
AWS
re:Invent

IOT303

Developing and deploying modern edge applications at scale

Richard Barry

Senior Principal Engineer, AWS IoT

FreeRTOS Founder

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What to expect from this session

Learn about AWS options for device software

Understand the value AWS provides to the FreeRTOS user base

Look at the functionality of new libraries

See how to use that functionality in your projects

See how to use that functionality at a huge scale



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

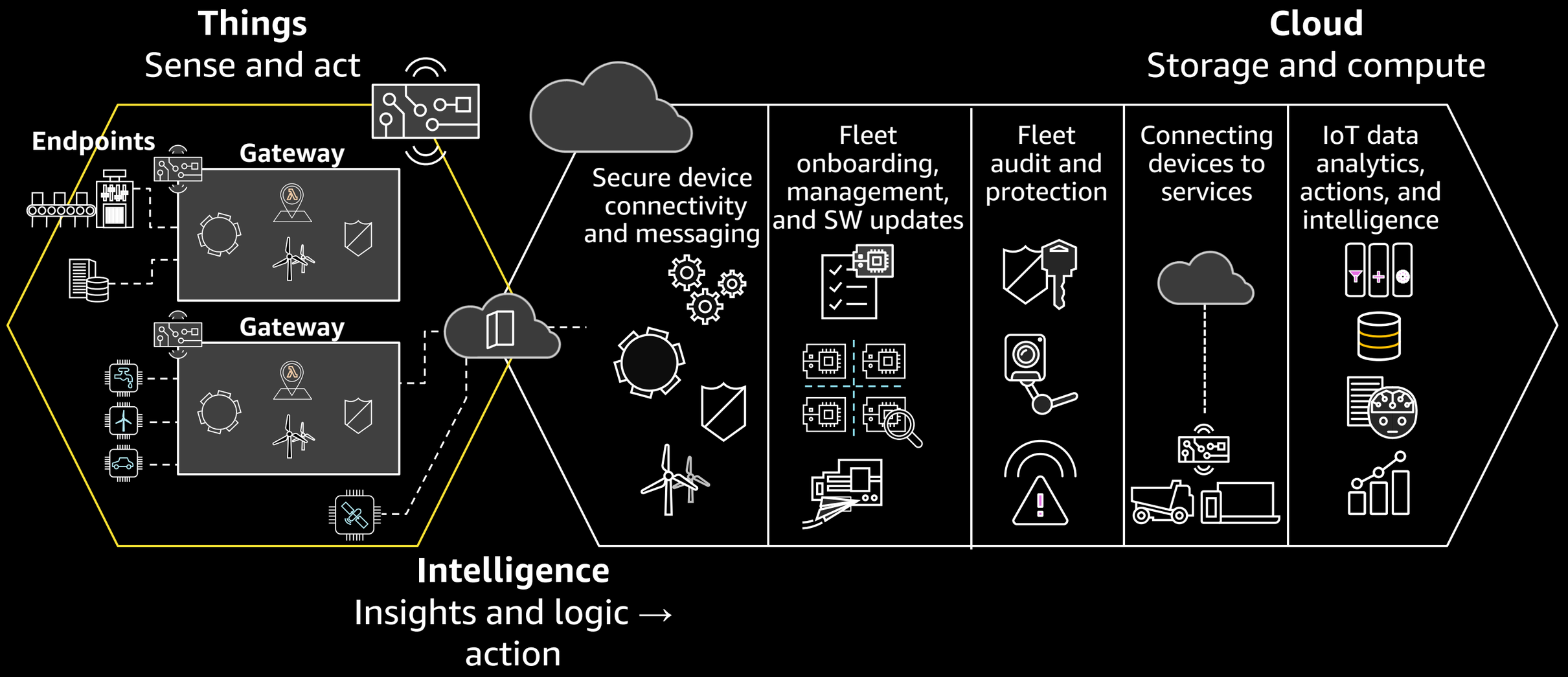
AWS IoT



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS IoT: Device and cloud software



