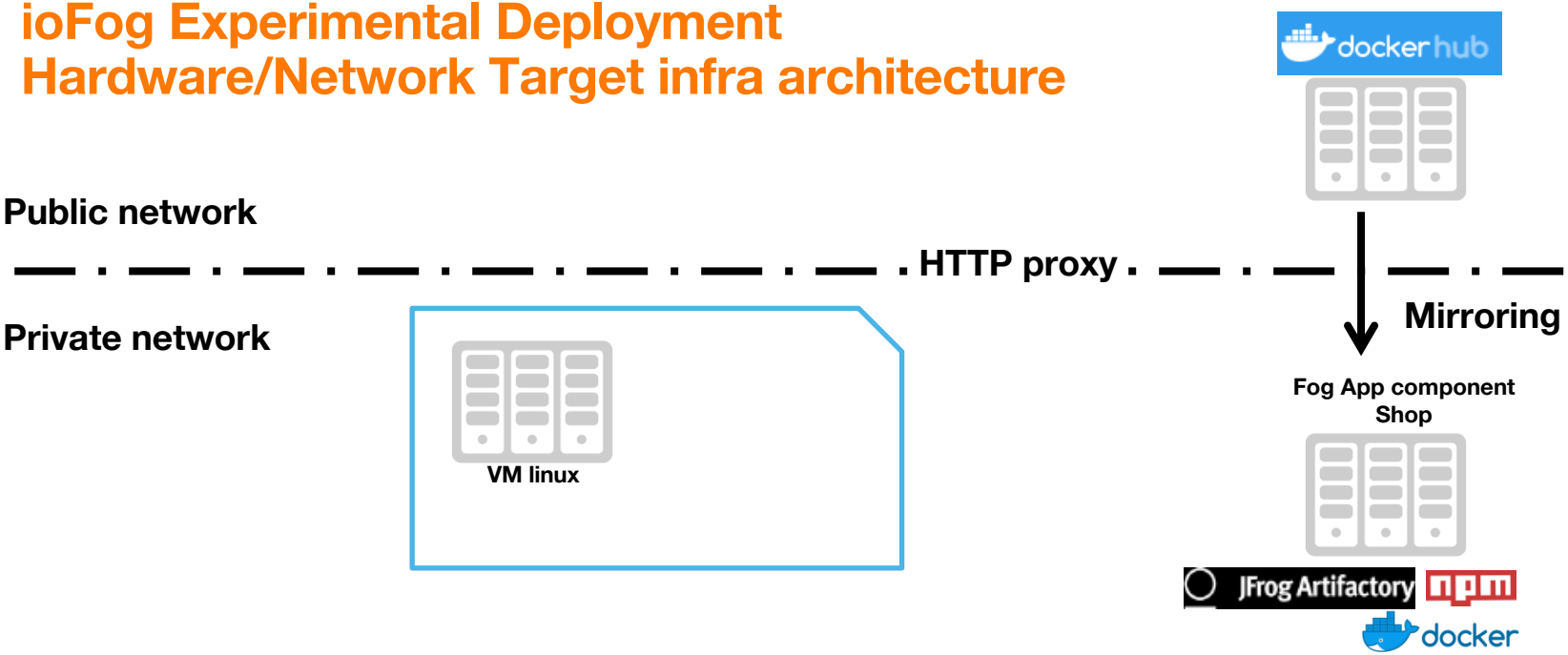
- **VM x84 4GB, 2vCPU, 40GB**
- **Raspberry Pi 3b+ /**
- **Nvidia jetson nano**

Deployment logic:

- **deploy 1 controller , 3 agents 1 connector**
- **deploy example application**

https://iofog.org/docs/1.3.0/getting-started/quick-start.html

ioFog Experimental Deployment
Hardware/Network Target infra architecture



Orange

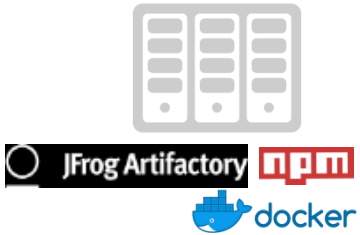
ioFog Experimental Deployment
ioFog SW deployment on Target infra architecture

