AWS re:Invent

IOT312-R

AWS IoT and industrial automation at Amazon

Rudy Chetty
Senior Solutions Architect
Amazon Web Services

Josh Davis
Software Development Engineer
Amazon

Cameron Fulford
Software Development Engineer
Amazon

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

AWS IoT Greengrass

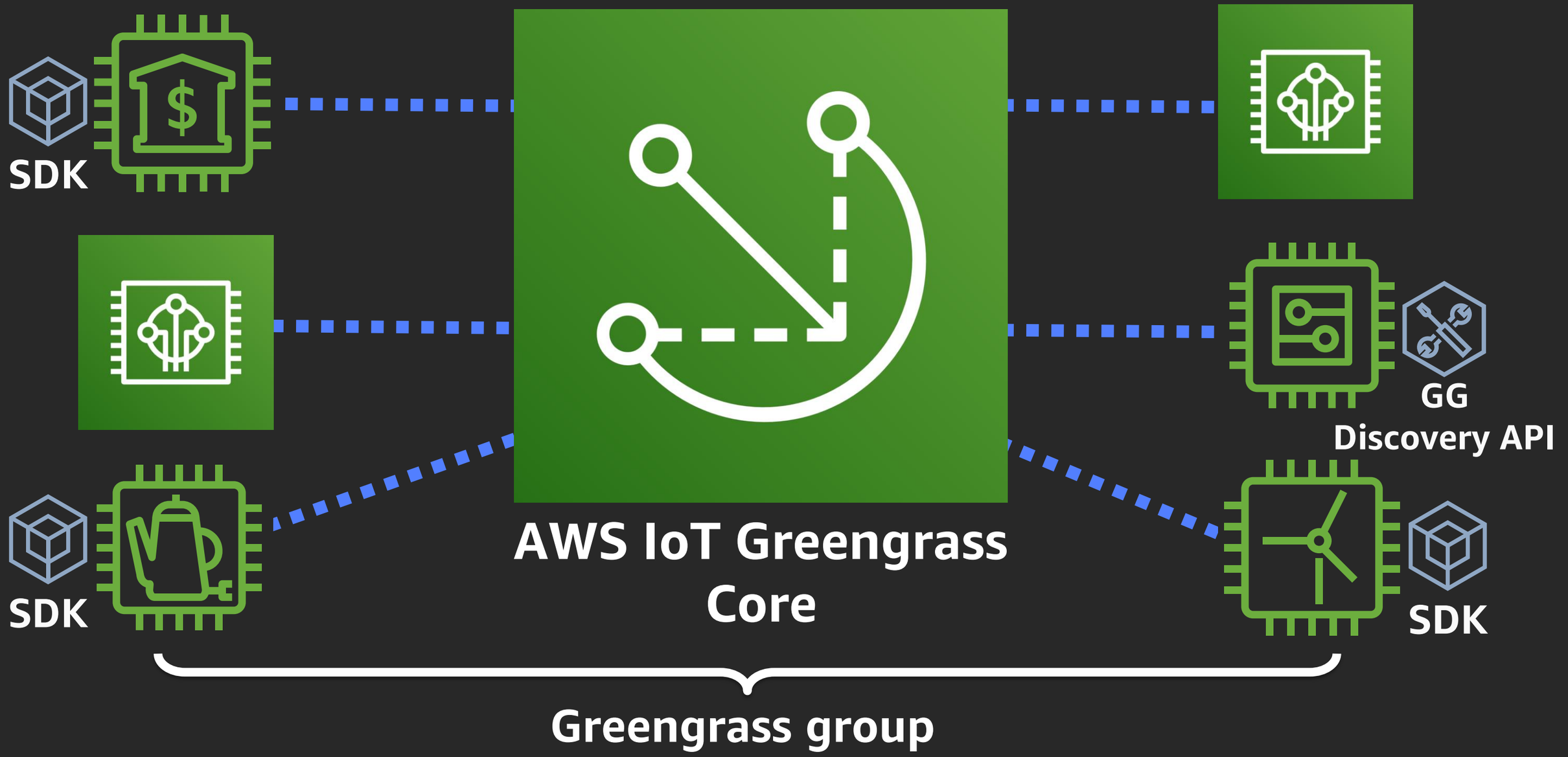
Customer spotlight: Palletizer

Customer spotlight: Edge Work Cells

