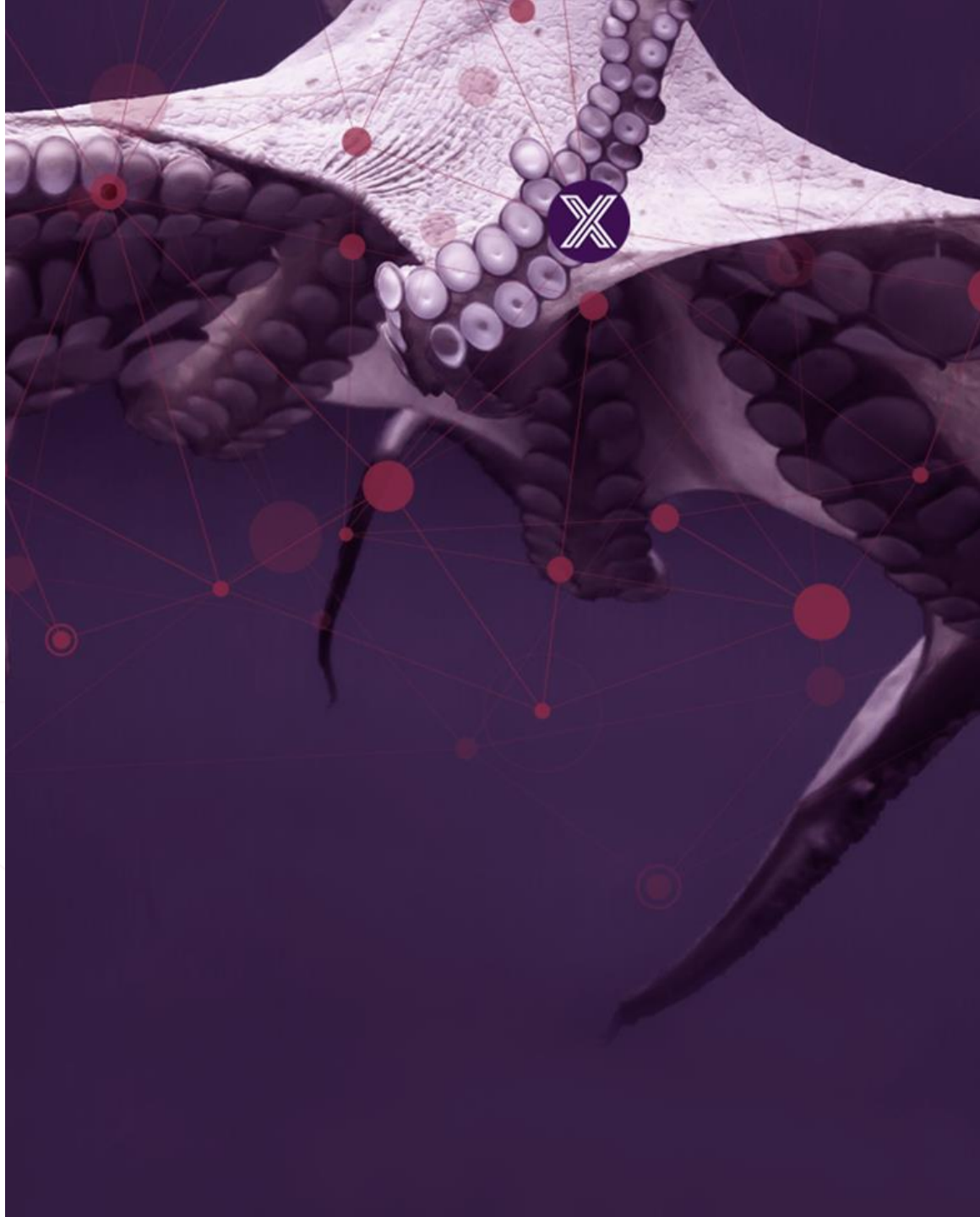
EDGE X FOUNDRY™

2018 Tech Talk Series

User's Guide to Getting Started with EdgeX Foundry



Where are you attending from today?