Public network

HTTP proxy

Private network

ioFog Controller
ioFog Connector
ioFog Agent
VM linux



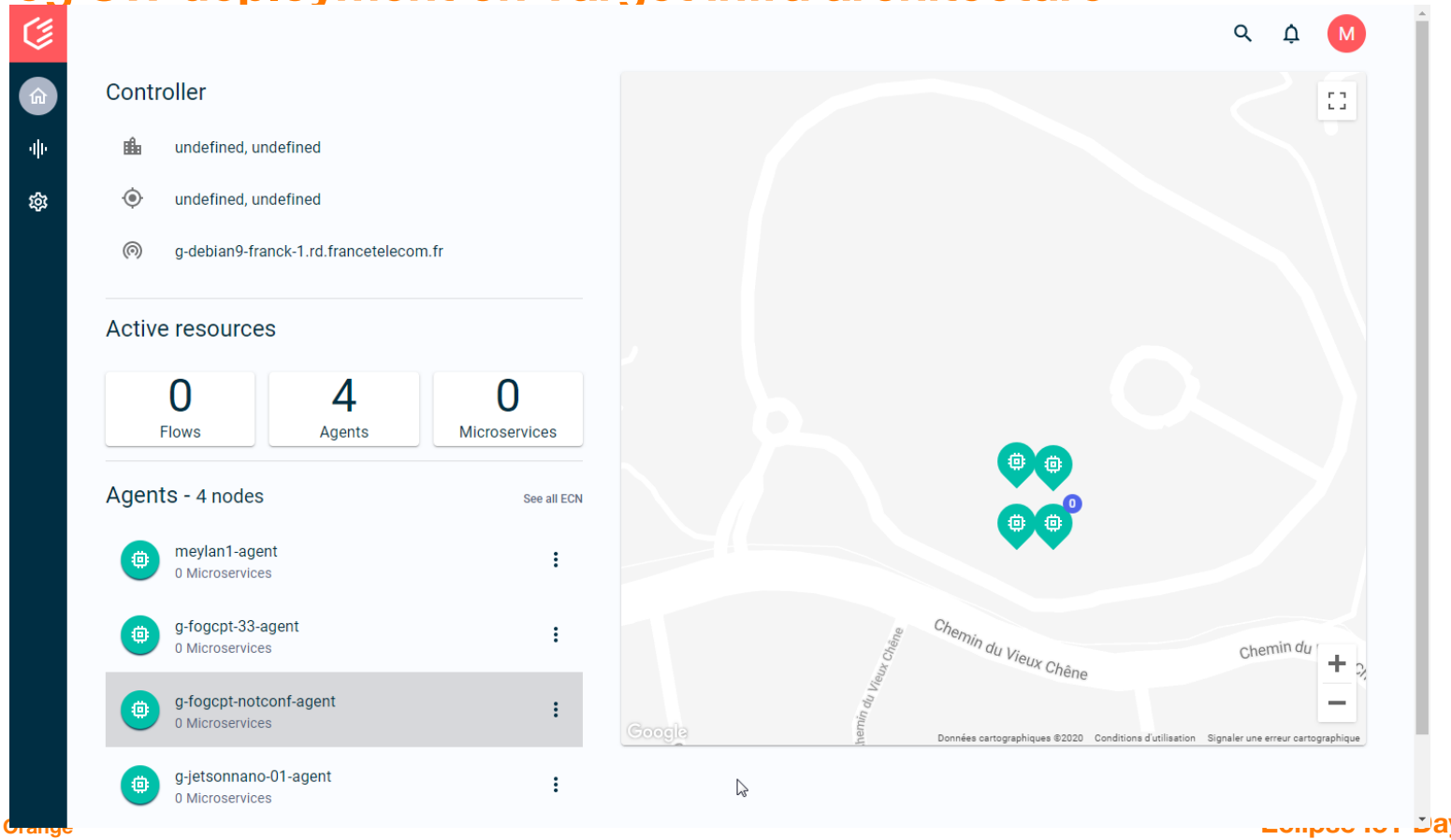

ioFog Agent
Jetson nano


ioFog Agent
Raspi 3

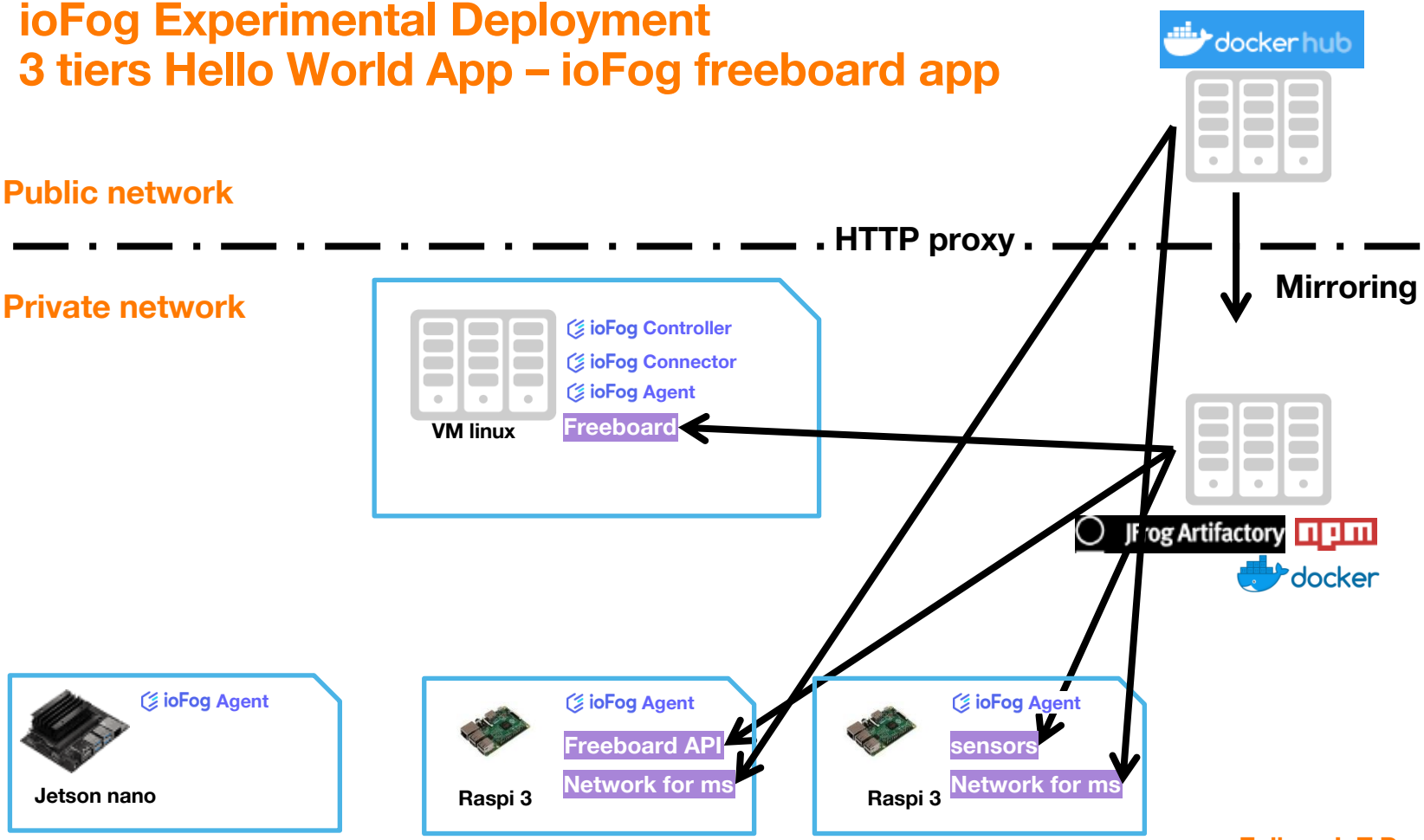

ioFog Agent
Raspi 3

Orange

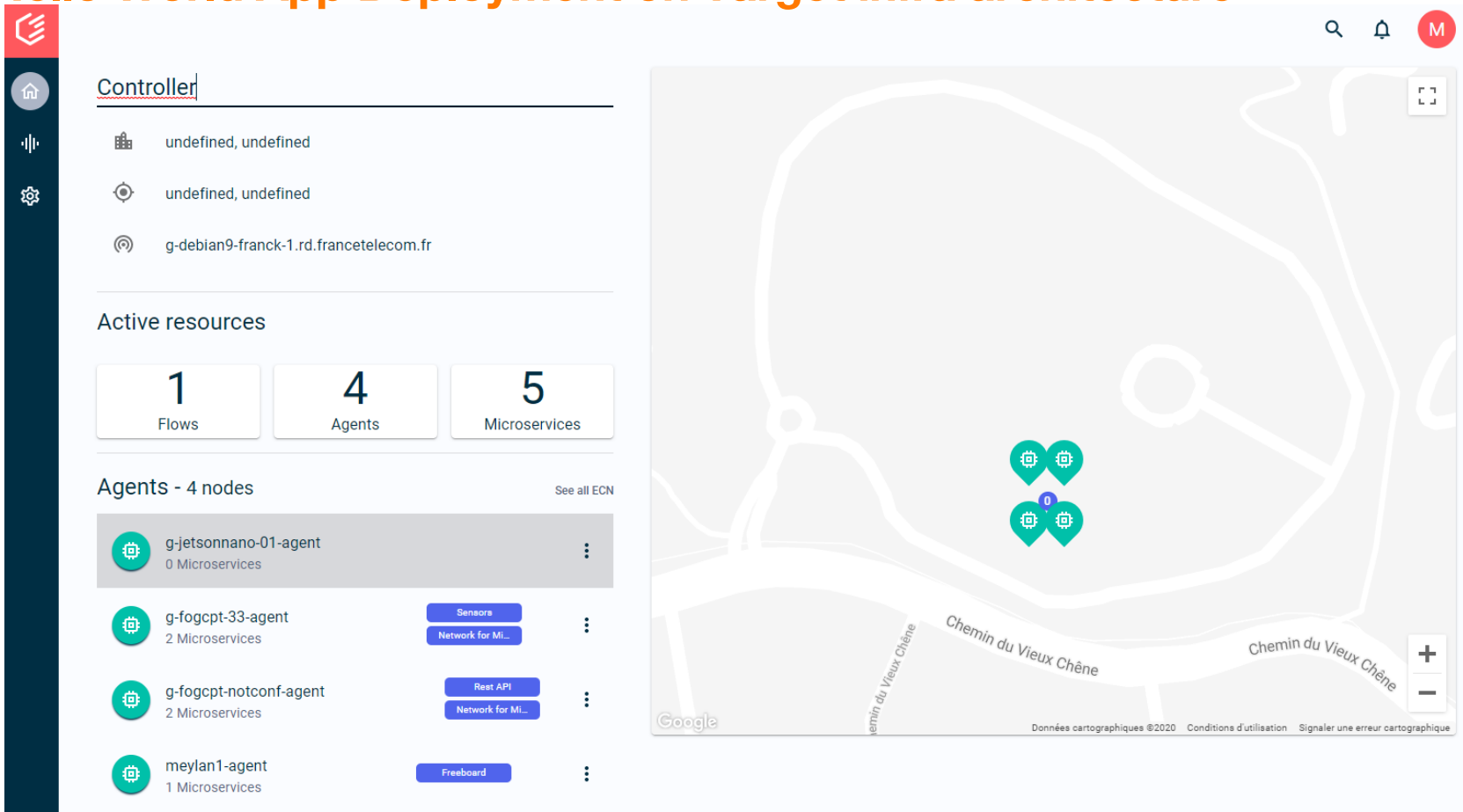
ioFog Experimental Deployment
ioFog SW deployment on Target infra architecture



ioFog Experimental Deployment
3 tiers Hello World App – ioFog freeboard app



ioFog Experimental Deployment
Hello World App Deployment on Target infra architecture



Install ioFog command line tools

- mkdir iofog
- curl https://packagecloud.io/install/repositories/iofog/iofogctl/script.deb.sh | sudo bash
- sudo apt-get install iofogctl=1.3.0

Can also get ioFog Demo resource:

- wget http://www.edgeworx.io/downloads/demo-sdk/edgeworx-iofog-sdk_1.3.0-beta.tar.gz
- tar xvfz edgeworx-iofog-sdk_1.3.0-beta.tar.gz
- ./bootstrap.sh

Check the version of iofogctl

- iofogctl version

latest release(2019.Nov.03): 1.3.0

```
cloud@robot-service-machine:~/iofog$ iofogctl version
iofogctl Unified Cli

Copyright (C) 2019, Edgeworx, Inc.

version: 1.3.0
platform: linux/amd64
commit: e8a2486d90a9291d801825ad4d334aed1a5dd54f
date: 2019-11-20T22:14:36+0000
```