AWS IoT Greengrass

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Greengrass group



AWS IoT Greengrass messaging

```
aws iot create-thing --thing-name Thingamajig

aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile ...

aws iot attach-thing-principal --thing-name Thingamajig --principal arn:xxxxx

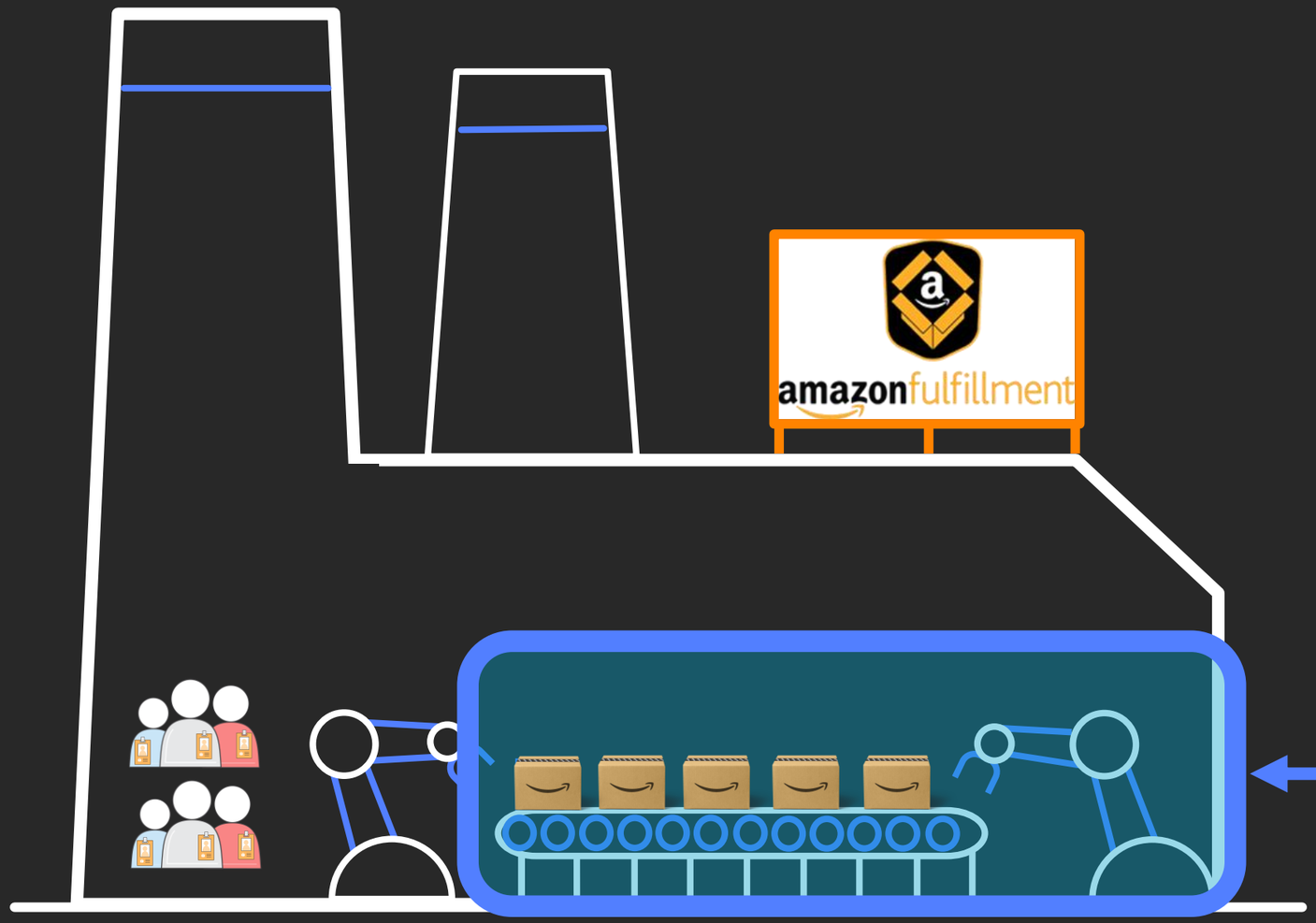
aws greengrass create-group gg-group

...
```