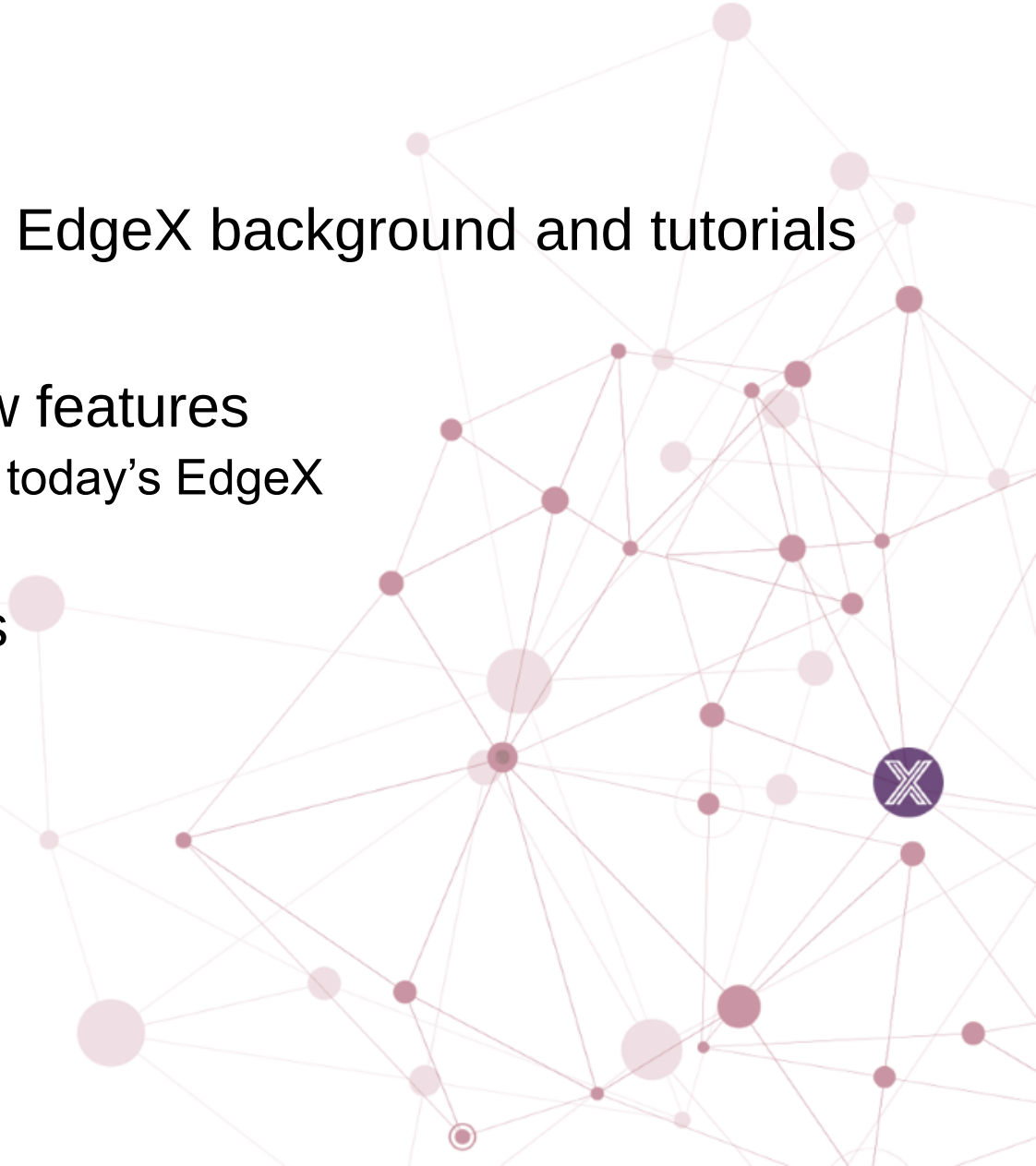
About Me

- Jim White (james_white2@dell.com)
  - Dell Technologies IoT Solutions Division – Distinguished Engineer
  - Team Lead of the IoT Platform Development Team
  - Chief architect and lead developer of Project Fuse
    - Dell's original IoT platform project that became EdgeX Foundry
    - Yes – I wrote the first line(s) of code for EdgeX (apologies in advance)
  - EdgeX Foundry …
    - Vice Chairman, Technical Steering Committee
    - Systems Management Working Group Chair
    - Ad hoc and unofficial lead architect

DELL Technologies

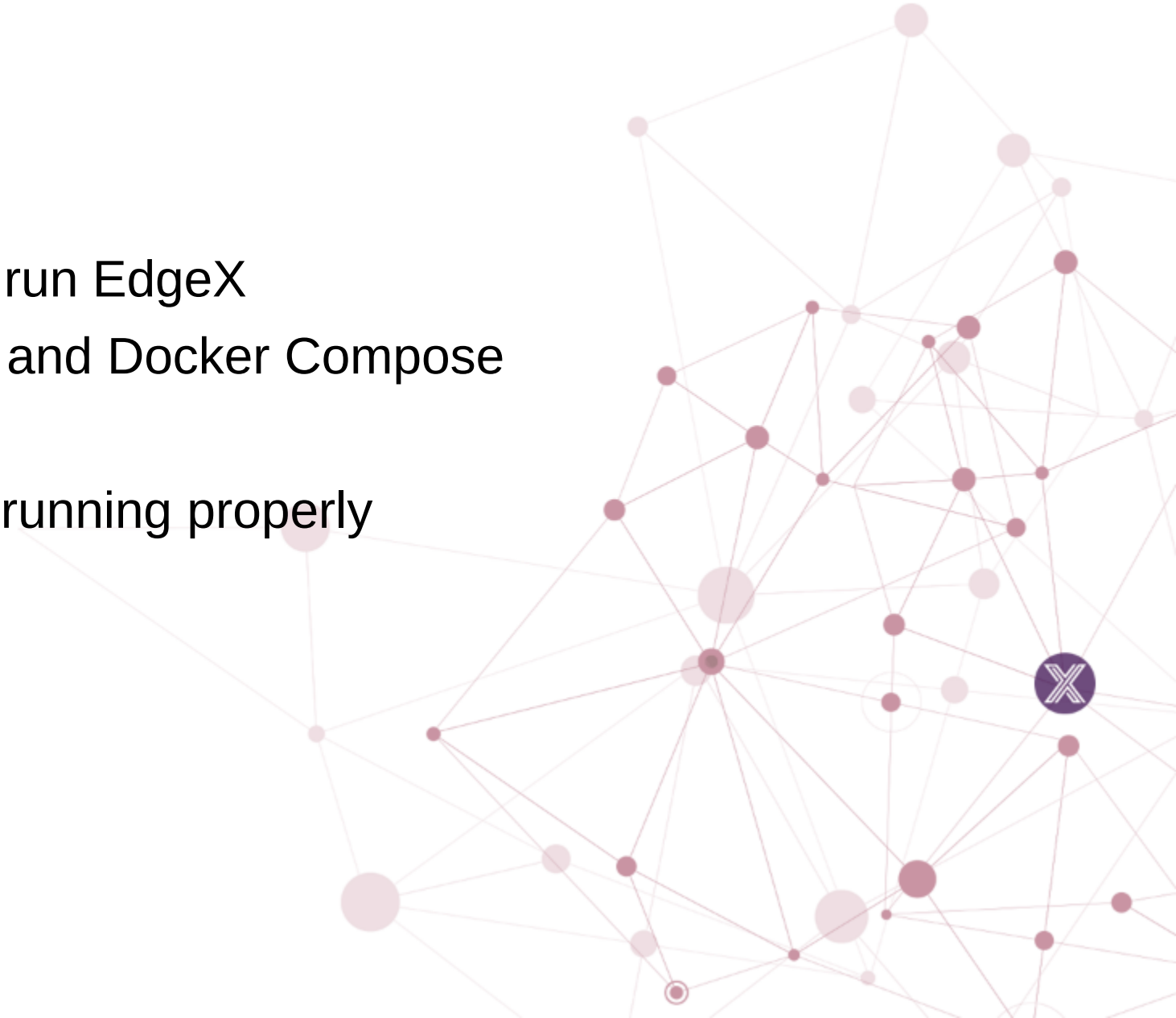
EdgeX Tech Talks

- Goal: provide new community members with EdgeX background and tutorials
- Started in 2017
- Updating to address new code base and new features
  - Be careful when viewing an old Tech Talk against today's EdgeX
  - Some information may be out of date
- Additional Topics – please email suggestions
  - More at the end of this talk



Talk Agenda

- Understand what EdgeX is
- Examine what tools you need to run EdgeX
- Explore fundamentals of Docker and Docker Compose
- Learn how to get and run EdgeX
- See how to check that EdgeX is running properly
- Future Topic Requests
- Q&A



Why is IoT hard to do?

- Heterogeneity of platforms
  - Diverse collection of OS and OS variants
    - Linux, Unix, Windows, VxWorks, embedded and RTOS, …
  - Various Hardware (Intel, AMD, ARM,…)
  - Cloud, gateway, smart thing (the “Fog continuum”)
- Thing protocol soup
  - Industrial: BACNet, Modbus, OPC-UA,…
  - Wireless: BLE, Z-Wave, Zigbee,…
  - Message: MQTT, AMQP, DDS, …
- Variety of cloud platforms
  - Azure IoT Hub, AWS IoT Platform, Google IoT Core, IBM Watson IoT Platform, …
- Add your favorite selection of…
  - Applications, edge analytics/intelligence, security, system management, …
- Difficulties in determining where to start

IoT is a post doctorate in all we know and have done in computing for the last 30-40 years

- Networks/protocols
- Mobile computing
- Distributed compute
- Cloud compute
- AI/Machine learning
- …

Introducing EdgeX Foundry

An open source, vendor neutral project (and ecosystem)