Options for device software

AWS IoT
Device SDKs



FreeRTOS



AWS IoT
Greengrass



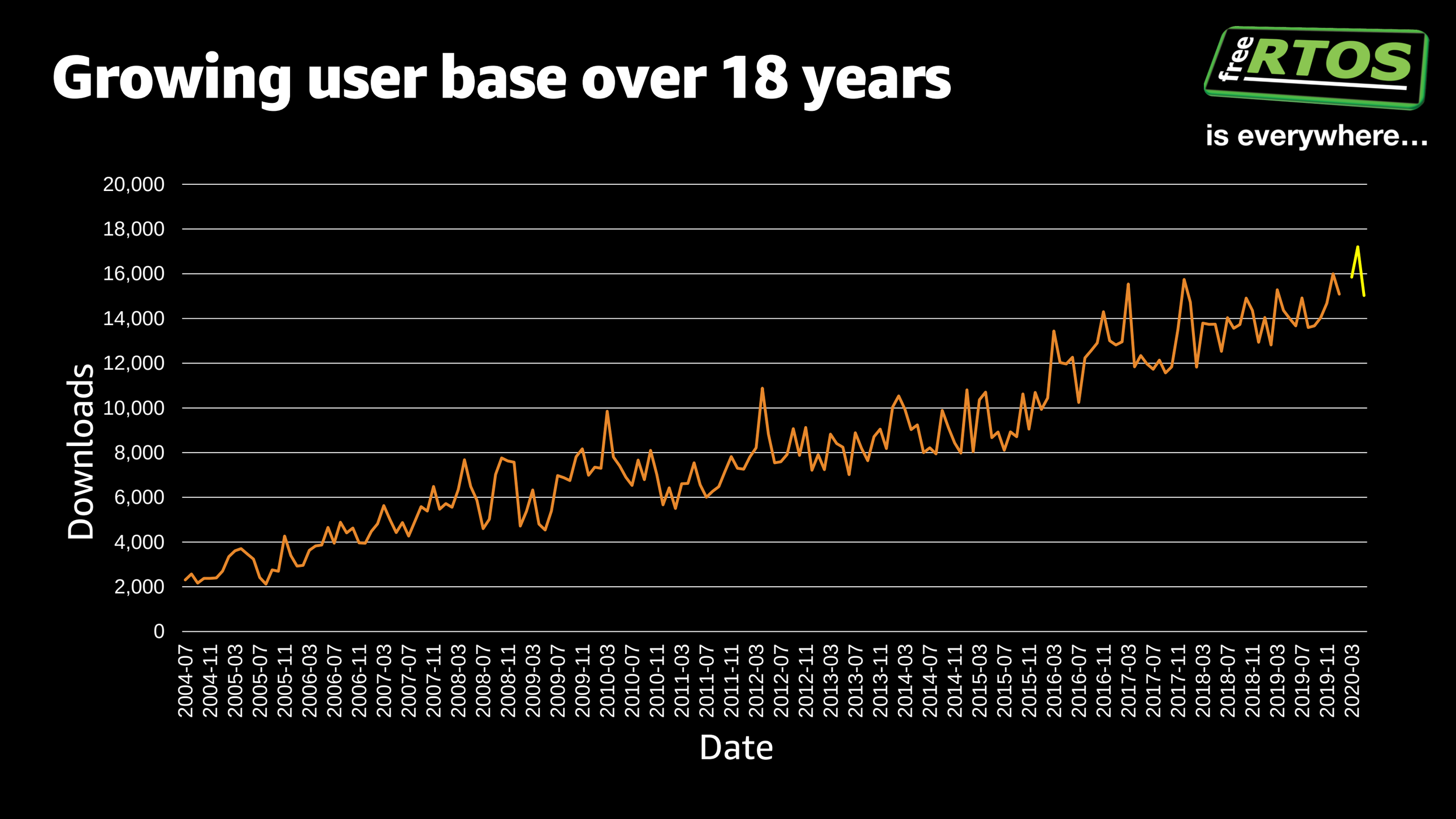
FreeRTOS



AWS re:Invent

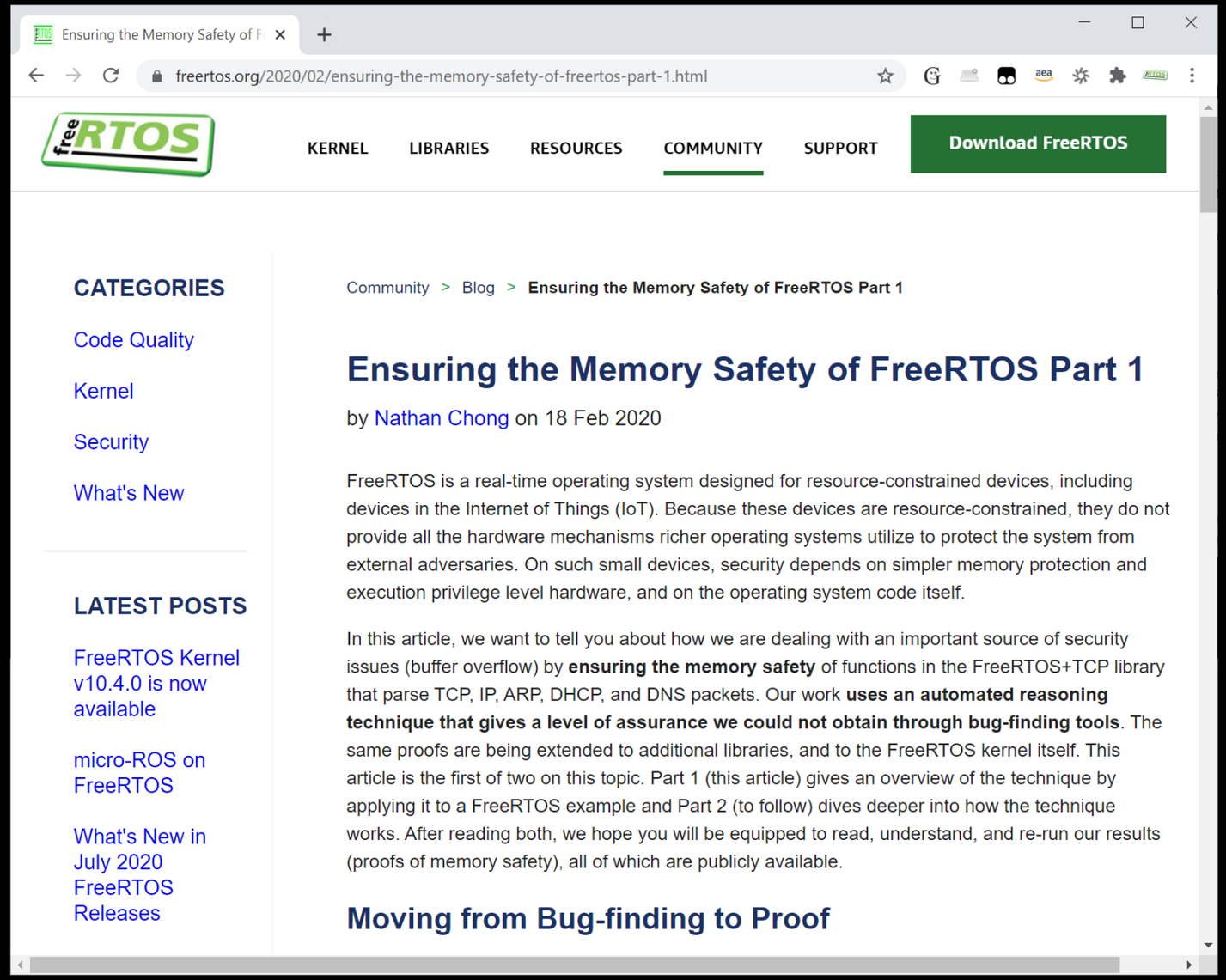
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Growing user base over 18 years



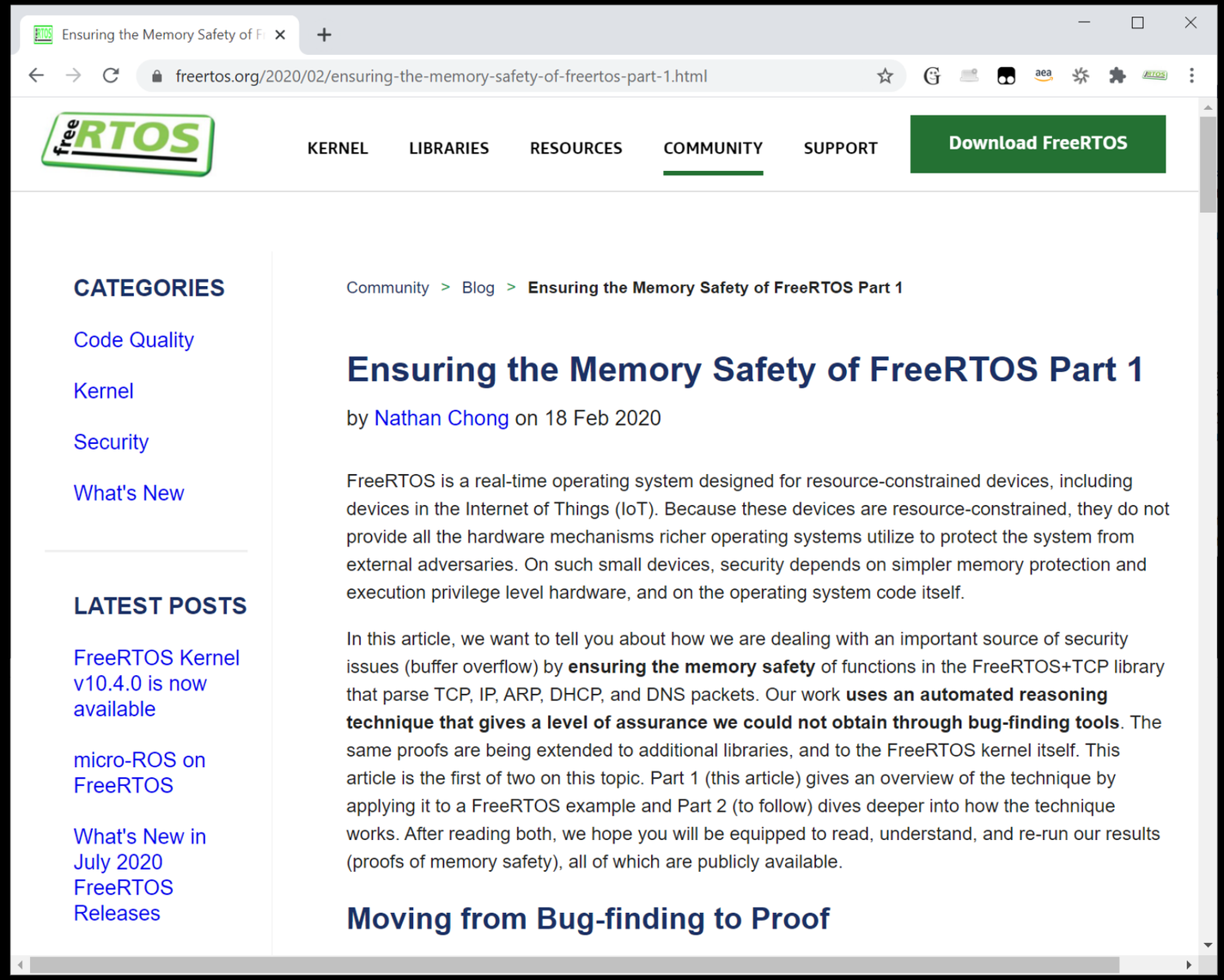
Amazon’s contributions to the community

- Global presence
- New functionality
- Kernel ports and enhancements
- Simplified licensing so open with no lock-in
- Professional incidence response processes
- Security expertise