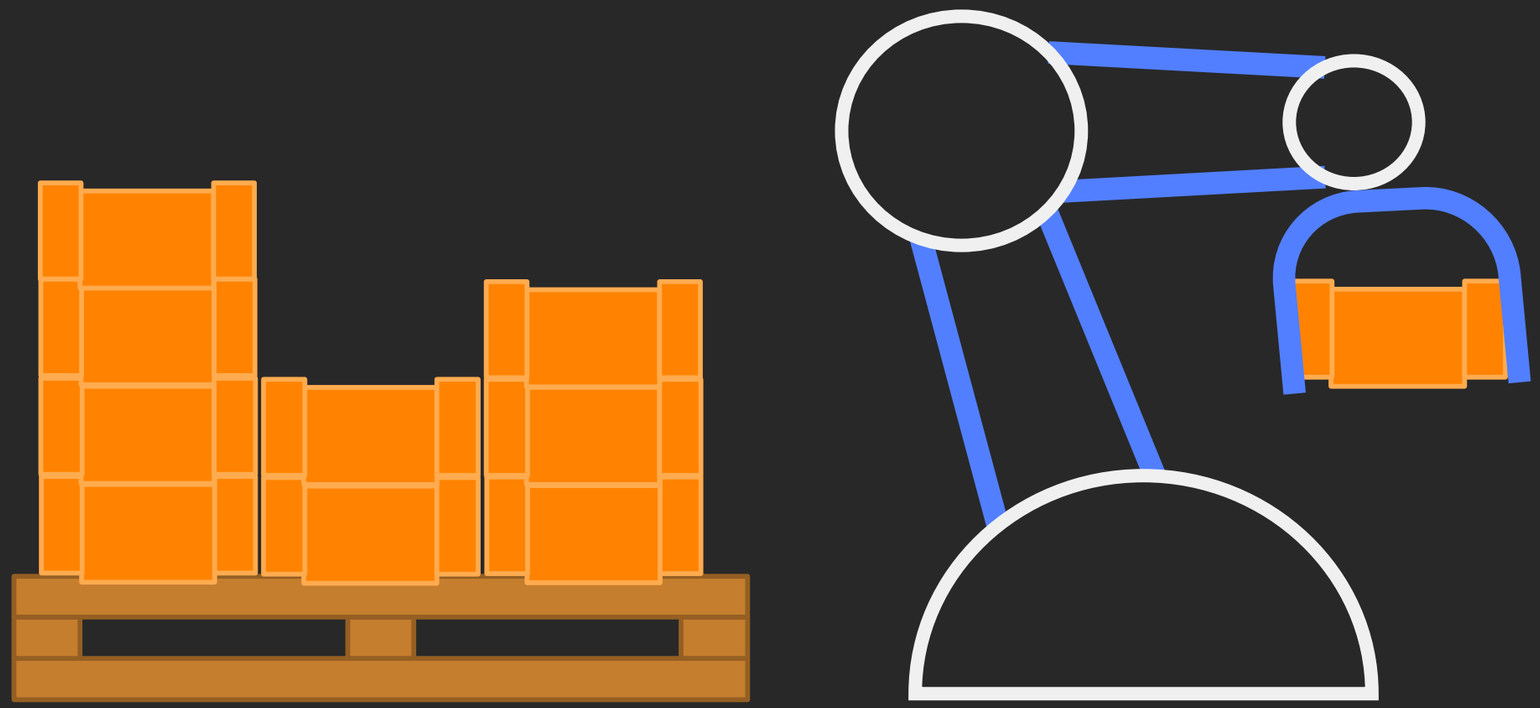
gg-group

What is a work cell?



