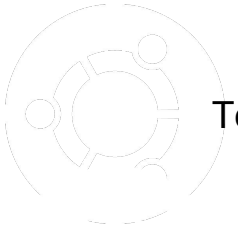
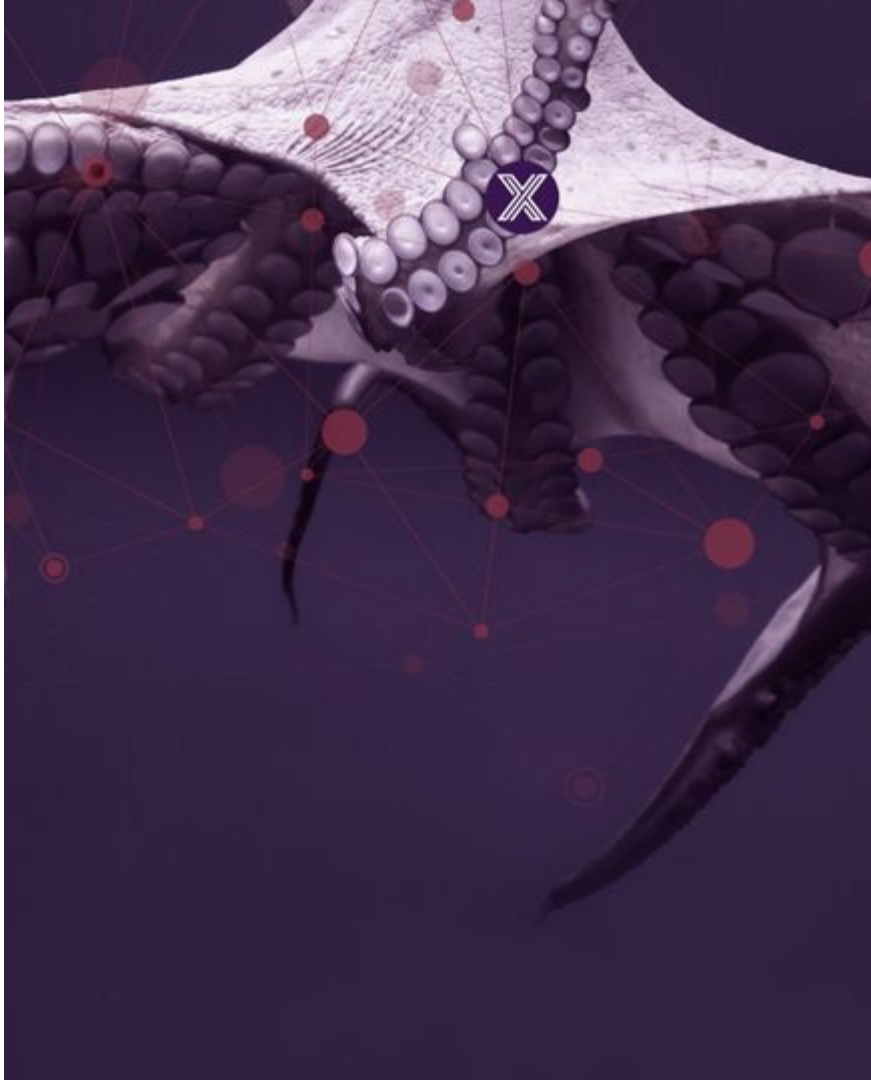
EDGE X FOUNDRY™

Snap Package



Tech Talks - Session 11



edgexfoundry.org | @edgexfoundry

Agenda

- Introduction to snaps
- Overview of the `edgexfoundry` snap
- How to install
- How to configure/manage/update
- How to use with additional device services
- Further references
- Upcoming tech talks
- Q&A

Ian Johnson

<ian.johnson@canonical.com>

- Canonical / Software Engineer
  Field Engineering - Devices & IoT
- Primary snap developer for Dehli release
- Contributed CI work for snap build
- Contributed code to security to decouple Docker-isms
- Involved in testing most services for Dehli
- Contributed bug fixes to SMA
- Member of the DevOps working group

Tony Espy

<espy@canonical.com>

- Canonical / Technical Architect
  Field Engineering - Devices & IoT
- Technical Steering Committee member
- Former Device Services WG chair
- Author of Device Services SDK Requirements
- Original developer of device-sdk-go
- Active member of Core, Device Services & Security working groups
- Created first EdgeX snap prototype

Introduction

What's a snap?

Snaps are containerised software packages that work on all major Linux distributions without modification. Simple to create and publish, they automatically update safely.

Introduction

Snaps are...