Amazon's contributions to FreeRTOS

- Global presence
- New functionality
- Kernel ports and enhancements
- Simplified licensing so open with no lock-in
- Professional incidence response processes
- Security expertise
- **Long-term support**



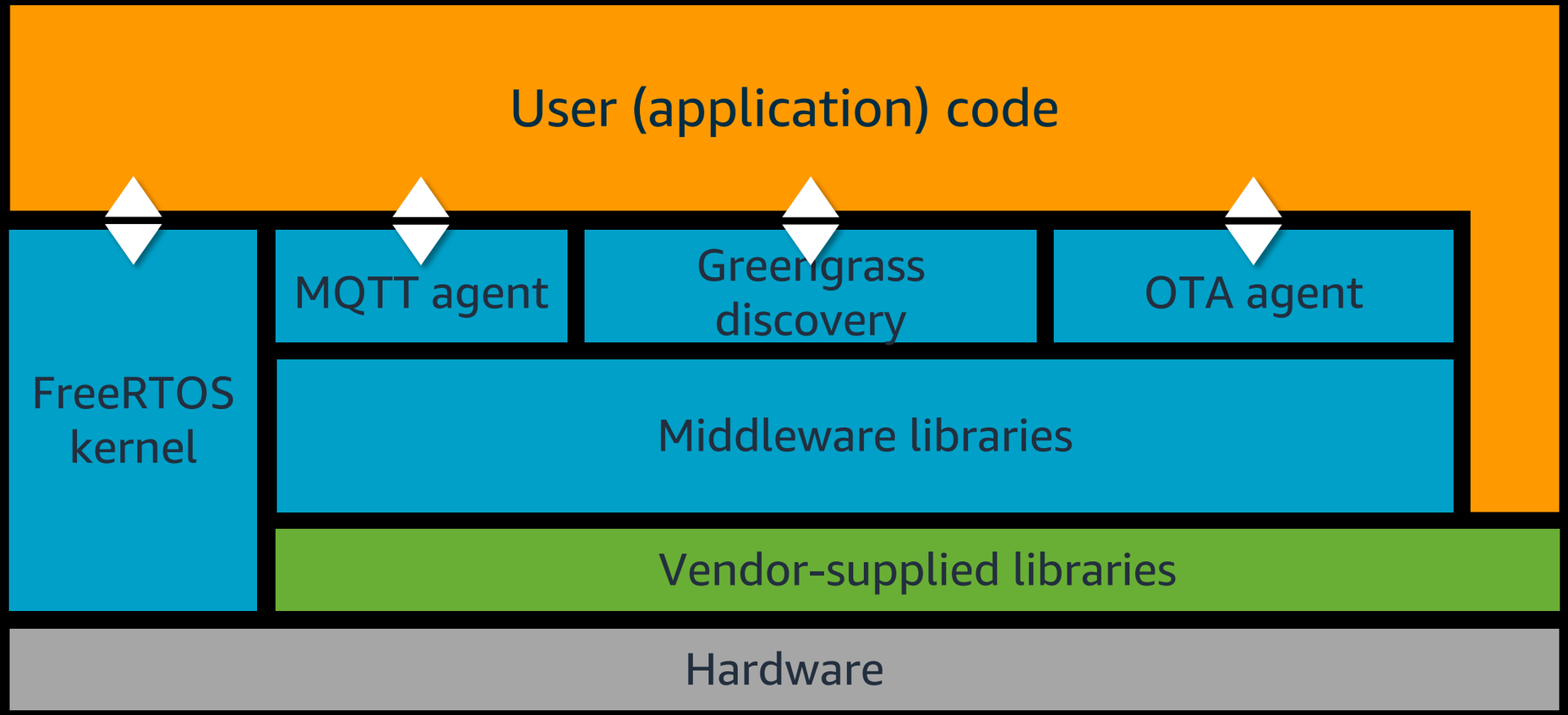
FreeRTOS: Latest capabilities



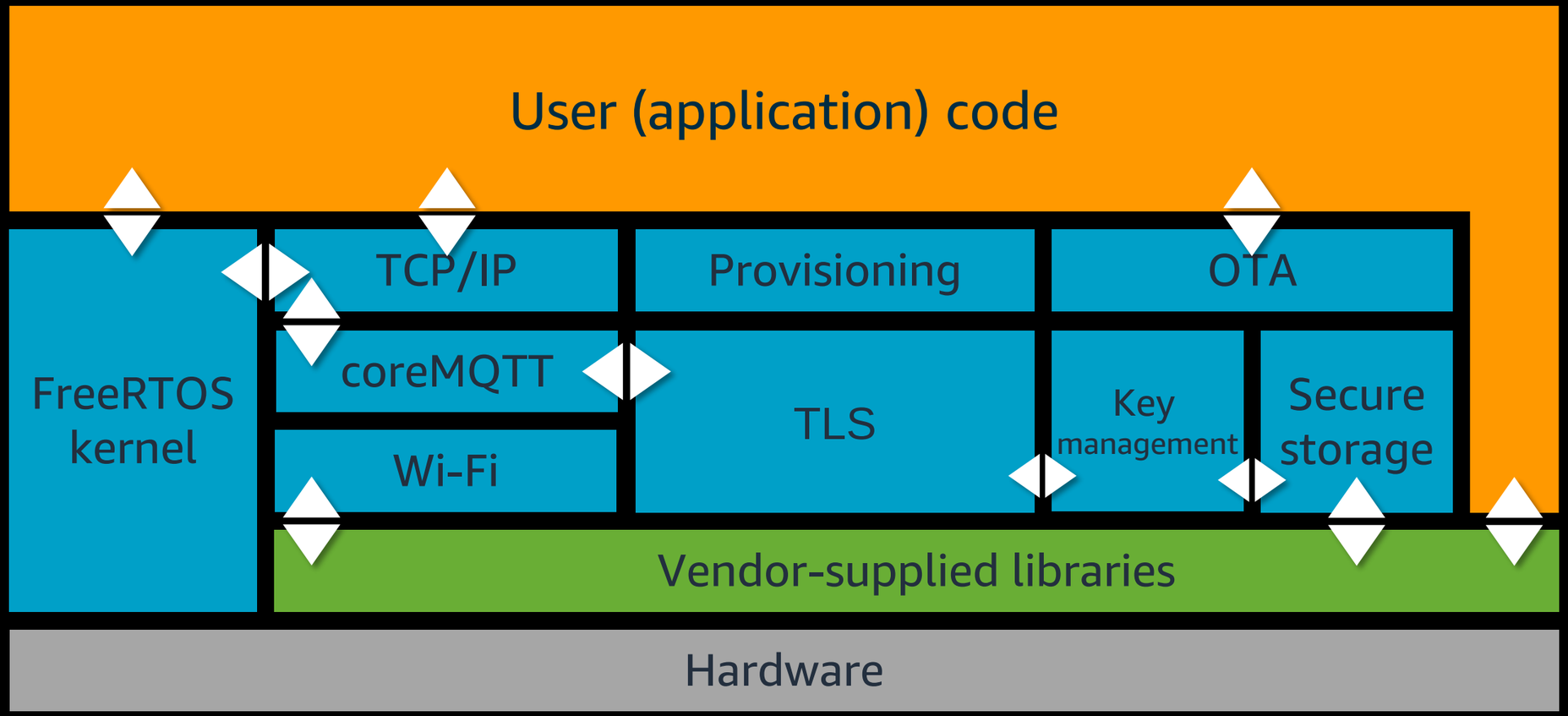
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