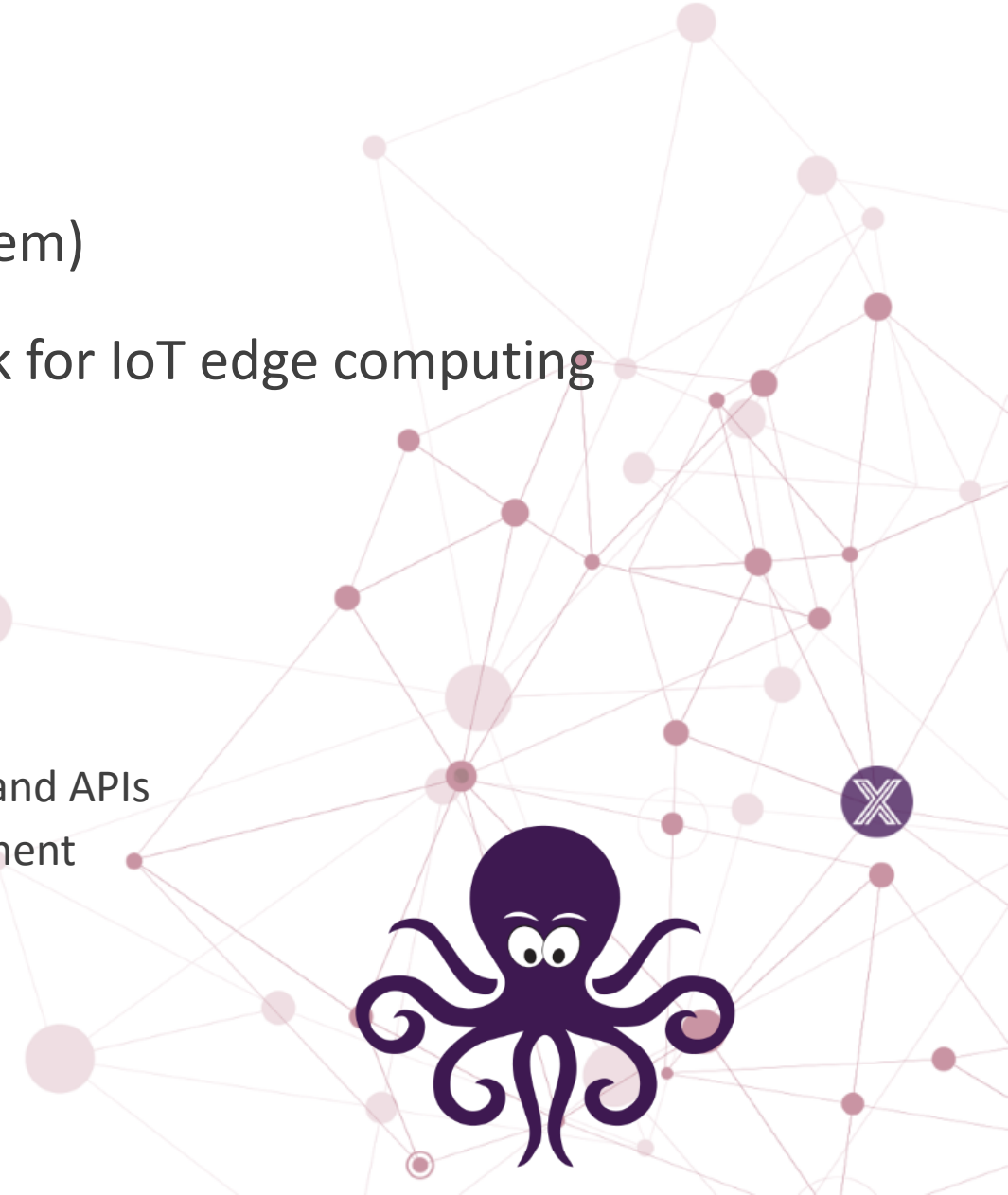
A **micro service**, loosely coupled software framework for IoT edge computing

Hardware and OS agnostic

Linux Foundation, Apache 2 project

Goal: enable and encourage growth in IoT solutions

- The community builds and maintains common building blocks and APIs
- Plenty of room for adding value and getting a return on investment
- Allowing best-of-breed solutions



EdgeX Foundry Goals

- Build and promote EdgeX as the common open platform unifying edge computing
- Enable and encourage the rapidly growing community of IoT solutions providers to create an ecosystem of interoperable plug-and-play components
- Certify EdgeX components to ensure interoperability and compatibility
- Provide tools to quickly create EdgeX-based IoT edge solutions
- Collaborate with relevant open source projects, standards groups, and industry alliances to ensure consistency and interoperability across the IoT

A Brief EdgeX History

- Chartered by Dell IoT marketing in July 2015
  - A Dell Client CTO incubation project (Project Fuse)
- Designed to meet interoperable and connectivity concerns at the IoT edge
- Started with over 125,000 lines of Dell code
- Entered into open source through the Linux Foundation on April 24, 2017
  - Started with nearly 50 founding member organizations; today we have more than 75
- Release Cadence: 2 formal releases a year

'Barcelona' Release
(Released Oct 20 2017)

'California' Release
(June 2018)

'Delhi' Release
(Oct 2018)