- Self-contained squashfs-based software packages
  - Containing one or more applications or services
- Cryptographically-signed by publisher and tamper-proof
- Published to risk-based update channels
- Updated automatically via binary-delta downloads
- Updated transactionally, and as such, support rollback
- Multi-architecture
- Sand-boxed by default
  - Using a set of standard Linux Security Modules (e.g. AppArmor, Seccomp, …)

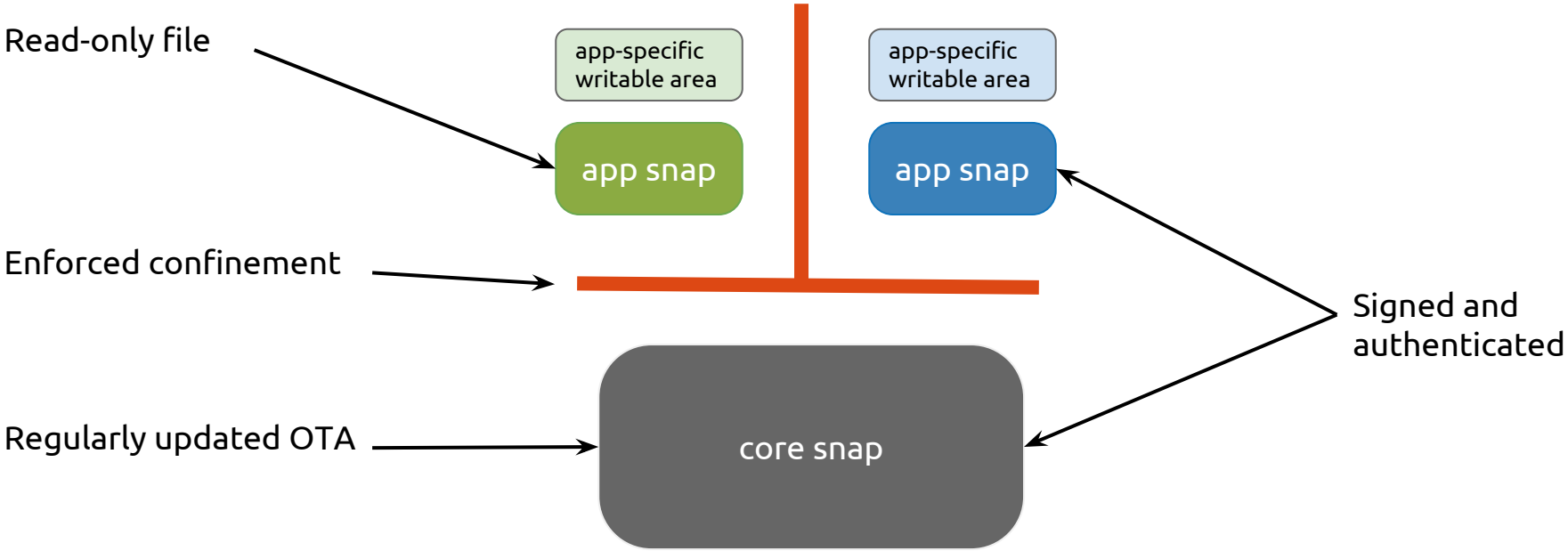
Introduction

Snaps...

- Contain all[1] required runtime dependencies
  - No need to rely on OS updates
- May contain open source or proprietary software
- Can be used commercially without license
- Can also be provided via private "Brand" stores which:
  - Allow snaps to be limited to specific brands and/or device models
  - Allow the brand to grant automatic permissions to snaps (e.g. TPM access)

1 - excludes any binaries or shared libraries provided by the base/core snap

Application isolation



https://docs.snapcraft.io/core/snapd

edgexfoundry snap

- In contrast to Docker deployment, the edgexfoundry snap contains:
  - All of the EdgeX Go-based microservices
    - Including all of the new go SDK based device services
    - Including the security services
  - Consul, Kong, MongoDB, Cassandra and Vault
- Each service has a systemd service unit auto-generated on install
  - Enabled services are auto-started on boot
  - Services that exit prematurely are restarted
  - Services can be enabled/disabled via snap configuration
  - Services can be started/stopped/restarted via snap service command
- Service config files (incl config-seed) can be modified within the snap
- Security services are enabled by default

How to install?

- First install snapd: https://docs.snapcraft.io/installing-snapd
- The snap can be installed on any system that supports snaps
- For full security confinement, install on:
  - Ubuntu 16.04 LTS or later (Desktop or Server)
  - Ubuntu Core 16 or later

How to install contd...

- To view available versions of the snap:

  ```
  $ snap info edgexfoundry
  ```

- To install the most recent stable release (currently Delhi)

  ```
  $ sudo snap install edgexfoundry
  ```

- To install the Dehli snap:

  ```
  $ sudo snap install edgexfoundry --channel=delhi
  ```

- To install the latest daily build of Edinburgh:

  ```
  $ sudo snap install edgexfoundry --edge
  ```

How to configure/manage/update

- To list running services:

  ```
  $ snap services edgexfoundry
  ```

- To enable/disable a service:

  ```
  $ snap config edgexfoundry <service-name>=on|off
  ```

- To manually* update the snap:

  ```
  $ sudo snap refresh edgexfoundry
  ```

- Logs for all services can be found in $SNAP_COMMON (usually /var/snap/edgexfoundry/common)

* snaps automatically update themselves - commercial customers can control updates with a brand store

How to use with additional device services?

- On most Linux distros, additional device services can be deployed via Debian packages or snaps.
- On Ubuntu Core, additional device services can only be provided via snap packages.
- A device service is responsible for ensuring that its dependent services (i.e. Core Metadata, Core Data, …) are running before becoming operational.
- A new device service is responsible for seeding its initial configuration into the registry on first run. If existing configuration is found, the DS will just use it.

Q&A