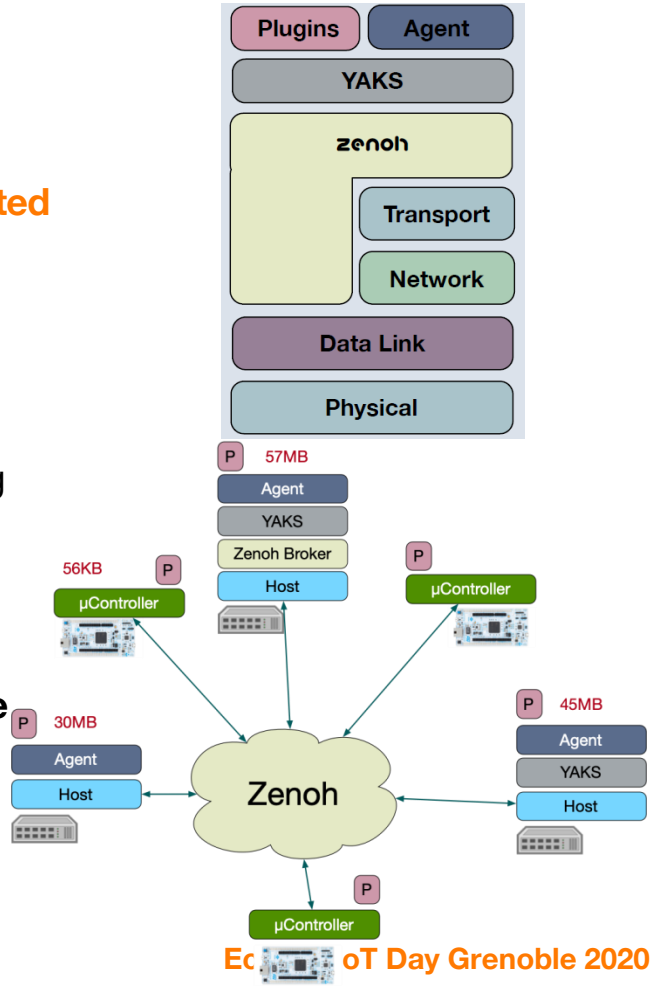
Eclipse fog05

Project Overview

- **“Eclipse fog05 aims at providing full Management and Orchestration stack for the Fog Computing environment”**
- **“The End-to-End Compute, Storage and Networking Virtualisation solution.”**
- **One of the main goals of fog05 is the unification of different frameworks coming from Telcos and Industry, in particular ETSI MEC, ETSI NFV and the OpenFog Consortium (merged in one stream of Industrial Internet Consortium now), there three framework deals with applications, but from different perspectives and focusing on different aspects of the application.**
  - not clear if fog05 will keep following the spec after OFC merged with IIC
- **Source code repositories: https://github.com/eclipse-fog05**
- **Licenses: Eclipse Public License 2.0 or Apache 2.0**
  - Note：Yaks and Zenoh have Apache 2.0 license.
- **Source code language: OCaml, Python, Go**
- **Project leading contributors: ADLink**

Fog05 Architecture

- **fog05 features a decentralized and plug-in-based architecture. It's designed to run as either a process on a traditional OS or as a trusted service inside a Trusted Execution Environment (TEE).**
- **fog05 is composed by:**
  - Zenoh: zero network overhead protocol - a protocol for extremely resource constrained environment (XRCE), Zero Overhead Pub/sub, Store/Query and Compute
  - Agent: The core logic of fog05, it takes care of managing, monitoring and orchestrating entities through plugins
  - Plugins: Plugins provide supports for atomic entities, OS, networks, etc.
- **The agent represents a manageable resource**
  agent has two stores, one representing the actual state of the node and the other representing the desired state for the node.

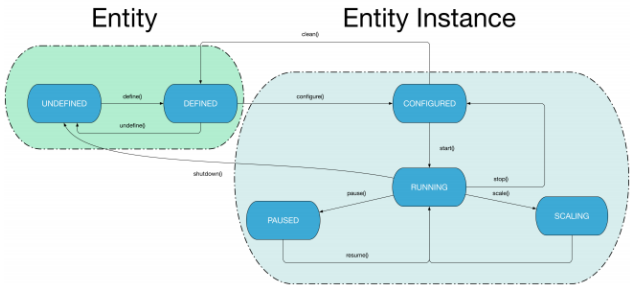


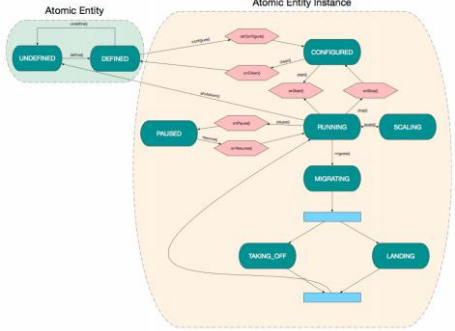
Fog05 Plugins

- **OS plugins: abstract the underlying operating system and provide primitives to *agent* and other plugins.**
  - Linux plugin
  - Windows plugin
- **Network manager plugin: create all the network related components (*network*, *routers* and *ports)***
  - Linux bridge with VXLAN virtual networks
- **Runtime plugin: life-cycle management for Fog Deployment Unit *(FDU)***
  - KVM virtual machine
  - Linux containers (LXD)
  - Native applications
  - Containerd (docker) containers (in roadmap)
  - ROS2 robotic framework (in roadmap)
- **Device plugin: designed to allow the deployment and discovery of micro-controller and IoT board that does not have an operating system**
  - an example implementation for a micro-controller that can be used as base for other implementation (in roadmap)