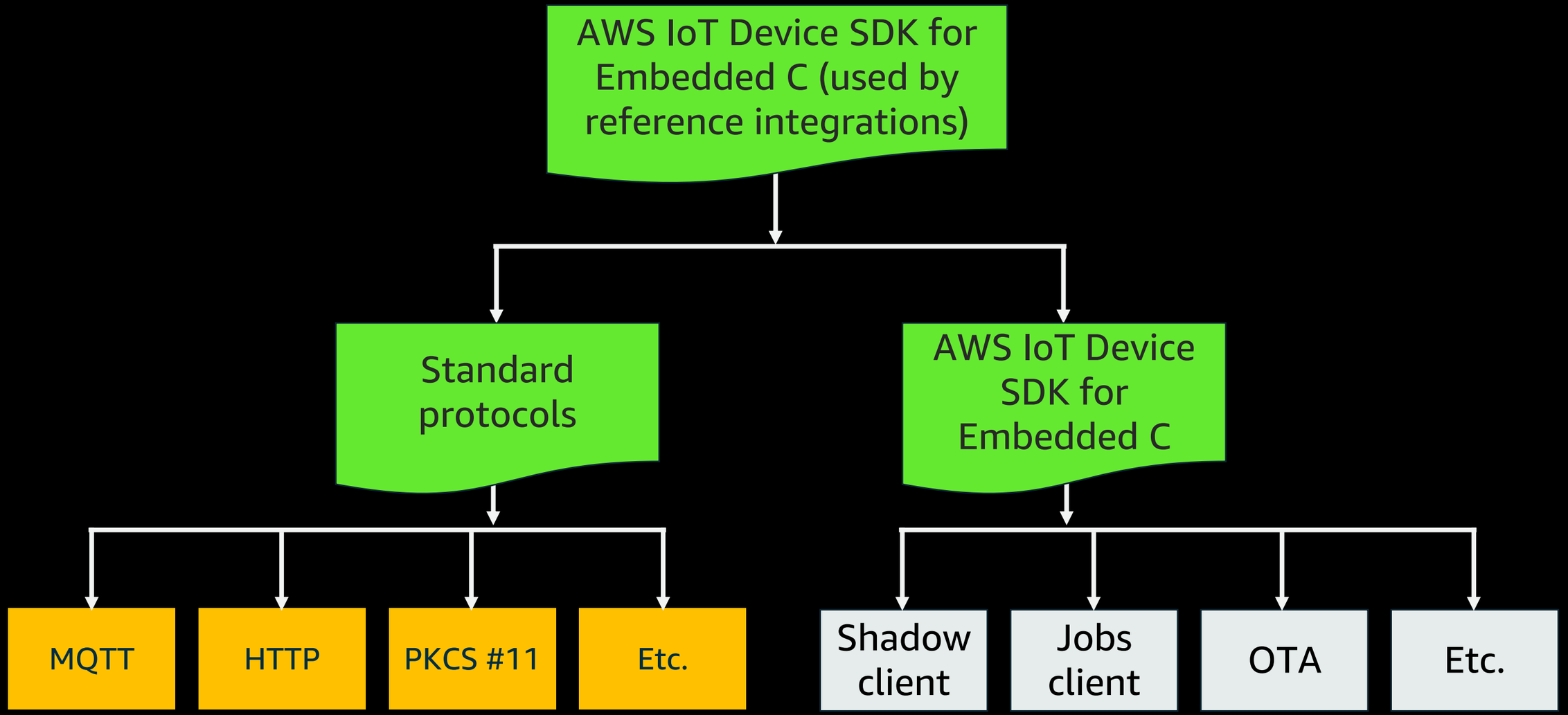
IoT reference integrations (devices.amazonaws.com)