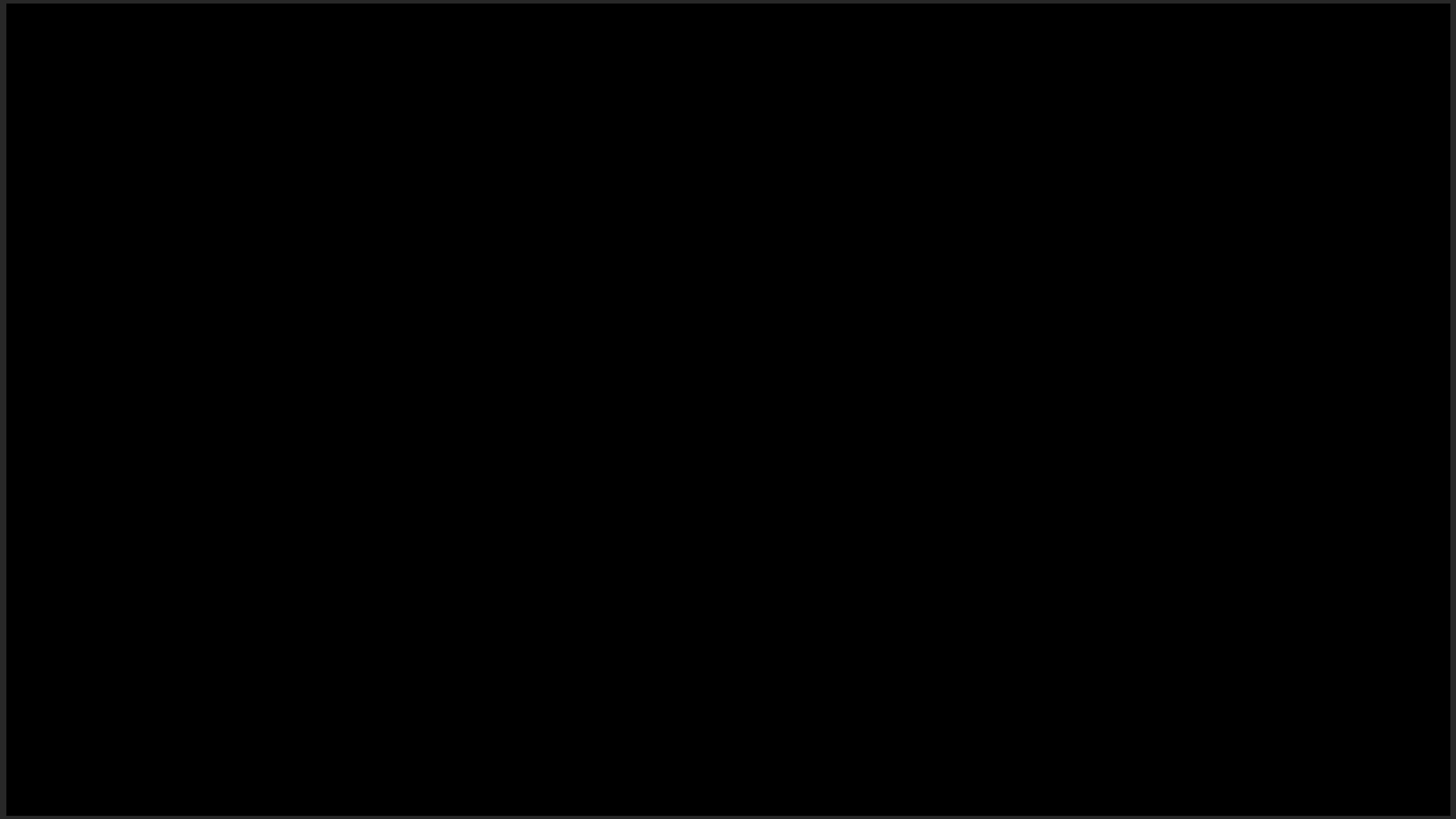
Robotics Work Cell

What is the palletizer?