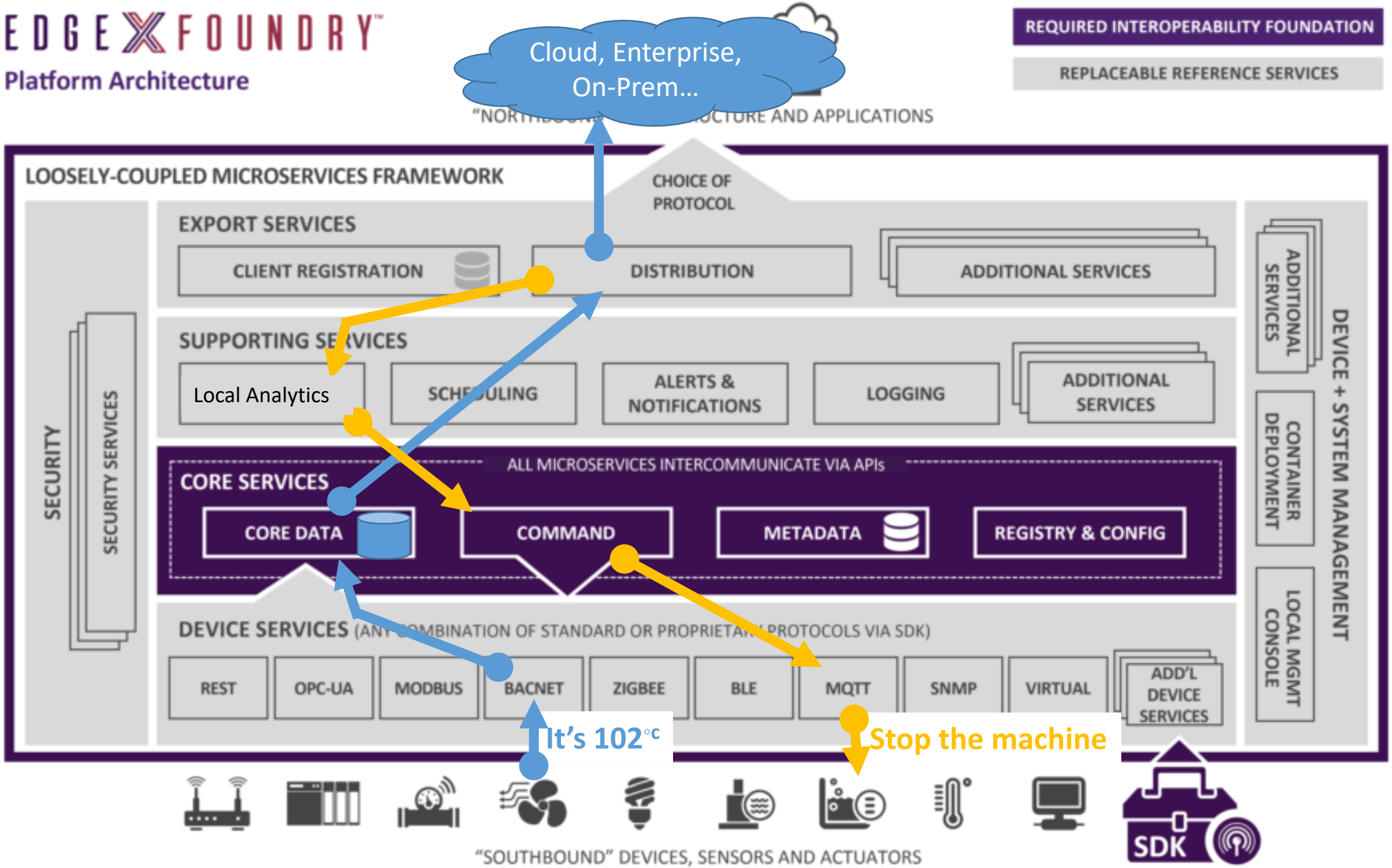
'Edinburgh' Release
(Apr 2019)

EdgeX Primer - How it works

- A collection of a dozen+ micro services
  - Written in multiple languages (Java, Go, C, … we are polyglot believers!!)
- EdgeX data flow:
  - Sensor data is collected by a **Device Service** from a thing
  - Data is passed to the **Core Services** for local persistence
  - Data is then passed to **Export Services** for transformation, formatting, filtering and can then be sent “north” to enterprise/cloud systems
  - Data is then available for edge analysis and can trigger device actuation through Command service
  - Many others services provide the supporting capability that drives this flow
- REST communications between the service
  - Some services exchange data via message bus (core data to export services and rules engine)
- Micro services are deployed via Docker and Docker Compose

EDGE X FOUNDRY™

Platform Architecture

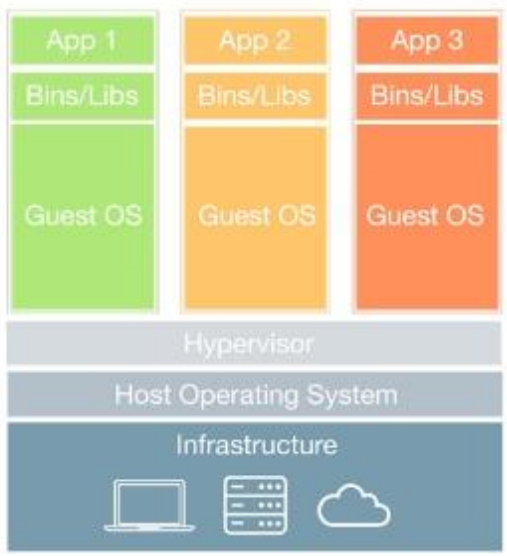


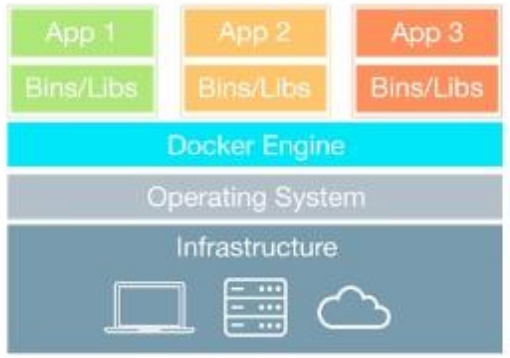
Options to Getting & Running EdgeX

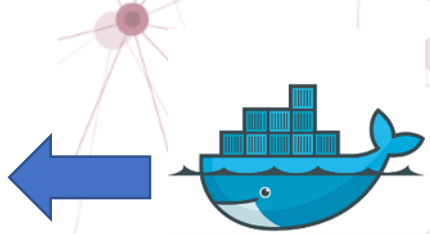
- EdgeX micro services can be built and deployed in a number of ways
  - “Contributors Approach”
    - Get the raw code, build it, and deploy the services to the target platform(s)
  - “Users Approach”
    - Get EdgeX Docker container images and deploy/run to a platform where Docker is installed
  - “Hybrid Approach”
    - Get, build and deploy some of the services on your own
    - Get and use Docker container images for the other services
  - Docker Compose is a tool to help get and run multiple containers
    - Docker Compose can be used with either the User or Hybrid approaches
- We are going to focus on the **User Approach** today

What is Docker?