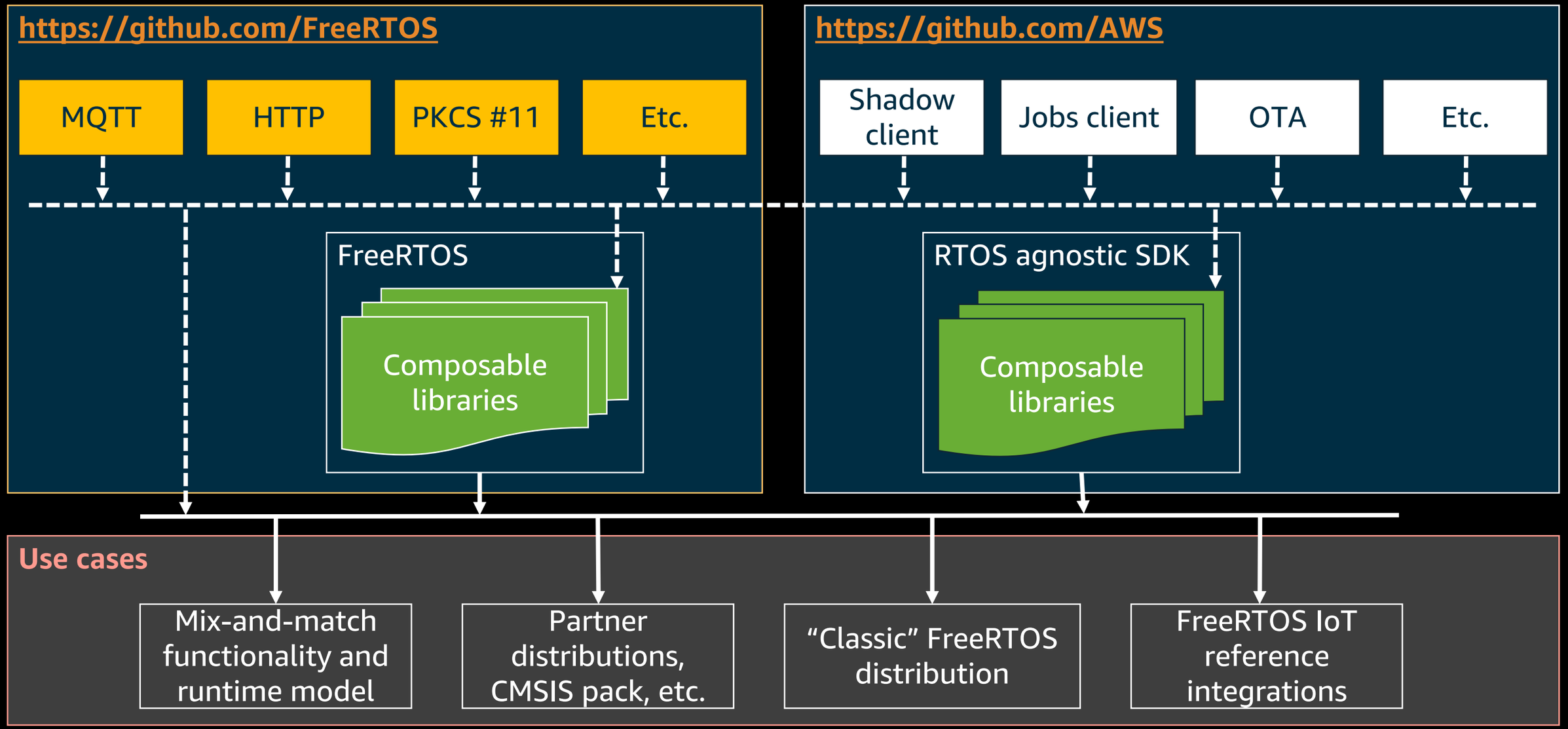
Reference integrations: Internal view



Refactoring for distribution from FreeRTOS.org



Libraries in FreeRTOS and AWS GitHub accounts



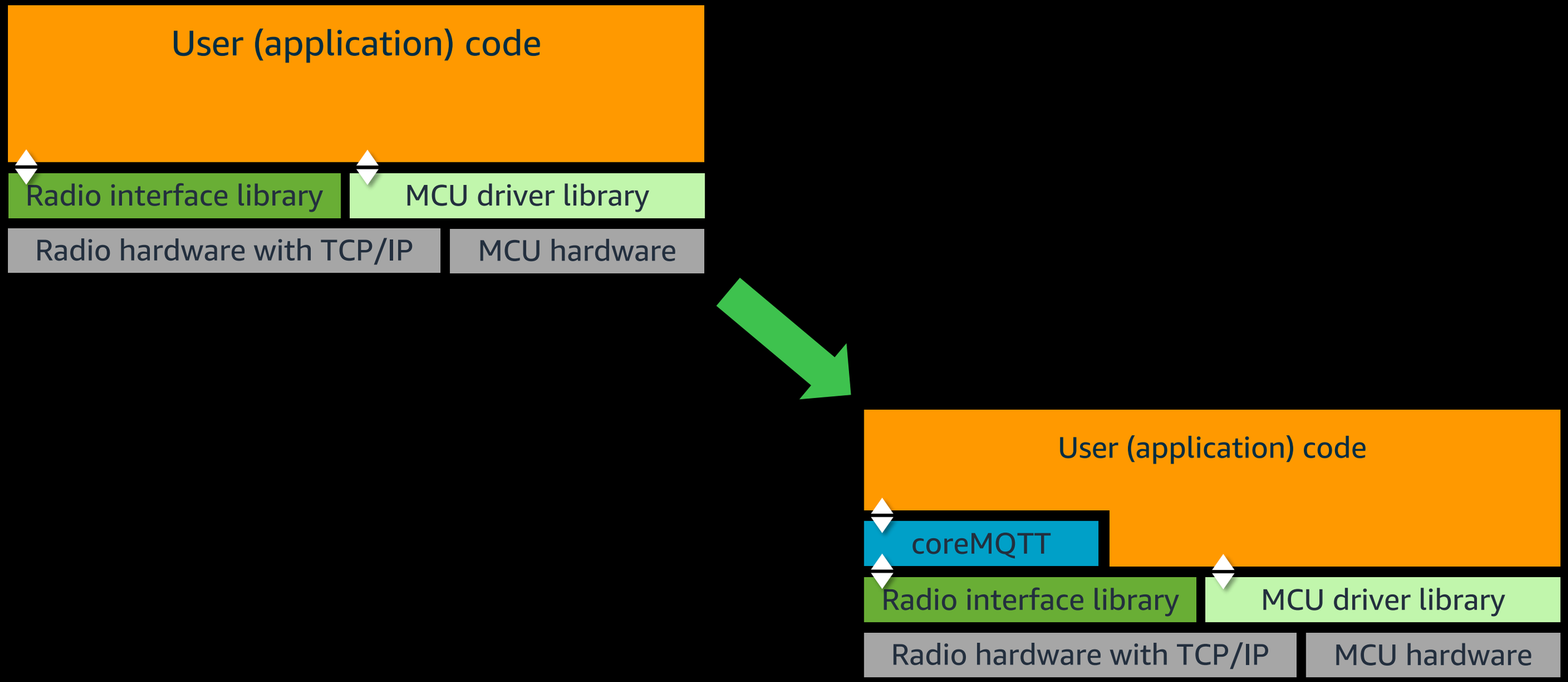
Creating and updating connected applications



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use case 1: Updating brownfield applications



coreMQTT network interface

```c
static MQTTStatus_t prvCreateMQTTConnectionWithBroker( MQTTContext_t * pxMQTTContext,
                                                       NetworkContext_t * pxNetworkContext )
{
    MQTTStatus_t xResult;
    MQTTConnectInfo_t xConnectInfo;
    TransportInterface_t xTransport;

    /* Fill in Transport Interface send and receive function pointers. */
    xTransport.pNetworkContext = pxNetworkContext;
    xTransport.send = my_tls_send_function;
    xTransport.recv = my_tls_recv_function;

    /* Initialize MQTT library. */
    xResult = MQTT_Init( pxMQTTContext, &xTransport, prvGetTimeMs, prvEventCallback, &xBuffer );
    return xResult;
}
```

Single-threaded code example: Publish()

```c
static void prvMQTTPublishToTopic( MQTTContext_t * pxMQTTContext )
{
    MQTTStatus_t xResult;
    MQTTPublishInfo_t xMQTTPublishInfo;

    /* Some fields are not used by this demo so start with everything at 0. */
    memset( ( void * ) &xMQTTPublishInfo, 0x00, sizeof( xMQTTPublishInfo ) );

    /* This demo uses QoS0. */
    xMQTTPublishInfo.qos = MQTTQoS1;
    xMQTTPublishInfo.retain = false;
    xMQTTPublishInfo.pTopicName = mqttexampleTOPIC;
    xMQTTPublishInfo.topicNameLength = ( uint16_t ) strlen( mqttexampleTOPIC );
    xMQTTPublishInfo.pPayload = mqttexampleMESSAGE;
    xMQTTPublishInfo.payloadLength = strlen( mqttexampleMESSAGE );

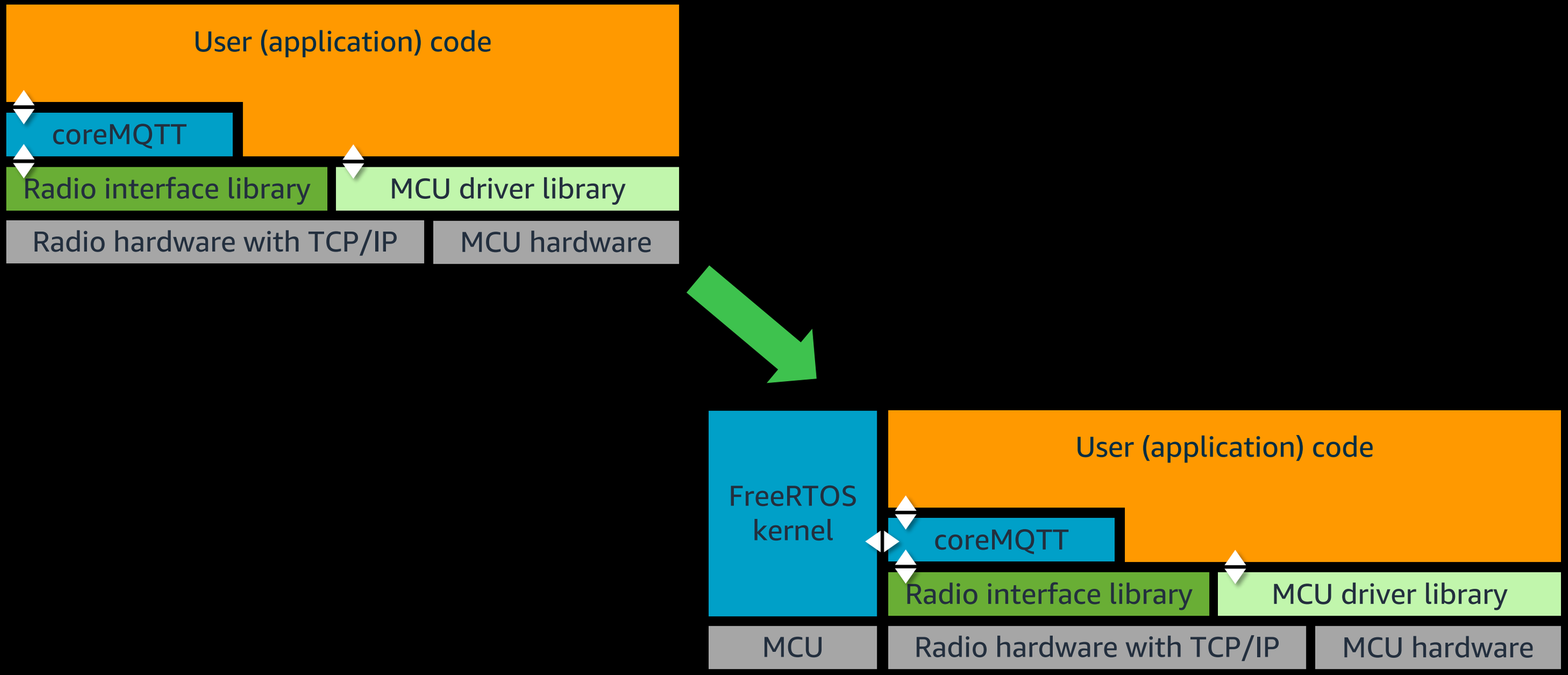
    /* Send PUBLISH packet. Packet ID is not used for a QoS0 publish. */
    xResult = MQTT_Publish( pxMQTTContext, &xMQTTPublishInfo, 0U );
    assert( xResult == MQTTSuccess );
}
```