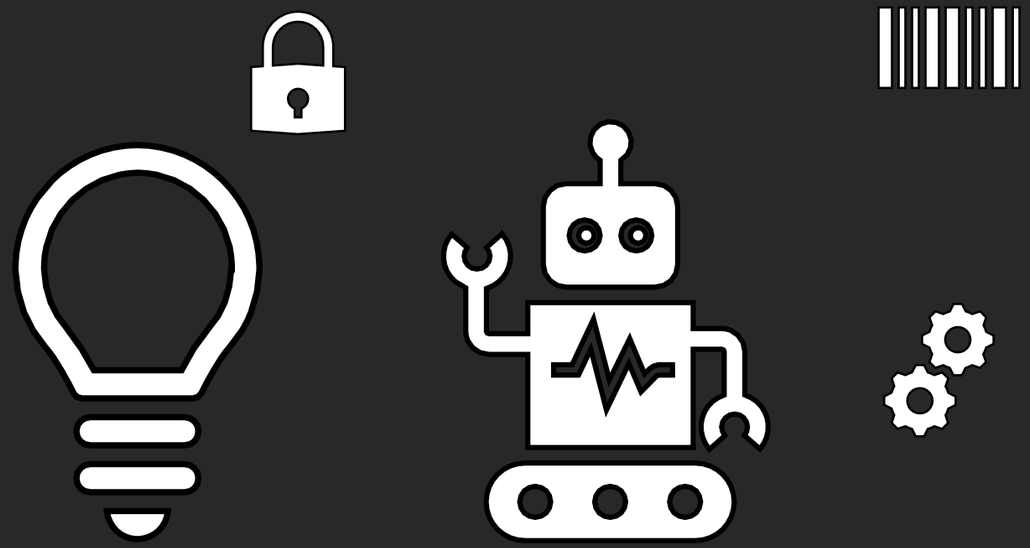
Palletizer

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Background

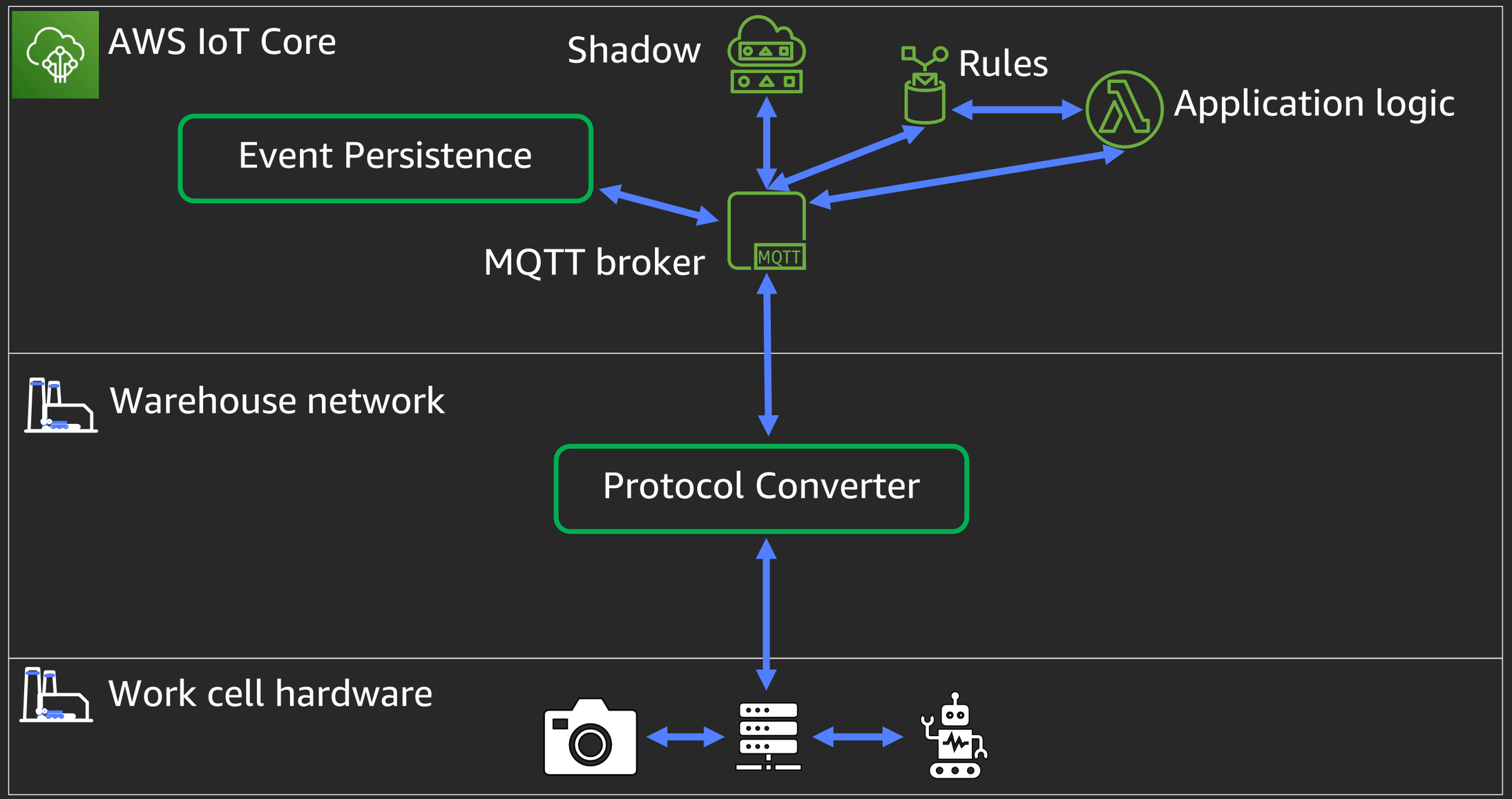
How many engineers does it take to change a lightbulb?