Fog05 Core Concepts: FDU, Atomic Entity & Entity

- **A Fog Deployment Unit (FDU) is an indivisible unit of deployment, such as, a binary executable, a uni-kernel, a container or a virtual machine. An FDU requires a certain kinds of resources as a precondition to its execution. The life-cycle of an FDU is defined by a Finite State Machine (FSM).**
- **An Atomic Entity is a functional unit of deployment, meaning that it is composed by one or more FDU, that can be of the same type or different types.**
- **An Entity is a service/application deployment and it is composed by one or more Atomic Entities.**
- **Unit manifest**
  - fog05 entities are described through JSON manifests.
  - These manifests are compatible with ESTI MEC/NFV manifests as well with OpenMANO.
- **Entity & Atomic entity lifecycle (6 states for entity and 9 for atomic entity)**





Fog05 Experimental Deployment

Deployment environment:

- **2 virtual machines on top of openstack: 4GB, 1vCPU, 40GB**
- **ubuntu 16.04, python 3.6, libev4, libssl1.0 , openssl…**

Deployment logic:

- **deploy 2 Fog05 nodes:**
  - **Yaks server, Eclipse fog05 agent, Linux plugin, Linux bridge plugin and LXD plugin**
- **deploy example application**

Reference

https://github.com/eclipse-fog05/fog05/blob/master/INSTALL.md

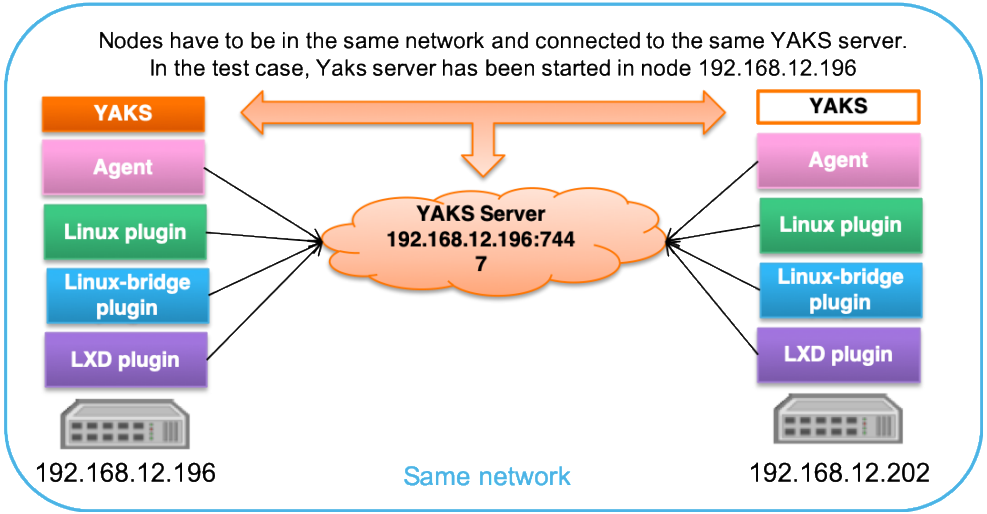
Issues during the experimental deployment has been raised in the github for supporting

https://github.com/eclipse/fog05/issues

Fog05 Experimental Deployment

Experimental deployment architecture:

- **Node1: { IP: 192.168.12.196 ; LXDbridge: 196lxdbr0 ; NodeID: f6717a4f-65b8-4f11-a822-96a13764cc6e }**
- **Node2: { IP: 192.168.12. 202 ; LXDbridge: 202lxdbr0; NodeID: dccd6c65-b371-4bef-a631-eda7853a1ca9}**
- **Example FDU UUID: d790a23d-ee56-5645-8c8d-fb014d9c4776**
- **YAKS server: 192.168.12.196**

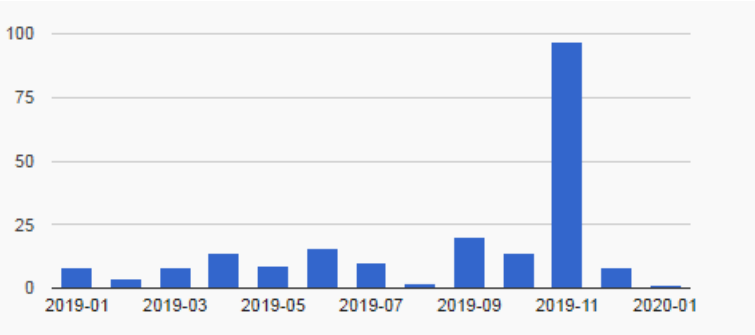


Outline

- Edge Computing in the Eclipse Foundation
- ioFog Introduction
- fog05 Introduction
- Comparisons
- Summary & Next steps