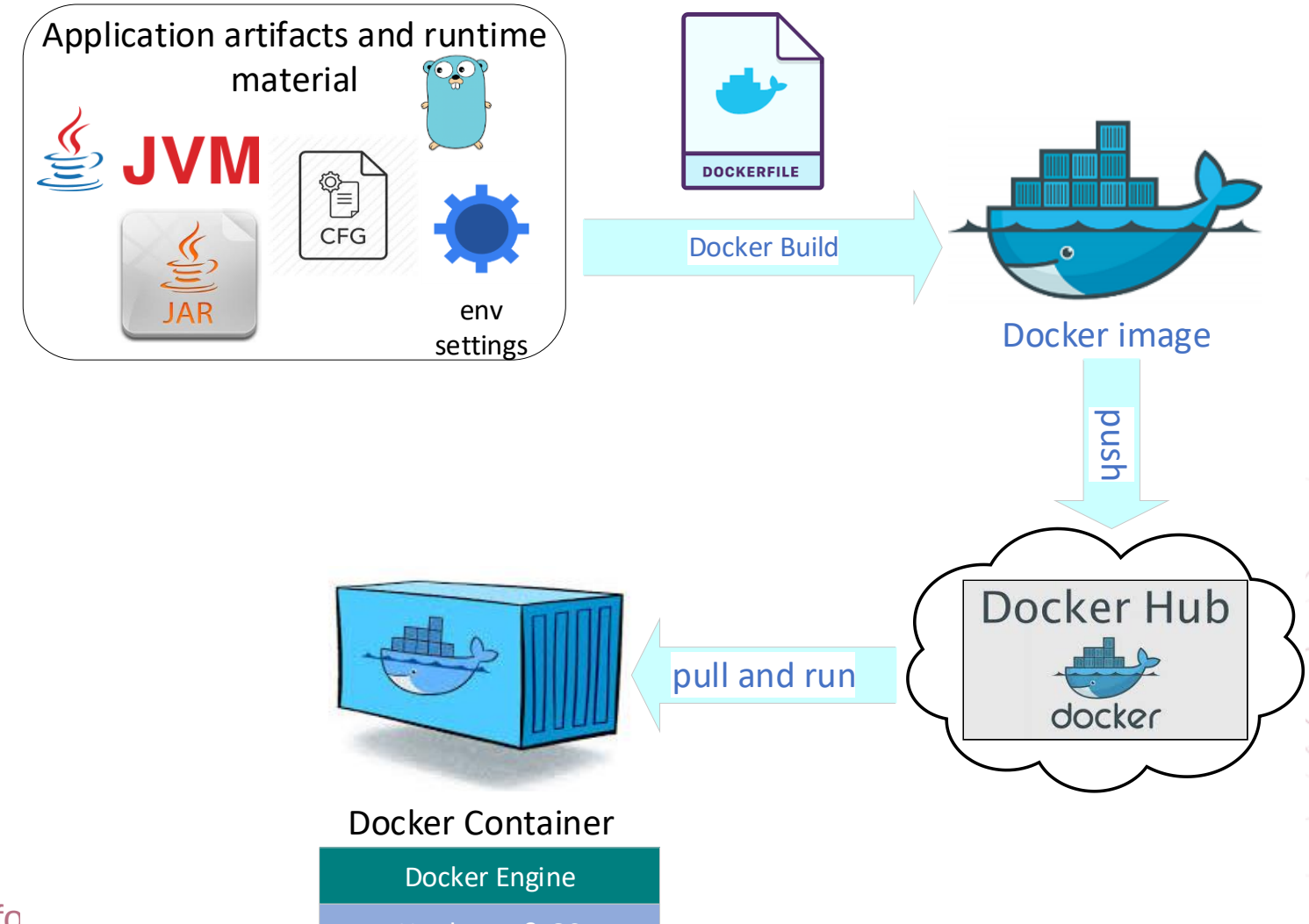
- Docker is a bit like virtualization
  … but allowing some elements (like OS) to be obtained from the underlying host
- Docker provides the ability to package and run an application in a loosely isolated environment called a ***container***
- Many containers can run simultaneously on a given host





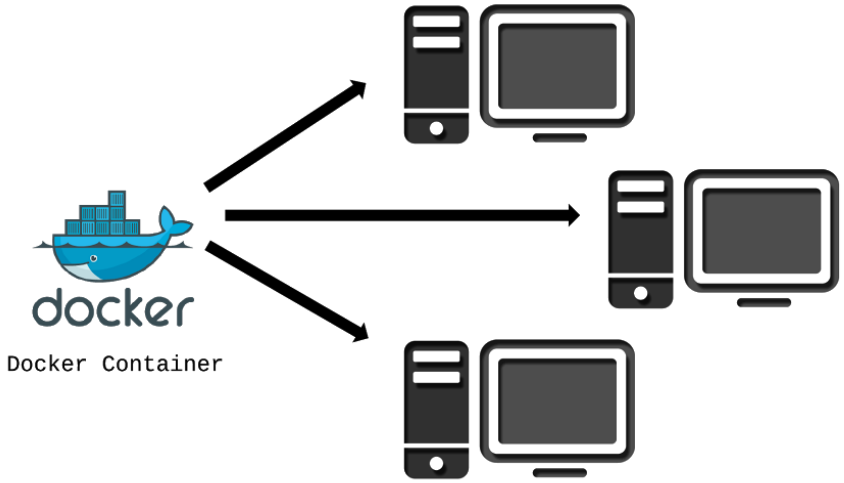


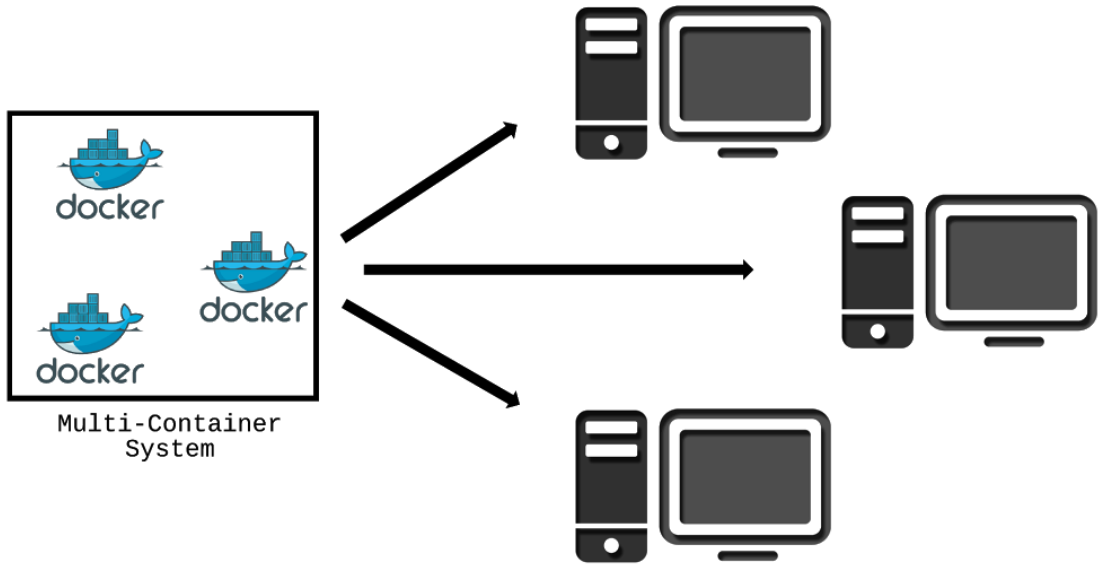
Docker Images, Containers, Engine



Docker & Docker Compose

- Docker allows the application to be package up and deployed with all of the parts it needs (libraries, other dependencies, etc.)
  - It provides a certain amount of OS independence
- Docker Compose is a tool for defining, running, and managing complex multi-container systems





EdgeX via Docker Tools

- To use EdgeX via Docker containers, you do not need any development tools
- You do need Docker (Community Edition is fine) for your particular OS
  - See https://docs.docker.com/engine/installation/
- You will find it very helpful to also have Docker Compose
  - Already comes with Docker for Windows or Docker for Mac
  - Other environments need to install separately
  - See https://docs.docker.com/compose/install/
- Going to assume you can get and setup Docker, Docker Compose, etc.
  - If you desperately need help, please contact me