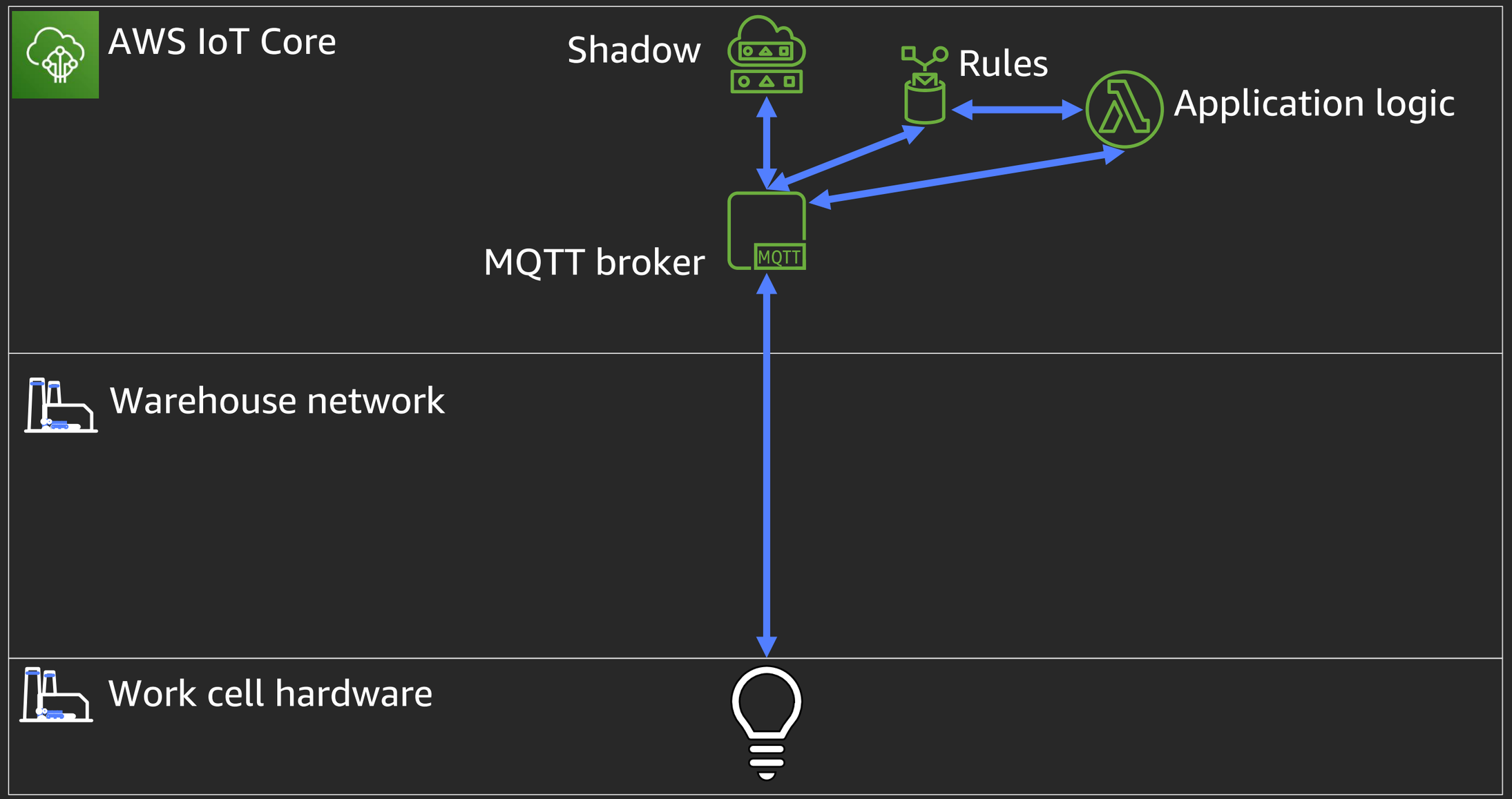
Topics

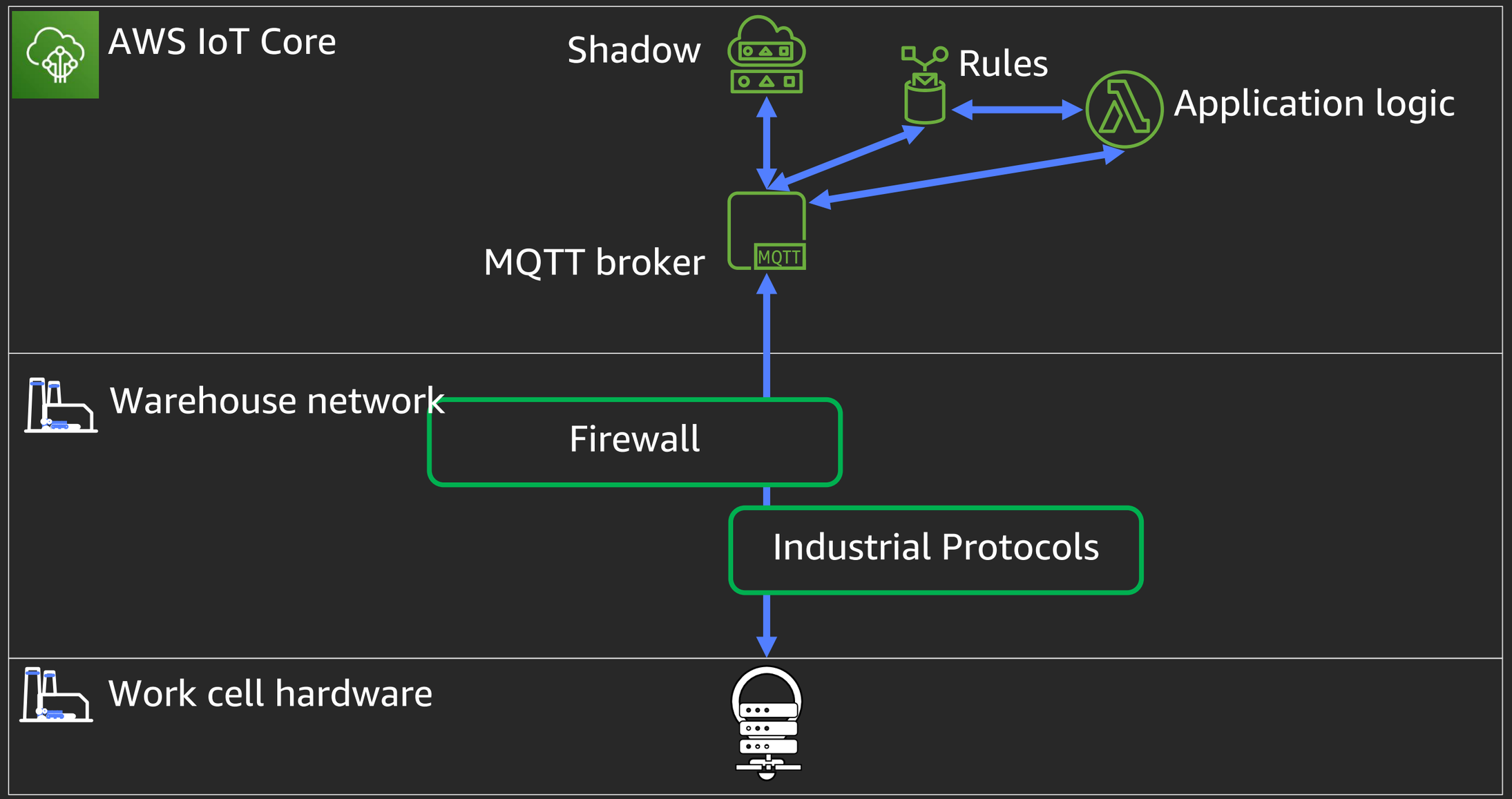
- AWS IoT connectivity and integration
- Orchestration and control
- PLC integration

