PARSEC

Parsec – Platform Abstraction for Security

Project Introduction For Yocto Virtual Summit

May 26th 2021

Speakers

Paul Howard

Principal System Solutions Architect

paul.howard@arm.com

@paulhowardarm

Anton Antonov

Senior Engineer

anton.antonov@arm.com

@anta5010

Agenda

What is Parsec?

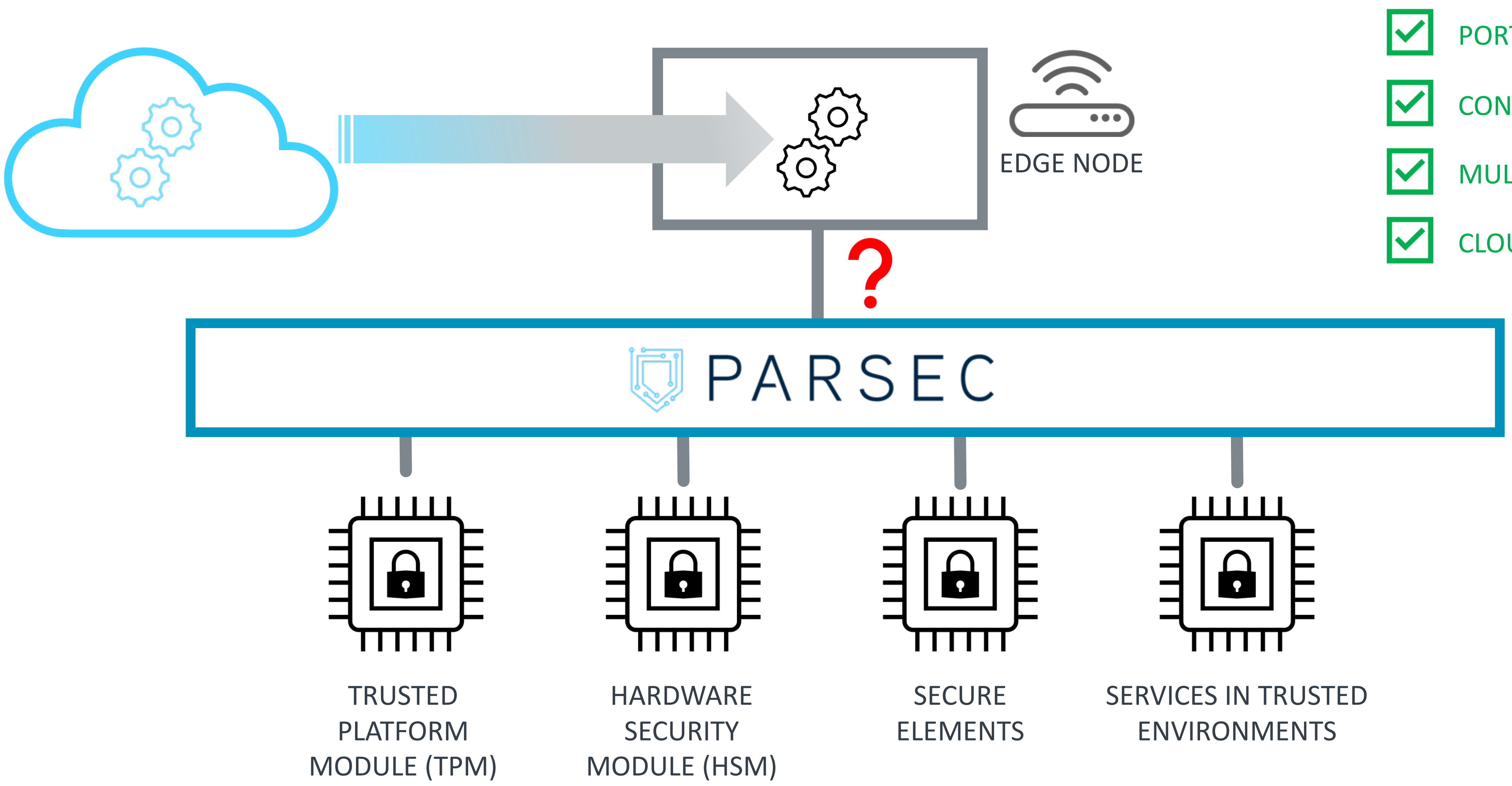
Using Parsec

Details of Parsec/Yocto Integration

Questions

PARSEC

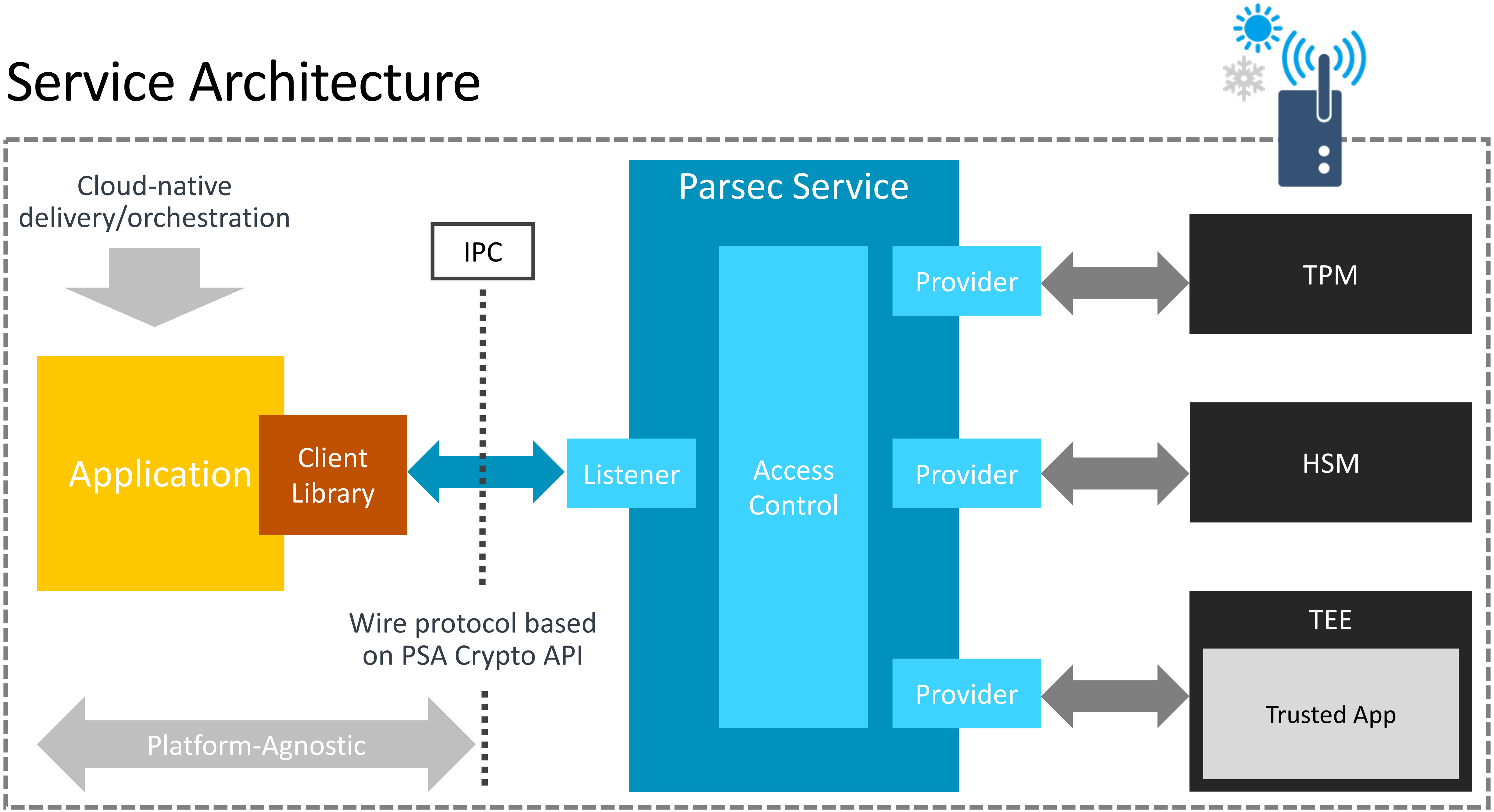
Parsec: A **Platform Abstraction** For **Security**