Single-threaded code example: ProcessLoop()

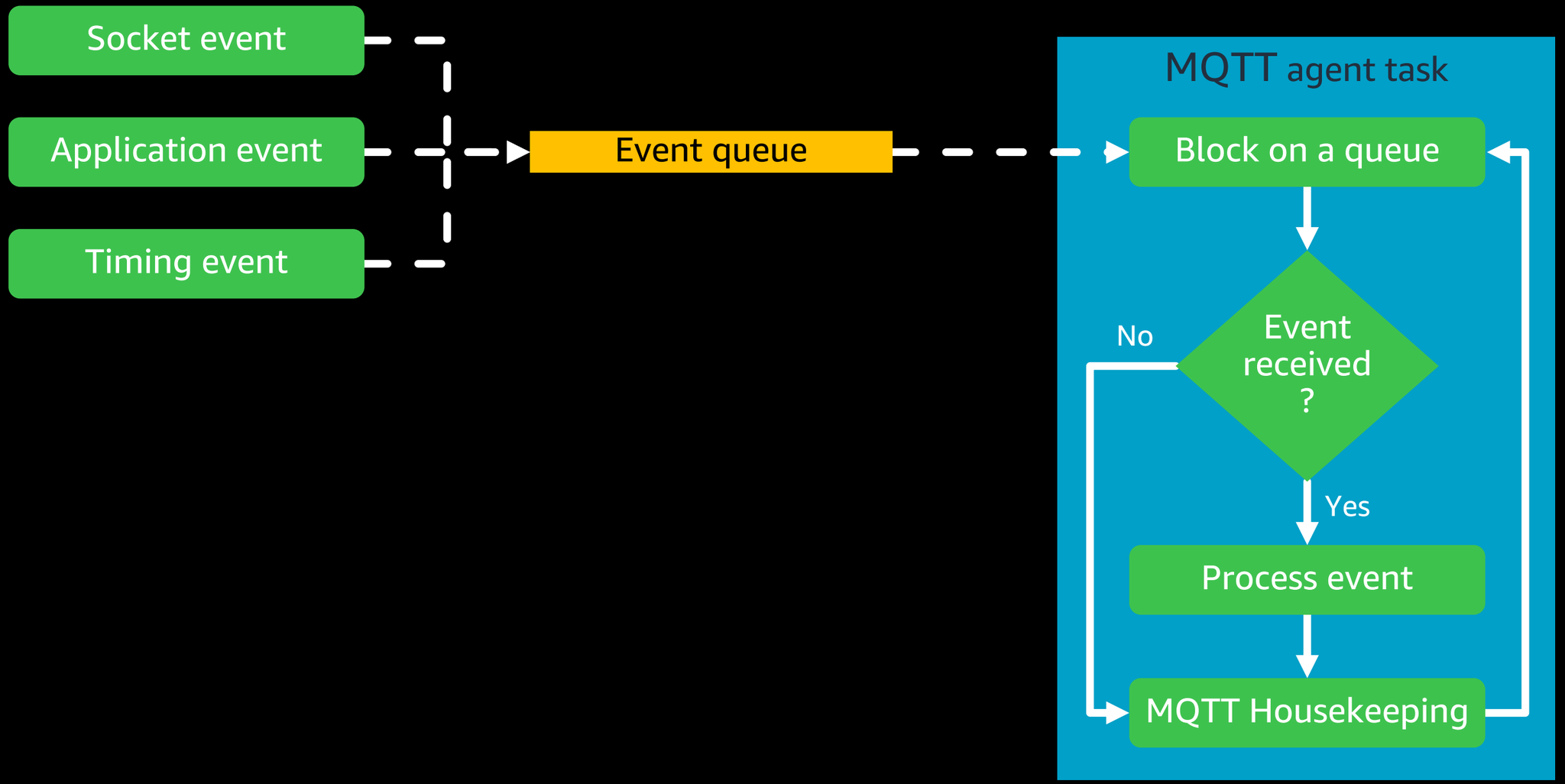
```
/* Publish messages with QoS0, send and process Keep alive messages. */
LogInfo( ( "Publish to the MQTT topic %s.", mqttexampleTOPIC ) );
prvMQTTPublishToTopic( &xMQTTContext );

/* Process incoming publish echo, since application subscribed to the same
 * topic the broker will send publish message back to the application. */
LogInfo( ( "Attempt to receive publish message from broker." ) );
xMQTTStatus = MQTT_ProcessLoop( &xMQTTContext, mqttexamplePROCESS_LOOP_TIMEOUT_MS );
assert( xMQTTStatus == MQTTSuccess );
```

Use case 2: Multithreaded integration



Making coreMQTT thread safe



Structure of the agent (daemon) task

```
static void prvMQTTAgentTask( void *pvParameters )
{
    for( ;; )
    {
        /* Wait with timeout for next command. */
        xCommand.xCommandType = NONE;
        xQueueReceive( xCommandQueue, &xCommand, exampleTICKS_TO_WAIT );

        switch( xCommand.xCommandType )
        {
             case PUBLISH:
                pxPublishInfo = xCommand.pxCmdContext->pxPublishInfo;
                MQTT_Publish( &globalMqttContext, pxPublishInfo, usPacketId );
                break;

            case /* Etc. for all other command types. */
        }

        MQTT_ProcessLoop( &globalMqttContext, 0 );
    }
}
```