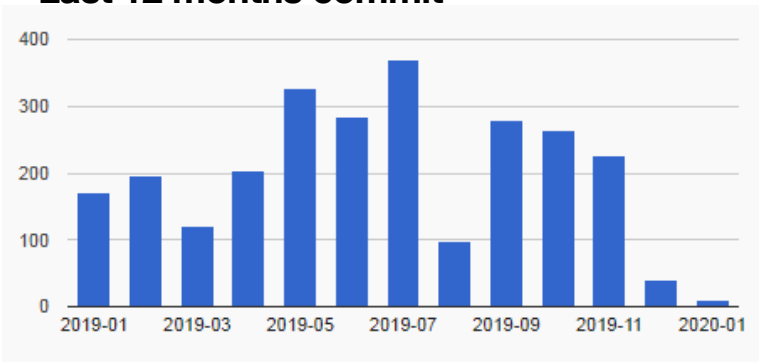
Project Active level

- **Github: stars: 17, forks: 15**
  (01/17/2020)
- **Release:**
  - v0.1 Feb 14 2020
  - (v0.1.3) in Oct.2018
- **Last 12 months commit**



https://github.com/eclipse/fog05

- **Github: stars: 249, forks: 30 (ioFog Agent)**
  (01/17/2020)
- **Release:**
  - **v1.3.0, Oct.21 2019**
  - **V1.2.0, Oct.07 2019**
  - **V1.1.0, Jun.19 2019**
- **Last 12 months commit**



https://github.com/eclipse-iofog

Community & Ecosystem

- **Foundation: Eclipse Foundation**
- **Parent project: Eclipse IoT**
- **Project leaders: ADLINK**
- **Related parties:**
  - OpenFog Consortium The Open Fog Consortium (merged with IIC) is currently considering to adopt fog05 as the reference implementation of the reference architecture and as one of the fog platforms to use in fog testbeds.
  - ITRI: ITRI has adopted fog05 for all fog computing, MEC and 5G
  - UC3M: The networking and edge computing group at the Universidad Carlos III de Madrid, leader by professors Arturo Azcorra and Antonio Oliva, are actively collaborating with us and contributing to fog05.
  - 5G Coral UE Project The 5G Coral EU project has adopted fog05 as the infrastructure for MEC and its actively contributing to its R&D.
  - Huawei: fog05 along and some of the services provided such as its decentralized key/value store are of extreme interest to Huawei Corporate R&D. Huawei has been actively discussing requirements and directions of these technologies.

- **Foundation: Eclipse Foundation**
- **Parent project: Eclipse IoT**
- **Project leaders: Edgeworx**
- **Related parties:**
  - Edgeworx launched out of stealth October 30, 2018 , announcing funding from Samsung NEXT, Sequoia Seed and CloudScale Capital Partners.
  - Collaboration with other Eclipse IoT projects:
    - hawkBit: by leveraging microservices running on the ioFog, remote updates can be achieved for devices that have complex update processes or connectivity that requires intelligent interfacing.
    - Vorto: the meta definitions of real-world objects can be represented in the authoring interface, with the implementation happening on a per-object basis through the use of microservices running on the edge near the actual physical object.
    - Kura: the ioFog (along with ComSat) can be used to add remote connectivity and serviceability for Kura gateways that remain safely behind a firewall or NATed network. Microservices can be instantiated on the ioFog to work with Kura gateways and add layers of behavior for large groups of them.

Licensing Info

- **Licenses** Eclipse Public License 2.0 or or Apache 2.0
- **fog05 has also dependencies on other Open Source projects. Some of those are not yet hosted by eclipse but plan to contribute as fog05 sub-projects. These projects may have a relative licensing impact:**
  - Eclipse Cyclone
  - Eclipse Cyclone Python API http://github.com/atolab/python-cdds/
  - Eclipse Cyclone OCaml API http://github.com/atolab/ocaml-cdds
  - DStore http://github.com/atolab/python-dstore

- **Licenses:** Eclipse Public License 2.0
- **ioFog has also included 3'rd party libraries:**
  - Netty (Apache 2 license)
  - HornetQ (Apache 2 license)
  - Docker-Java (Apache 2 license)

Supported Deployed Environments

fog05

- **General requirements:**
  - Eclipse fog05 is designed to be deployed from big servers to micro-controllers.
  - Eclipse fog05 can run in different systems, including windows, linux or uni-kernel with different OS Plugin
- **Minimum requirements:**

For normal node	• Processor: x86, x86-64, ARM, ARM64 CPU • RAM: 14M • Hard Disk: 20MB • Linux kernel 3+
For distributed virtual switch	• Processor: x86, x86-64, ARM, ARM64 CPU • RAM: 24M • Hard Disk: 20MB • Linux kernel 3+

ioFog

- **General requirements:**
  - Linux 3.10+ (ubuntu, centos, etc), macos 1.12+, windows 7+
  - Docker 1.10+
- **Minimum requirements of main components:**

Agent	• Processor: x86-64 or ARM Dual Core or better • RAM: 256 MB minimum • Hard Disk: 100 MB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc) • Java Runtime v8.0.0 or higher • Docker v1.10 or higher
Controller	• Processor: x86-64 or ARM Dual Core or better • RAM: 1 GB minimum • Hard Disk: 5 GB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc), macOS 10.12+, or Windows 7+ • Node.js v8+ and NPM
Connector	• Processor: x86-64 or ARM Dual Core or better • RAM: 1 GB minimum • Hard Disk: 5 GB minimum • Linux kernel v3.10+ (Ubuntu, CentOS, Raspbian, etc) • Java Runtime v8.0.0 or higher

Development Tools

- **Tech doc**
  - Fog05 Wiki
  - API documents (FIMAPI modules)
  - SDK documents
- **API & SDK**
  - Language: OCaml, Python3, GO
  - REST API are still under development
- **Command line tools**
  - “fos” command line tool (limited functions so far)

- **Tech doc**
  - online docs (multiple versions)
  - API references
  - SDK document
- **API & SDK**
  - Controller REST API Reference: /swagger/1.3.0/controller/swagger.yaml
  - Connector exposes API and it’s API where you have a set of identities.
  - The Agent daemon supports a local API with REST-like endpoints as well as WebSockets.
  - The ioFog SDK is an optional library. Together with Connector, they provide an easy way for edge nodes to communicate with each other: C#, C/C++, Go, Java, JavaScript (Node.js), Python
- **Command line tools**
  - Command line tool: iofogctl (works as control plane of ECN)