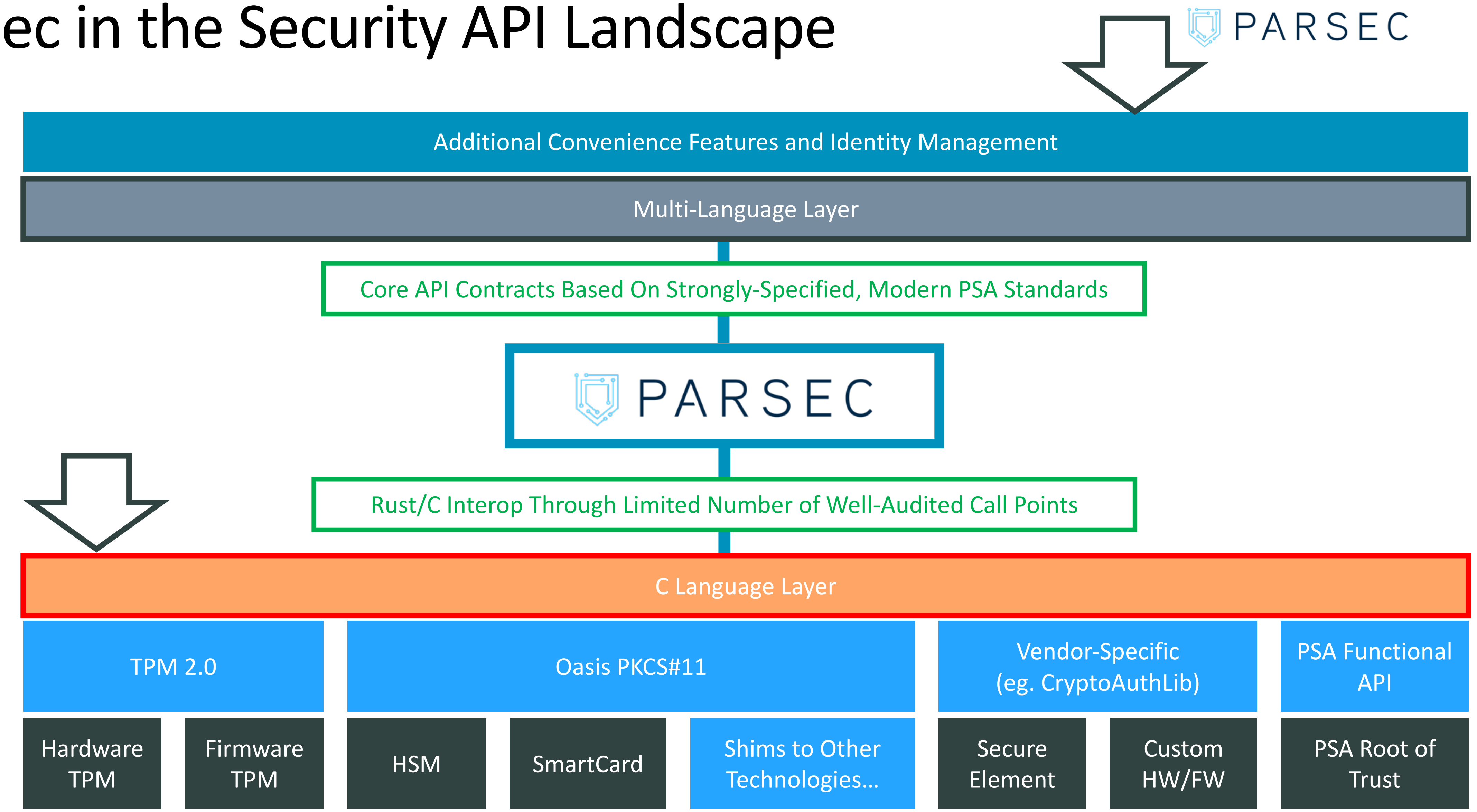
EDGE NODE
?
PARSEC
TRUSTED PLATFORM MODULE (TPM)
HARDWARE SECURITY MODULE (HSM)
SECURE ELEMENTS
SERVICES IN TRUSTED ENVIRONMENTS

- ✓ PORTABLE
- ✓ CONVENIENT
- ✓ MULTI-TENANT
- ✓ CLOUD-NATIVE

Service Architecture



Parsec in the Security API Landscape


PARSEC
Additional Convenience Features and Identity Management
Multi-Language Layer
Core API Contracts Based On Strongly-Specified, Modern PSA Standards
PARSEC
Rust/C Interop Through Limited Number of Well-Audited Call Points
C Language Layer
TPM 2.0
Oasis PKCS#11
Vendor-Specific (eg. CryptoAuthLib)
PSA Functional API
Hardware TPM
Firmware TPM
HSM
SmartCard
Shims to Other Technologies…
Secure Element
Custom HW/FW
PSA Root of Trust

The Growing Ecosystem

NXP

56K CLOUD

docker

MIRANTIS

arm

CLOUD NATIVE SANDBOX

Sandbox Projects
Incubating Projects
Graduated Projects
EARLY STAGE

pelion

spiffe

SPIRE

SolidRun
Embedded Edge Computing

nixu
cybersecurity.