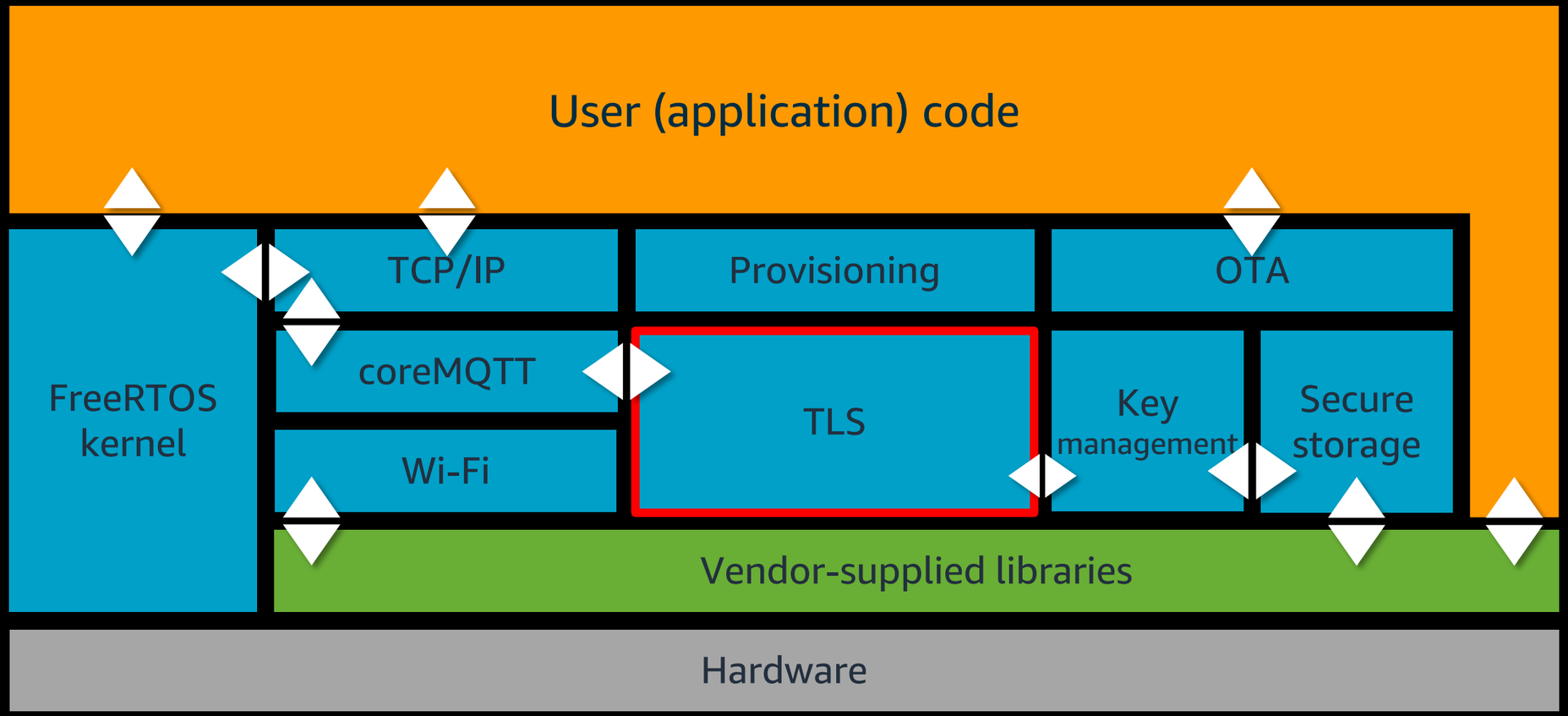
Scaling secure IoT applications



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reference integrations: TLS



Authentication: Proving your identity



- Secure pipe
- Anonymous

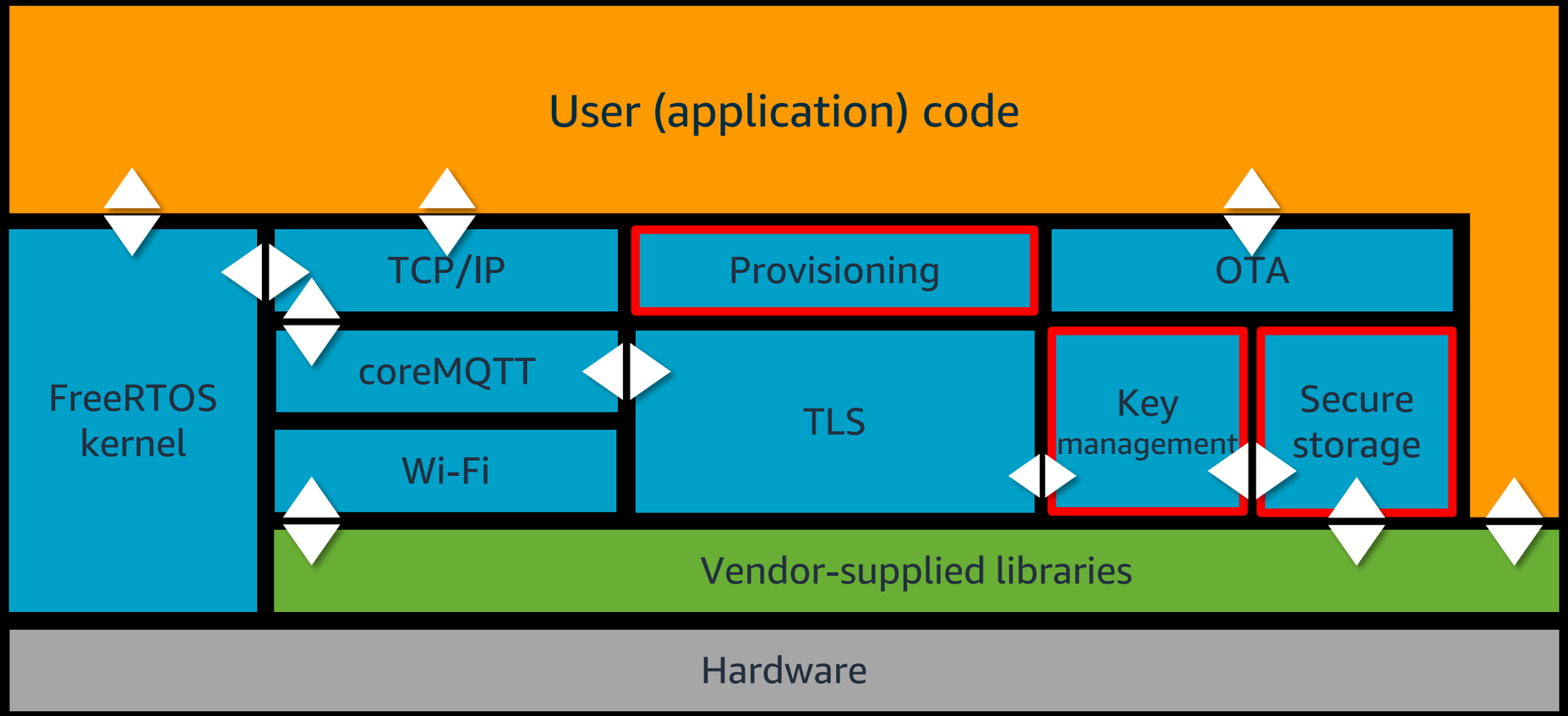
Web server



- Secure bidirectional pipe
- Mutual proof of identity

AWS IoT Core

Reference integrations: Provisioning



corePKCS #11: API to cryptographic tokens

FreeRTOS functionality	Required PKCS #11 API family
Any	Initialize, Finalize, Open/Close Session, GetSlotList, Login
TLS	Random, Sign, FindObject, GetAttributeValue
FreeRTOS+TCP	Random
Over-the-air update	Verify, Digest, FindObject, GetAttributeValue
Provisioning	GenerateKeyPair, CreateObject, DestroyObject, InitToken, GetTokenInfo