Compliance & Interoperability

- **Compliance with MEC**
  - fog05 entities are described through JSON manifests, and these manifest are compatible with ETSI MEC / NFV
- **Compliance ETSI OSM**
  - fog05 is used as VIM for Extreme-Edge devices (lamppost) enabling MEC and LXD container instantiation through ETSI OSM (5Gppp 5GCity).
- **fog05 is one of the infrastructure identified as compliant with the 5G principles and requirements by the EU 5GPPP working group**

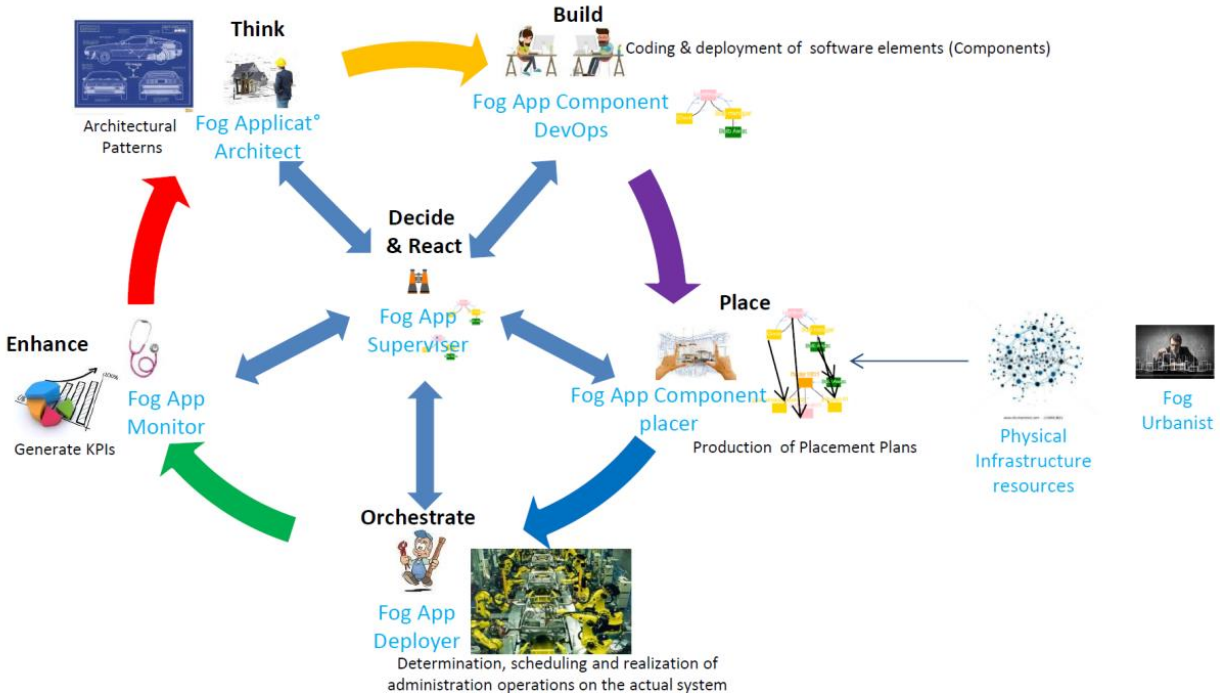
- **Compliance with Kubernetes**
  - IoFog engine has been integrated with Kubernetes, enabling Kubernetes to orchestrate microservices down to the edge.

Verification test

- **If it's easy to test:**
  - During installation and verification, several bugs and issues have been met which can't be easily solved rely only its available official doc.
  - demo examples has been provided: https://github.com/atolab/fog05_demo
- **Any issues:**
  - Some issues' link on Fog05 GitHub:
    https://github.com/eclipse/fog05/issues/143
    https://github.com/eclipse/fog05/issues/146
  - REST API are still under development
  - tech docs' information are not always up-to-date
  - Lack of full guidance on installation and deployment
  - Command line tool is not full-functioning
- **Ways to get support:**
  - on Eclipse fog05 repository: https://github.com/eclipse/fog05/issues
  - on its gitter channel: https://gitter.im/atolab/fog05
  - by e-mail

- **If it's easy to test:**
  - Creating and managing ECN (Edge Compute Networks) on a local machine by following the guide haven't met too many issues.
  - API reference and SDK description are clear and detailed.
  - Command line tools works well for testing.
  - Hard to setup behind entreprise proxy
- **Any issues:**
  - Some questions have been raised mainly related with the ioFog usage.
- **Ways to get support**
  - iofog discussion forum: https://discuss.iofog.org/
  - To report security issues: https://www.eclipse.org/security/
  - Join Slack to get updates and ask questions: iofog.slack.com
  - By email

Fog Functions for Service Life Cycle Management



Fog Infrastructure
- Node discovery
- Node resource abstraction
- Node configuration
- Node monitoring

Fog Application Deployment
- Fog orchestration
- Runtime engines

Fog Application Monitoring
- Application service discovery
- Application monitoring
- Analytics service
- Application data management

Security
- Security consideration

Fog Infrastructure Related Functions

Items	fog05	ioFog
Node resource discovery	- **computational resources**, such as, CPU, FPGA, GPU, **storage resources**, such as, RAM, block storage, object storage, **networking resources**, such as, 802.11 devices, tunnels and **I/O resources**, such as, COM, CAN, GPIO and I2C (some of them are not ready yet) - In order to expose its resources in the system each node has either a running fog05 agent or a device plugin.	- Hardware resource abstraction and monitoring - Realtime update on the node status e.g. memory and CPU usage - Location information is automatically obtained and recorded
Node discovery	- The agent is in charge of advertising of node resources and functionalities to the whole system.	- ioFog agent should connect to controller for reporting node status
Node configuration	- APIs capabilities of updating desired status of a node entity	- API exposed for node configuration - Control plane command line tools
Node monitoring	- Node agent provide the entry point for the node management and expose the capabilities with APIs	- Node status monitoring - Control plane command line tools - Web portal for node visualization