Red Hat

openSUSE

yocto PROJECT VIRTUAL SUMMIT

PARSEC

Why Add Parsec To Yocto?

- Architecture Neutral
- Supports Diverse Hardware Through Customization
- Common Developer Experience Across Platforms
- Commitment to Open Development
- Targeting IoT/Embedded/Edge Space
- Complements Packaging For Off-The-Shelf Distros

Meta-parsec is a sublayer of meta-security since **Hardknott**

https://git.yoctoproject.org/cgit/cgit.cgi/meta-security/tree/meta-parsec

How To Use Parsec In Yocto

Build

Include **meta-parsec, meta-rust and meta-clang** in your layers list. Include **parsec-service** into your image.

Configure

Parsec is configured simply with a **TOML** file. Examples are provided to connect the service with **TPM**, **HSM/PKCS#11** or **software** back-ends.

Run

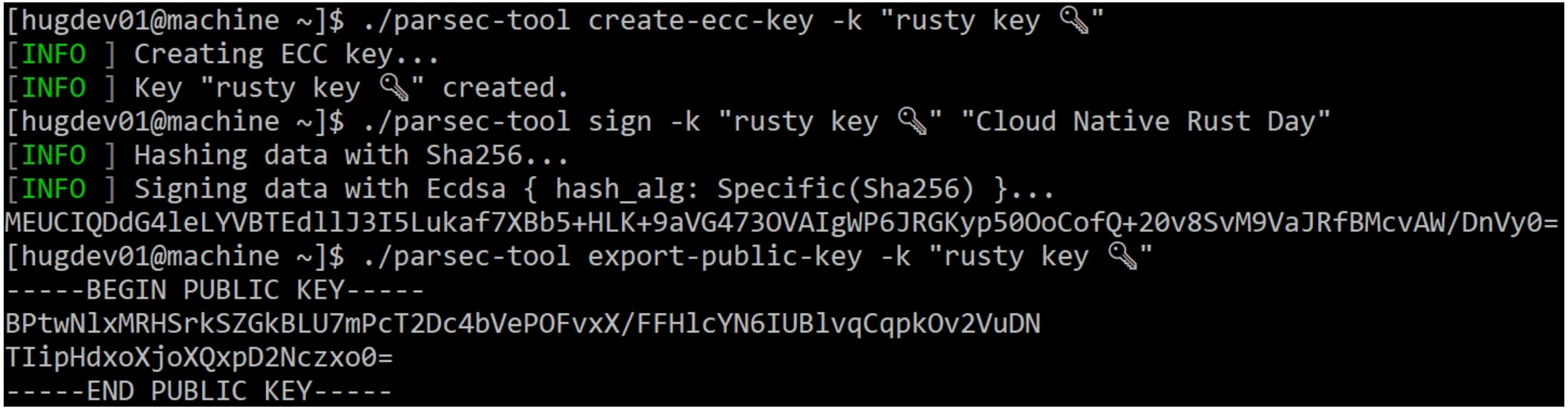
The **Parsec service** is a single executable that runs locally. It can be managed with **systemd** (recommended), or SysV init scripts.

Consume

Use the command-line **parsec-tool** if desired or consume the APIs into your code from languages including **Rust**, **C** and **Go,** with more to come…