How to do this efficiently at AWS IoT scale



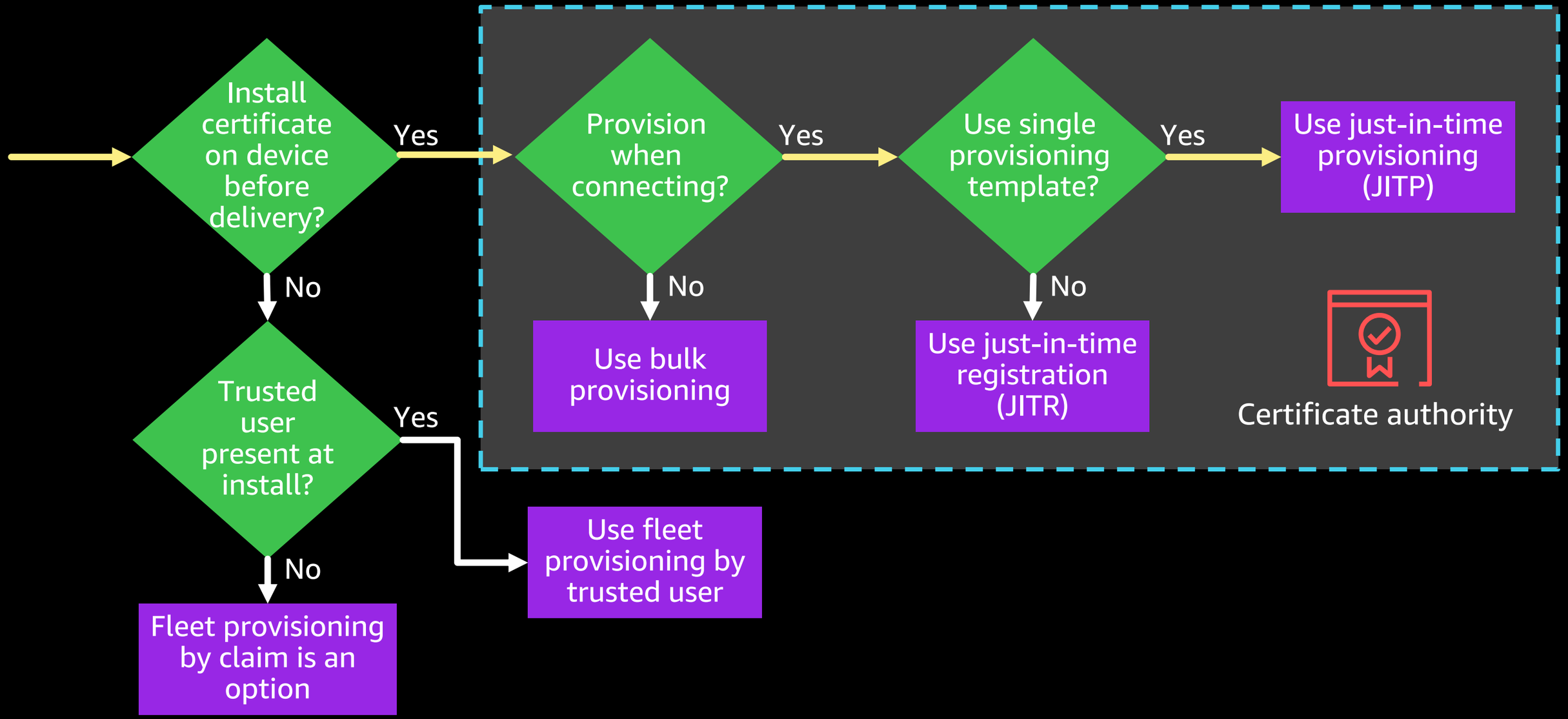
Provisioning IoT devices at scale



AWS re:Invent

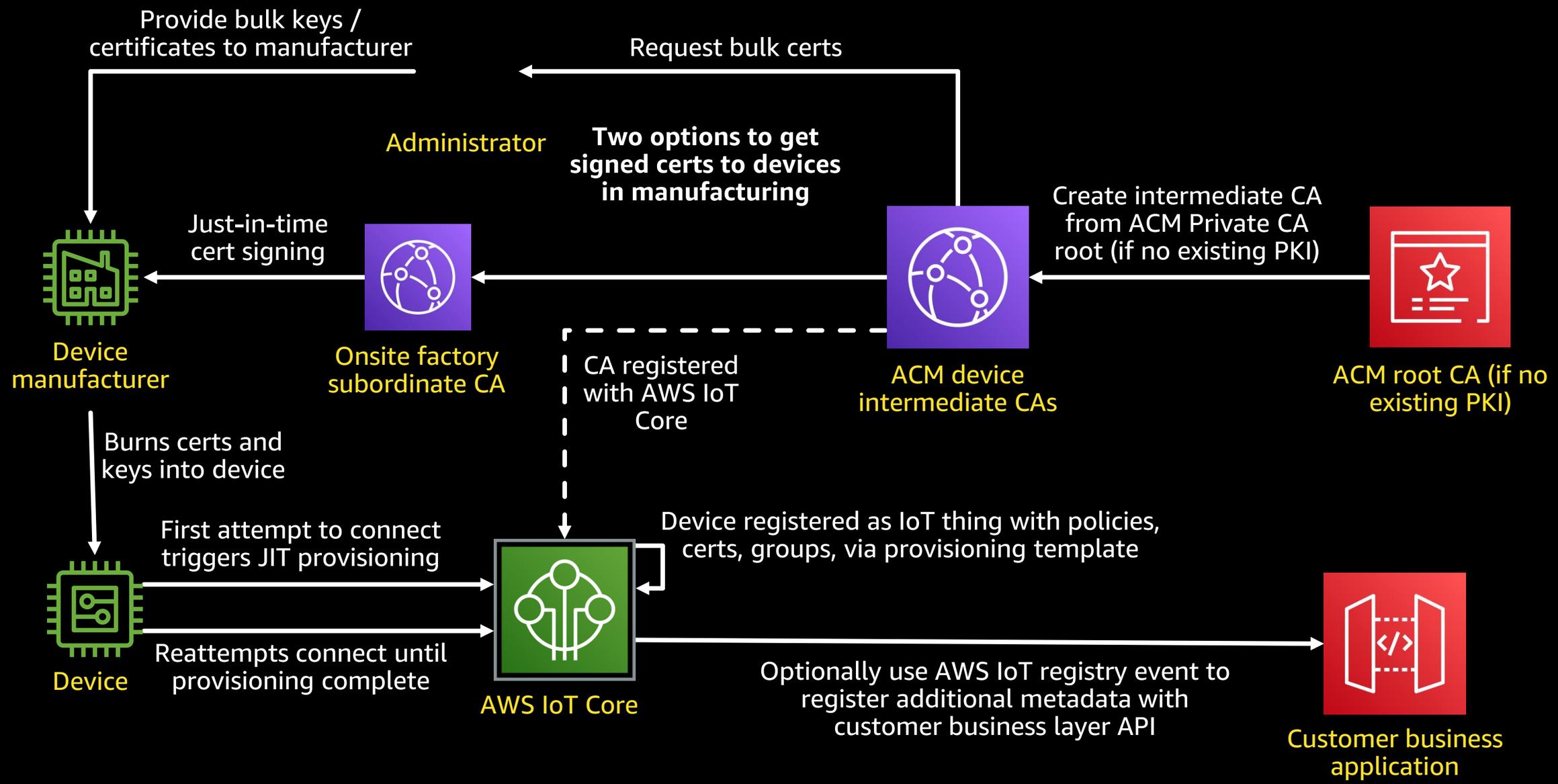
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning and registration options



https://docs.aws.amazon.com/iot/latest/developerguide/iot-provision.html

Just-in-time provisioning



Conclusions

- AWS device software makes secure cloud connectivity easier and faster
- AWS’s stewardship of FreeRTOS has tangible and intangible benefits for all FreeRTOS users
- Device onboarding at AWS scale is challenging; AWS’s provisioning options help overcome that challenge
- https://www.FreeRTOS.org
- https://github.com/FreeRTOS
- https://github.com/AWS

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.