User Approach to Get & Run

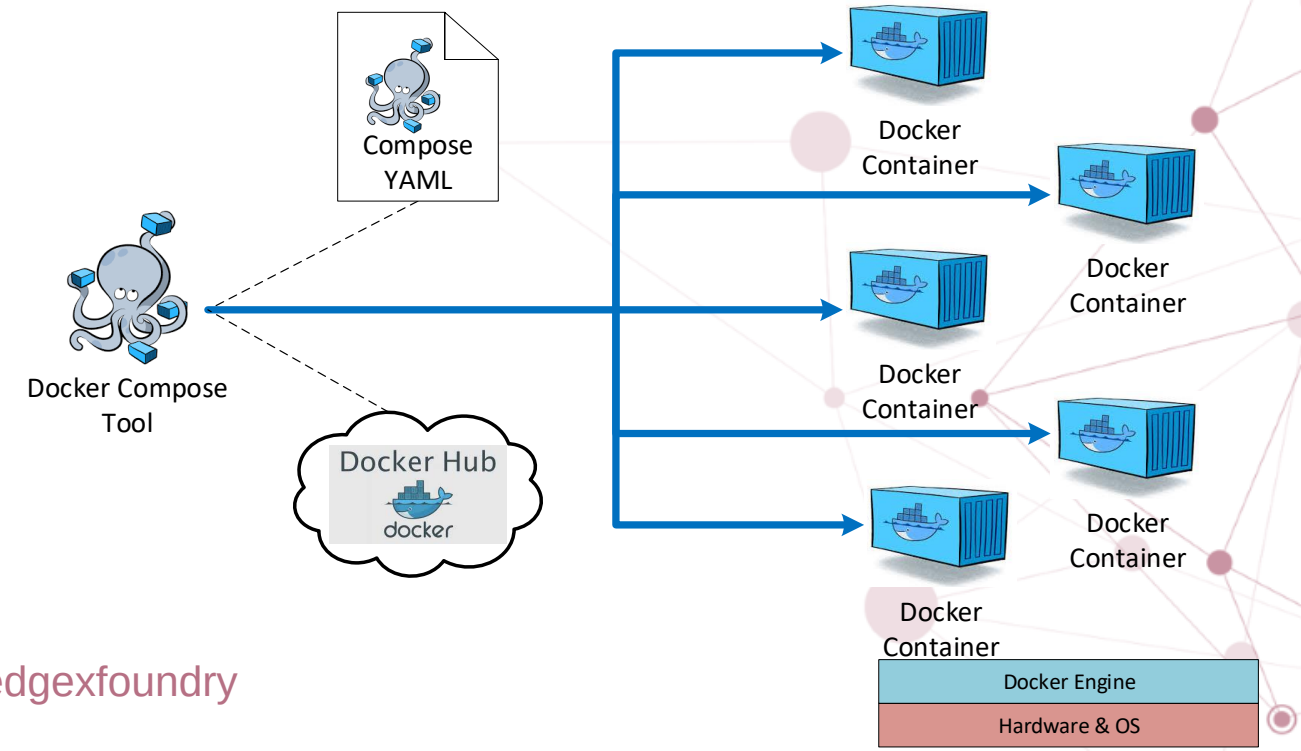
- The EdgeX community provides a Docker container image for each micro service (and underlying infrastructure such as the database)
  - This convenience allows users to quickly get pre-built EdgeX micro services
  - Because the container images have all the necessary environment (OS, configuration, etc.) for the micro services, it makes deploying EdgeX easier
  - The container images can be run on any platform that runs Docker
    - There are different container images for hardware platforms (Intel or Arm)
- The EdgeX Docker container images are available in Docker Hub (hub.docker.com)
  - The most recent code is always built to “developer” container images
  - These are made available from a Linux Foundation Nexus repository
  - These should only be used when you need the latest developer work

Docker Compose Simplifies Getting/Running EdgeX

- You could use Docker to manually pull and start each EdgeX container
  - Complicated and would require many commands to start and stop each container
  - Would also require setting up a Docker network, volume, etc.
- Simplified by Docker Compose
  - 1 command to pull all the EdgeX containers
  - 1 simplified command to start a container
- Requires getting the EdgeX docker-compose.yml file
  - Go to https://github.com/edgexfoundry/developer-scripts
  - Click on compose-files folder
  - Click on the latest file

The Docker Compose File

- The Docker Compose YAML is a manifest file. It specifies to Docker …
  - What containers to pull down and start
  - What infrastructure (like a network) is needed for your containers
  - The order in which to start/stop containers
  - …



The EdgeX Docker Compose YAML

EDGE X FOUNDRY™

```
version: '3'
services:
  volume:
    image: edgexfoundry/docker-edgex-volume
    container_name: edgex-files
    networks:
      - edgex-network
    volumes:
      - /data/db
      - /edgex/logs
      - /consul/config
      - /consul/data
  …
```

```
  logging:
    image: edgexfoundry/docker-support-logging
    ports:
      - "48061:48061"
    container_name: edgex-support-logging
    hostname: edgex-support-logging
    networks:
      - edgex-network
    volumes_from:
      - volume
    depends_on:
      - volume
      - config-seed
      - mongo
  …
```

Docker Compose Commands

- Docker Compose is a command line tool
- Common commands
  - docker-compose **pull** -f <compose file name>
    - *pull the images but don't start them*
  - docker-compose -f <compose file name> **up** -d
    - *create and start all containers – the default compose file name is docker-compose.yml*
    - *pull the images if not already pulled*
    - *-d means to start them all as daemon processes*
  - docker-compose -f <compose file name> **up** <docker image> -d
    - *without the image name, all containers are brought up*
  - docker-compose -f <compose file name> **stop** <docker image>
    - *stop an existing container*
    - *stop all images if the image name is left off*
  - docker-compose -f <compose file name> **start** <docker image>
    - *start an existing container that has been stopped*
    - *start all images if the image name is left off*
  - docker-compose -f <compose file name> **logs** -f --tail=100 <docker image>
    - *look at the last 100 lines of a micro services logs*

Start EdgeX using Docker Compose

- From the command line, go to the folder holding the docker-compose.yml file
- Pull all the EdgeX Foundry containers with one command:
  - docker-compose pull
- Start all of EdgeX Foundry containers with one command:
  - docker-compose up –d
  - -d option is to start the container as a background daemon
- Start each individual EdgeX Docker Container with a command:
  - docker-compose up –d [compose name]
  - See the next slide for the compose names and start order