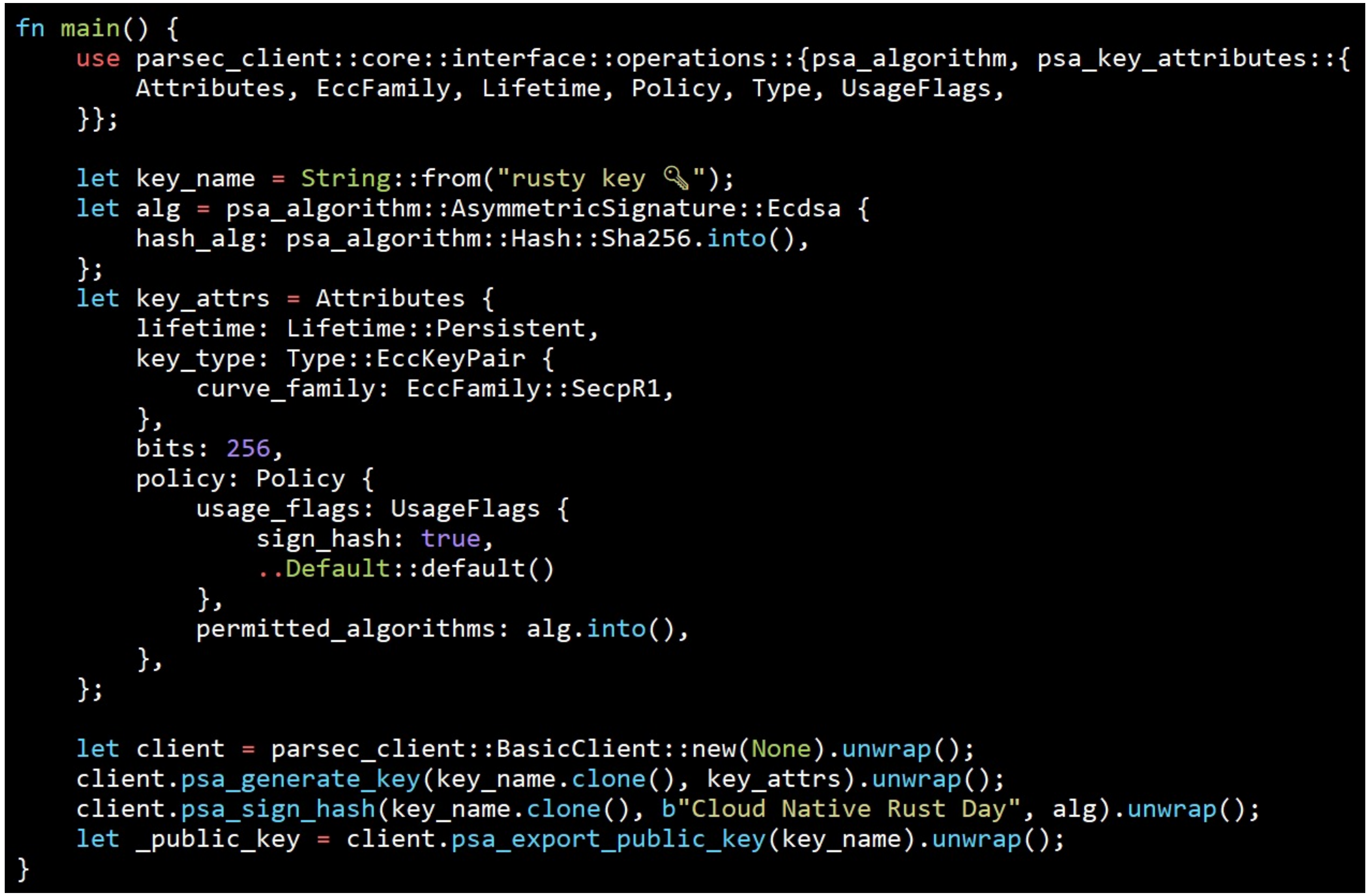
https://git.yoctoproject.org/cgit/cgit.cgi/meta-security/tree/meta-parsec/README.md

The Developer Experience: Command-Line Example



```
[hugdev01@machine ~]$ ./parsec-tool create-ecc-key -k "rusty key 🔑"
[INFO ] Creating ECC key...
[INFO ] Key "rusty key 🔑" created.
[hugdev01@machine ~]$ ./parsec-tool sign -k "rusty key 🔑" "Cloud Native Rust Day"
[INFO ] Hashing data with Sha256...
[INFO ] Signing data with Ecdsa { hash_alg: Specific(Sha256) }...
MEUCIQDdG4leLYVBTEdllJ3I5Lukaf7XBb5+HLK+9aVG473OVAIgWP6JRGKyp50OoCofQ+20v8SvM9VaJRfBMcvAW/DnVy0=
[hugdev01@machine ~]$ ./parsec-tool export-public-key -k "rusty key 🔑"
-----BEGIN PUBLIC KEY-----
BPtwNlxMRHSrkSZGkBLU7mPcT2Dc4bVePOFvxX/FFHlcYN6IUBlvqCqpkOv2VuDN
TIipHdxoXjoXQxpD2Nczxo0=
-----END PUBLIC KEY-----
```