- **Agent is a common solution to do the node resource exposure, remote configuration, and etc.**
- **The abstraction in iofog includes more low level network resources such as network bridge, OS interfaces.**
- **iofog contains the dynamic and real-time information of node status such as the CPU usage, RAM usage, last active timestamp etc.**
- **iofog provides more ways to monitor and visualize the infrastructure include querying API/SDK, using command line tools or viewing from web portal.**

Fog Deployment Related Functions

Items	fog05	ioFog
Fog orchestration	- Framework and basic abstractions are defined in Fog05 for supporting service orchestration. - This component is still in the design phase,	- Orchestration functions are ivolving toward native Kubernetes
Runtime engines	- fog05 can manage open-ended set of hypervisors and container technologies for which a Runtime Plugin was implemented. - KVM virtual machine - Linux containers (LXD) - Native application (Linux and Windows) - Planning: containerd, ROS2 Robotic framework	- Docker engine - Linux containers (lxc) - more lightweight runtime engine is planned in the roadmap

- **fog05 is now still in the design phase for main orchestration feature, which is planning to deploy entities in the suitable place with respect its constrains, I/O latency, geofencing and computing power.**
- **iofog has involving towards the native Kubernetes orchestration.**
- **fog05 support more types of runtime engines.**
- **Both of the projects claim that more lightweight runtime engines are planned in the roadmap.**

Fog Application Monitoring Related Functions

Items	fog05	ioFog
Application service discovery	- Service could be deployed as FDU - Global view of a FDU would be shared by YAKS server	- Microservices could be deployed in different edge clusters and discovered by control plane
Application data management	- support application status monitoring - No additional storage for temporary data	- support management by microservice - No additional data for temporary data
Application monitoring	- API capabilities for application status description, on-board and offload	- Control plane command line tool for service description - API capabilities for application status description - Visualization in web portal with mapping to hosted agent (ECN viewer)
Analytic services	- No, could be implemented by FDU	- No, could be implemented by micro-services

- **Both the two projects support fundamental functions on application/service discovery or monitoring either by distributed servers or through a centralized server.**
- **Both the two projects do not support service data management and analytics which can be regarded as advanced features of those type of platforms.**

Fog Security Related Functions

Items	fog05	ioFog
Security management	- Token based security measures - Security plugins in distributed storage model YAKS in fog05, including access control, authentication and crypto	Each node is constantly validating a comprehensive set of security rules with all the other nodes, looking for minor deviations or signals of rogue nodes. - Secure delivery of short-lived secrets - full control over data flow and policy-based geofencing

- **Both the projects have basic security measures on authorization, data protection during transmission, as well as policy support in different sub models.**
- **iofog have policy control on the geofencing since node and application geolocations are identified in the platform.**

And the App components devlopment

Items	fog05	ioFog
App shop	- Need to manage a FDU repository	• Docker hub • Private inhouse docker repository
Prerequisite infra	- ZENOH and YAKS infra and network must exists	• Debian OS family. Ssh key exchange, user with sudoers privilege (without password) • Auto deployement (Hard with proxy)
Component packaging	- Docker - uniKernel - Everthing but you have to manage the self deployment, starting , stopping , removing your component	• Docker image only
Messaging model	- YAKS – Distributed key/ value store	• ioMessage

Outline

- Edge Computing in the Eclipse Foundation
- ioFog Introduction
- Fog05 Introduction
- Comparisons
- Summary

Summary

- Both fog05 & ioFog address the key elements from cloud to edge, including cloud and edge infrastructure management, dynamic service deployment, communication between cloud and edge, and etc.
- ioFog and Fog05 use different approaches:
  - Design of fog05 is try to create a unified compute fabric with considering compatibilities of industry standards and resource constrained environments
    - *If the development of this project could follow the industry standard evolution should be considered*
  - ioFog focuses container approaches and embraces the container-based ecosystems to create a distributed edge compute network (ECN) for running and deploying services
    - *Advanced features already enabled in existing orchestration solutions such as k8s may be still lack of support in ioFog currently*
- So far, from verification test experiences, ioFog performs better than fog05 in terms of code maintenance, documentation quality and project active status.

Thanks

Feb. 2019

Fog05 Plugins

	Linux plugin	Bridge utils plugin	LXD plugin	KVM Libvirt plugin	Native applications plugin
description	This plugin allow fog05 to run on top on Linux	This plugin allow fog05 to manage networks with bridge utils	This plugin allow fog05 to manage lxd container	This plugin allow fog05 to manage vm	This plugin allow fog05 to manage native applications
supported operation	execute command check that file exists save on file read from file get HW information (cp,ram,network,disks) get uuid from motherboard get hostname send signal check if a pid exists install packages remove packages	create virtual bridge create virtual network add interface to network delete virtual interface delete virtual bridge delete virtual network		deploy destroy stop pause resume	deploy destroy {{ pid_file }} parameter in starting native applications
Todo	get detailed i/o informations get HW accelerators informations get pid from process name get monitoring information about network get gps information about node	create virtual interface remove interface from network	scale of vm deploy migrate destroy stop pause resume	migrate scale of vm	configure application with parameters

Fog05 Communication

- fog05 uses YAKS to maintain the actual and desired state for global and node-specific information
- Note: In this version the nodes have to be in the same network and connected to the same YAKS server