The EdgeX Containers

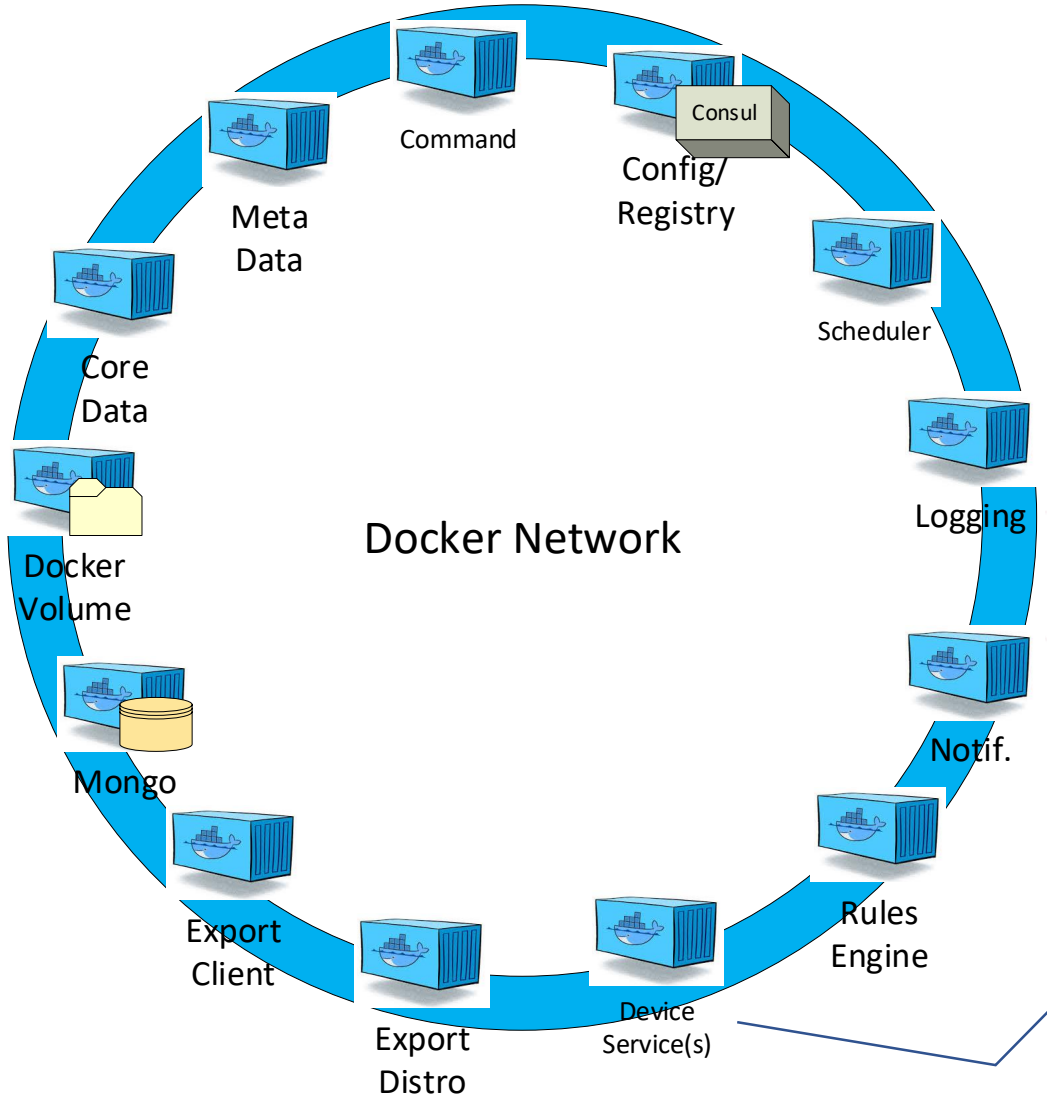
- Depending on the version of EdgeX as well as your use case, the EdgeX Docker Compose file will list several EdgeX containers

Container	Purpose
mongo	Mongo Database instance, and data initialization for the default NoSQL database for all of EdgeX
consul	Hashicorp's Consul configuration and registry service
data	Core Data, centralized persistence facility for data readings collected by devices and sensors
metadata	Core Metadata, knowledge about the devices and sensors and how to communicate with them
command	Core Command, enables the issuance of commands or actions to devices and sensors on behalf of other micro services, other applications, external systems
scheduler	Support Scheduling, provides facilities to kick off various events/actions on a timed schedule such as old data scrubbing
logging	Support Logging, central logging service for all micro services
notifications	Support Notifications, central alert and notification service for all micro services
rulesengine	Support Rules Engine, micro service "wrapped" Drools Rules Engine that monitors incoming sensor or device data for readings within target ranges and triggers immediate device actuation
export-client	Export Client, enables clients, whether they are on-gateway or off-gateway, to register as recipients of data coming through Core Data
export-distro	Export Distribution, receives data from Core Data, through a message queue, then filters, transforms, and formats the data per client request, and distributes to the appropriate endpoint by pre-registered protocol
device-virtual	Software that mimics the behavior of a sensor for purposes of demonstrating and exploring EdgeX

EdgeX Infrastructure

- EdgeX micro services won't be the only thing listed and brought up with Docker Compose
- EdgeX relies on a shared file space among services (called a Docker volume)
  - Allows the database files to be shared across services
  - Allows log file space to be shared across services
- EdgeX use MongoDB as its default persistence storage
  - Mongo has been containerized for EdgeX use
- EdgeX uses Consul as its registry and configuration service
  - Consul has been containerized for EdgeX use
- EdgeX config-seed is a service that initializes Consul with EdgeX configuration data
  - config-seed exits quickly after populating Consul (i.e. it is not long running)
- EdgeX needs all micro services to be connected to a virtual network
  - Docker provides a virtual network facility
  - The Docker Compose file specifies the network and includes all the services and infrastructure on that network