The Developer Experience: Rust Example



```rust
fn main() {
    use parsec_client::core::interface::operations::{psa_algorithm, psa_key_attributes::{
        Attributes, EccFamily, Lifetime, Policy, Type, UsageFlags,
    }};

    let key_name = String::from("rusty key 🔑");
    let alg = psa_algorithm::AsymmetricSignature::Ecdsa {
        hash_alg: psa_algorithm::Hash::Sha256.into(),
    };
    let key_attrs = Attributes {
        lifetime: Lifetime::Persistent,
        key_type: Type::EccKeyPair {
            curve_family: EccFamily::SecpR1,
        },
        bits: 256,
        policy: Policy {
            usage_flags: UsageFlags {
                sign_hash: true,
                ..Default::default()
            },
            permitted_algorithms: alg.into(),
        },
    };

    let client = parsec_client::BasicClient::new(None).unwrap();
    client.psa_generate_key(key_name.clone(), key_attrs).unwrap();
    client.psa_sign_hash(key_name.clone(), b"Cloud Native Rust Day", alg).unwrap();
    let _public_key = client.psa_export_public_key(key_name).unwrap();
}
```

PARSEC

Some Details of Parsec Integration into Yocto

See also "**Using Rust with bitbake and meta-rust**" with Steven Walter, 14:45 UTC (Presentation Room)

yocto PROJECT

Choices For Including Rust-Based Software in Yocto

Toolchain

meta-rust ✓

https://github.com/meta-rust/meta-rust

- Builds rust compiler and cargo build system from source
  - Provides "crate" fetch mechanism for dependencies

meta-rust-bin

https://github.com/rust-embedded/meta-rust-bin

- Uses pre-built upstream versions of compiler and cargo
  - Faster to build, but less flexible

Dependency Management

bitbake vendoring ✓

https://github.com/meta-rust/cargo-bitbake

- Dependencies modelled explicitly in the recipe
  - Needs to be kept in sync with **Cargo.toml**
    - Tools to auto-generate include files

cargo vendoring

- Cargo build system fetches crates by itself
- Add `CARGO_DISABLE_BITBAKE_VENDORING = "1"` to recipe

Parsec Service Recipe (Fragment)

```
inherit cargo
SRC_URI += "crate://crates.io/parsec-service/${PV} \
"


CARGO_BUILD_FLAGS += " --features all-providers,cryptoki/generate-bindings,tss-esapi/generate-bindings”
DEPENDS = "tpm2-tss"
TOOLCHAIN = "clang”
PARSEC_CONFIG ?= "${S}/config.toml"

do_install_append () {
  install -d -m 700 -o parsec -g parsec "${D}${libexecdir}/parsec"
  install -m 700 -o parsec -g parsec "${WORKDIR}/build/target/${CARGO_TARGET_SUBDIR}/parsec" ${D}${libexecdir}/parsec/parsec
}
require parsec-service_${PV}.inc
```

parsec-service_%.bbappend

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"SRC_URI += "file://config-tpm.toml \
"
PARSEC_CONFIG = "${WORKDIR}/config-tpm.toml"
```

Rust Recipes In CI Pipelines

- Rust tool chain requirements can result in lengthy image build
- Use of persistent **SSTATE_DIR** and **DL_DIR** recommended
- See https://git.yoctoproject.org/cgit/cgit.cgi/meta-arm/tree/.gitlab-ci.yml

```
.setup:
  stage: build
  variables:
    KAS_REPO_REF_DIR: $CI_BUILDS_DIR/persist/repos
    SSTATE_DIR: $CI_BUILDS_DIR/persist/sstate
    DL_DIR: $CI_BUILDS_DIR/persist/downloads
  before_script:
    - echo SSTATE_DIR = $SSTATE_DIR
    - echo DL_DIR = $DL_DIR
```

Learn More

Get the code

https://github.com/parallaxsecond/parsec

Read the book

https://parallaxsecond.github.io/parsec-book

Join The Community

GitHub

https://github.com/parallaxsecond/community

slack

#parsec on CNCF https://slack.cncf.io

zoom

Every **Tuesday** at **16:30** (UK), **11:30** (US East), **08:30** (US West)

PARSEC

Q&A