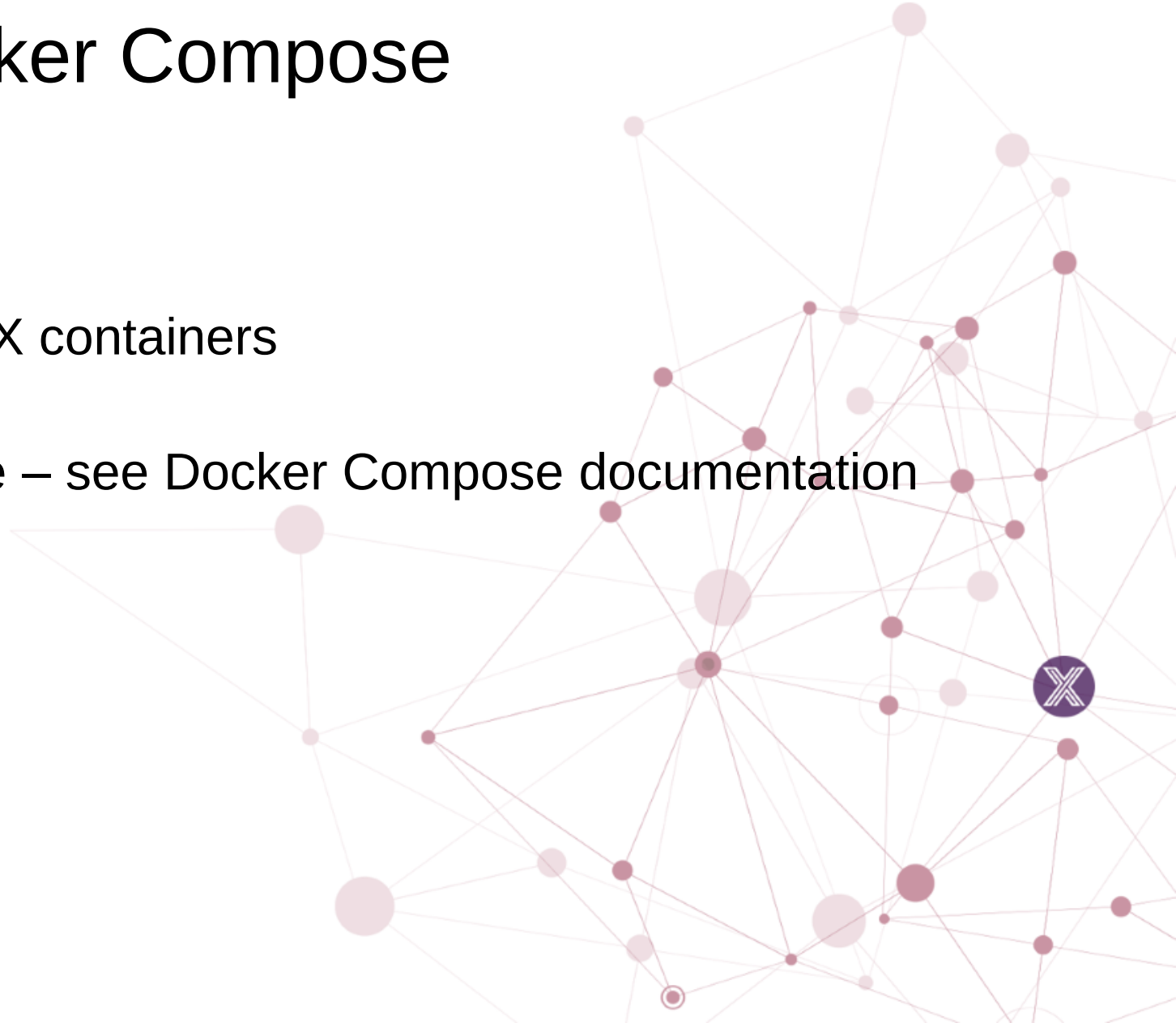
The Typical EdgeX Deployment



Number of device services will vary per use case and number of sensors/devices

Stop EdgeX using Docker Compose

- To stop all the EdgeX containers
  - docker-compose stop
- To stop and remove all the EdgeX containers
  - docker-compose down
- More command options available – see Docker Compose documentation

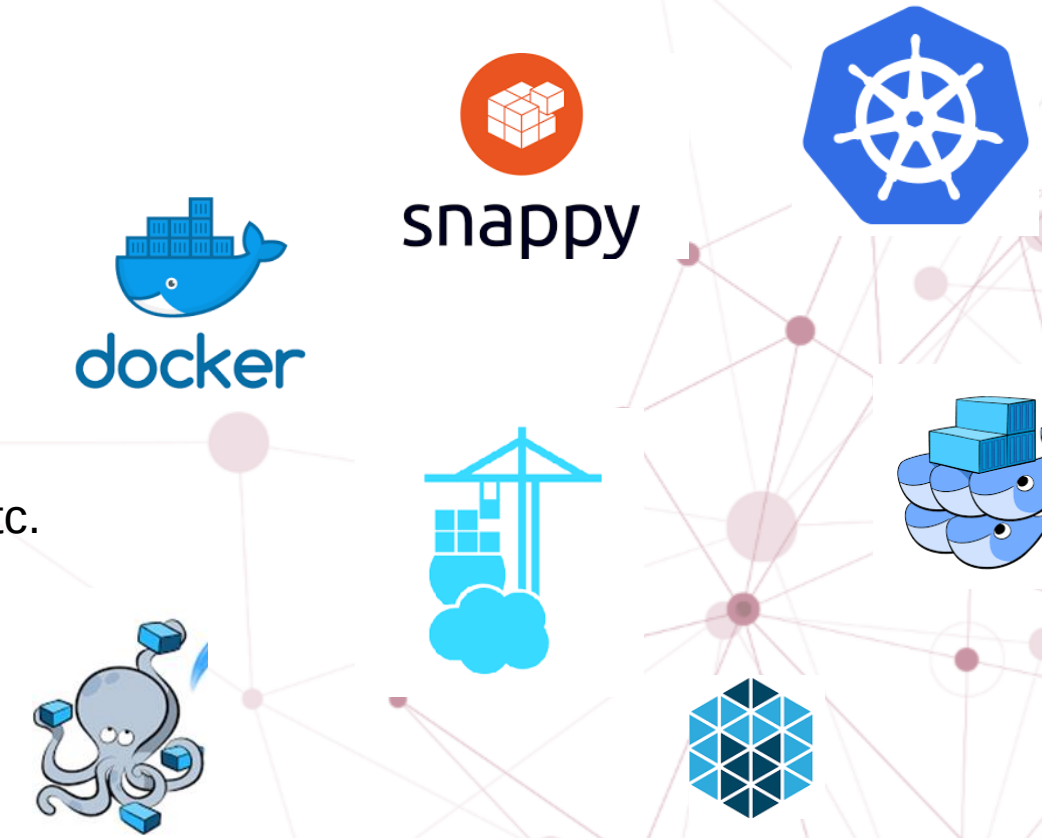


Check that EdgeX Containers are working

- Check with the EdgeX registry service to see which services are up
  - http://<host name>:8500
- Any container log can be viewed with:
  - docker-compose logs -f --tail 50 [compose name]
  - ex: docker-compose logs -f --tail 50 core-data
- Ping the service
  - http://<host name>:<service port>/api/v1/ping
  - ex: http://localhost:48080/api/v1/ping for core data ping
- See the event count collected
  - http://<host name>:48080/api/v1/event/count
- See the devices “connected”
  - http://<host name>:48081/api/v1/device
- See the event/reading data from a particular device
  - http://<host name>:48080/api/v1/event/device/<id>/<count limit>

Alternate Deployments

- EdgeX is agnostic with regard to deployment (and orchestration)
  - Docker is used as our reference deployment capability
  - We also provide Ubuntu Snaps
    - For Canonicals' Ubuntu Core
  - Members of our community use Portainer to deploy and orchestrate EdgeX
  - You could use Kubernetes, Swarm, Mesos, etc.
- EdgeX is BYOD



Key Project Links

Access the code:
https://github.com/edgexfoundry

Access the technical documentation:
https://docs.edgexfoundry.org/

Access technical video tutorials:
https://wiki.edgexfoundry.org/display/FA/EdgeX+Tech+Talks

EdgeX Blog:
https://www.edgexfoundry.org/news/blog/

Join an email distribution:
https://lists.edgexfoundry.org/mailman/listinfo

Join the Rocket Chat:
https://chat.edgexfoundry.org/home

Become a project member:
https://www.edgexfoundry.org/about/members/join/

LinkedIn:
https://www.linkedin.com/company/edgexfoundry/

Twitter:
https://twitter.com/EdgeXFoundry

YouTube:
https://www.youtube.com/edgexfoundry

James_White2@dell.com

Upcoming Tech Talks

- Creating a new Device Service (Steve Osselton – DS WG lead)
- Creating a new Export Client (Janko Isidorovic – Application WG lead)

- Email Michael or I other suggestions

EDGE X FOUNDRY™

Questions and Answer Time

Thank You!

james_white2@dell.com

mhall@linux.com