Summary architecture

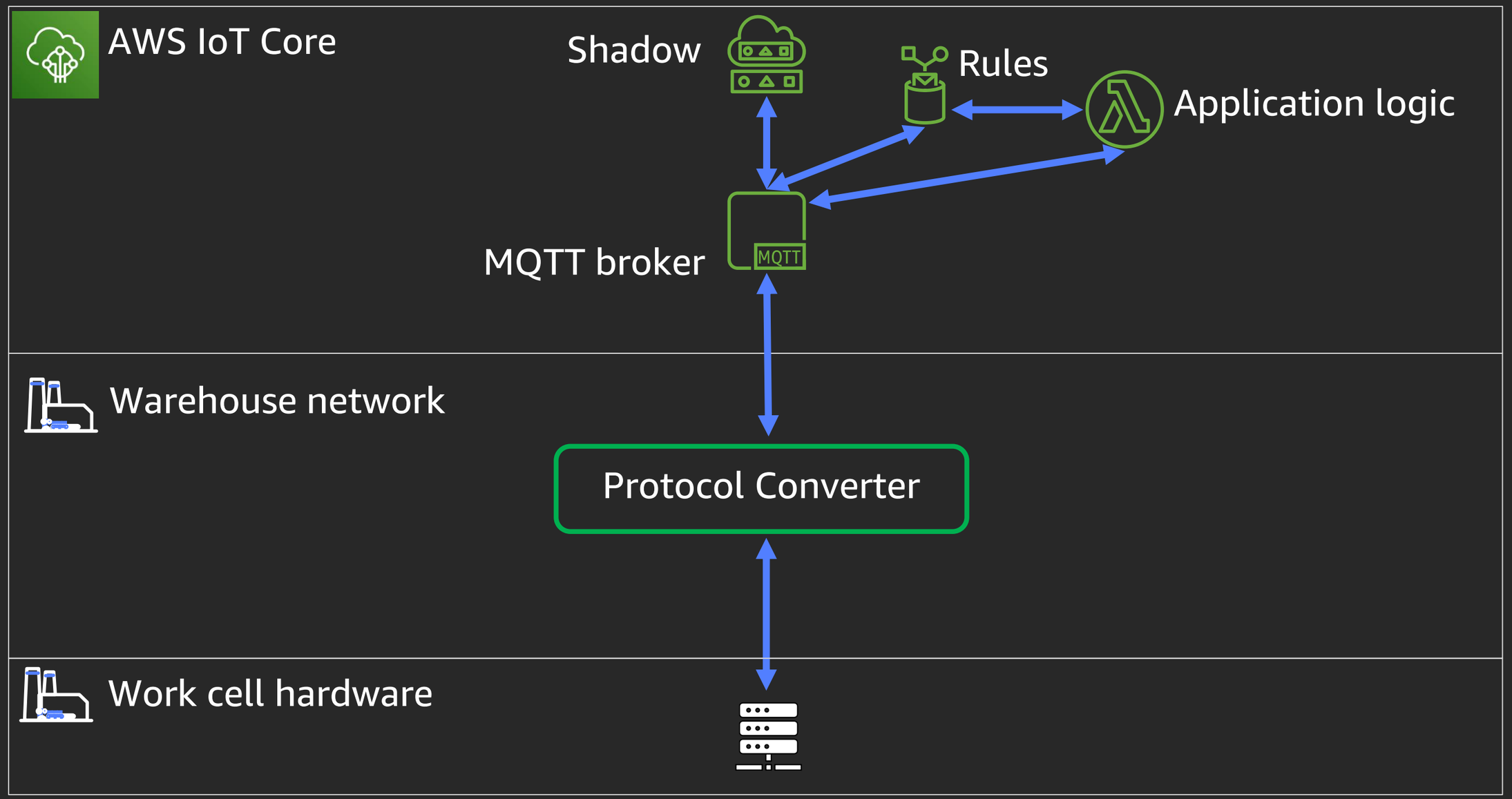


Let there be lightbulb

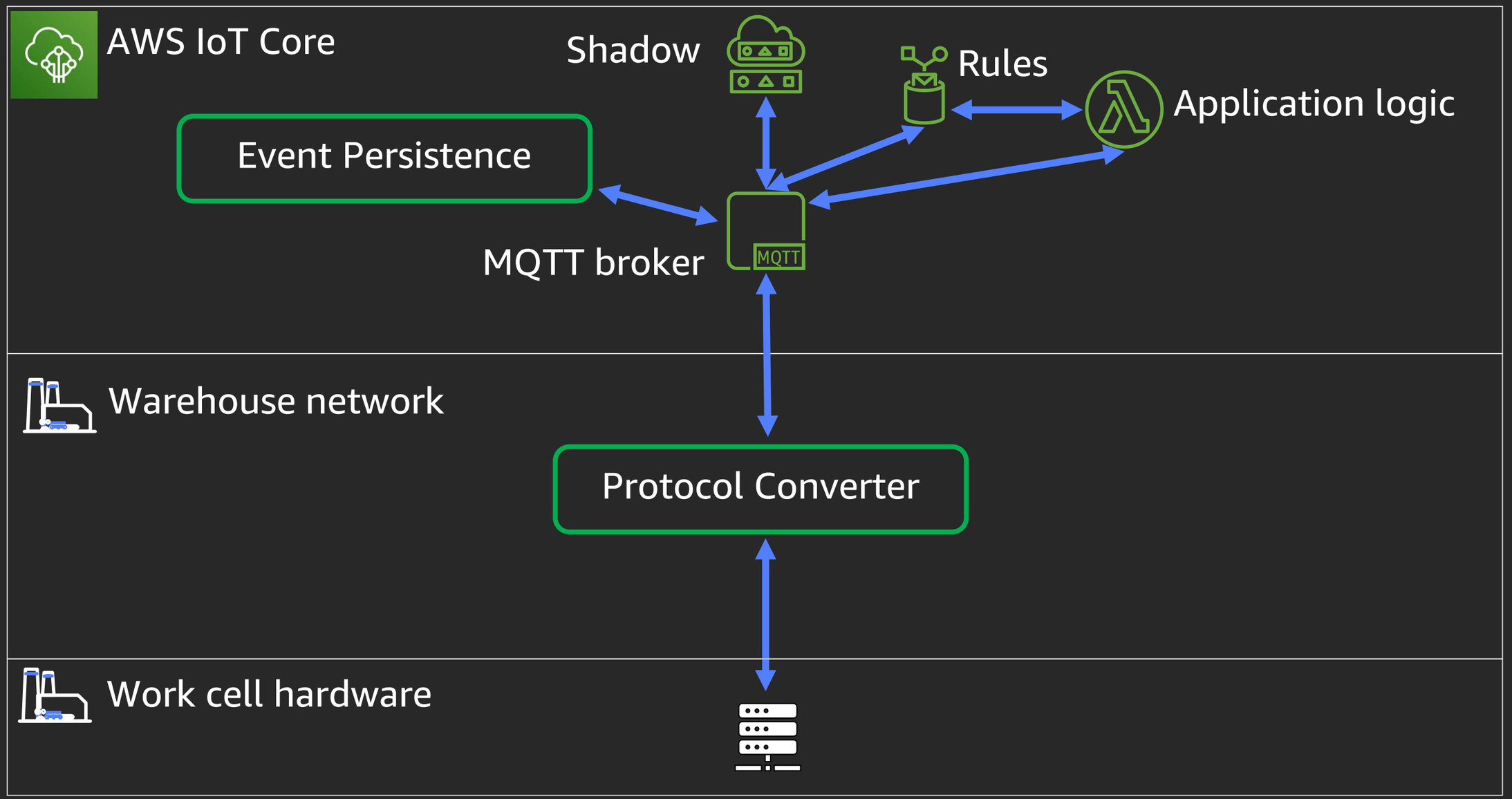



AWS IoT Core
Shadow
Rules
Application logic
MQTT broker
MQTT
Warehouse network
Firewall
Industrial Protocols
Work cell hardware

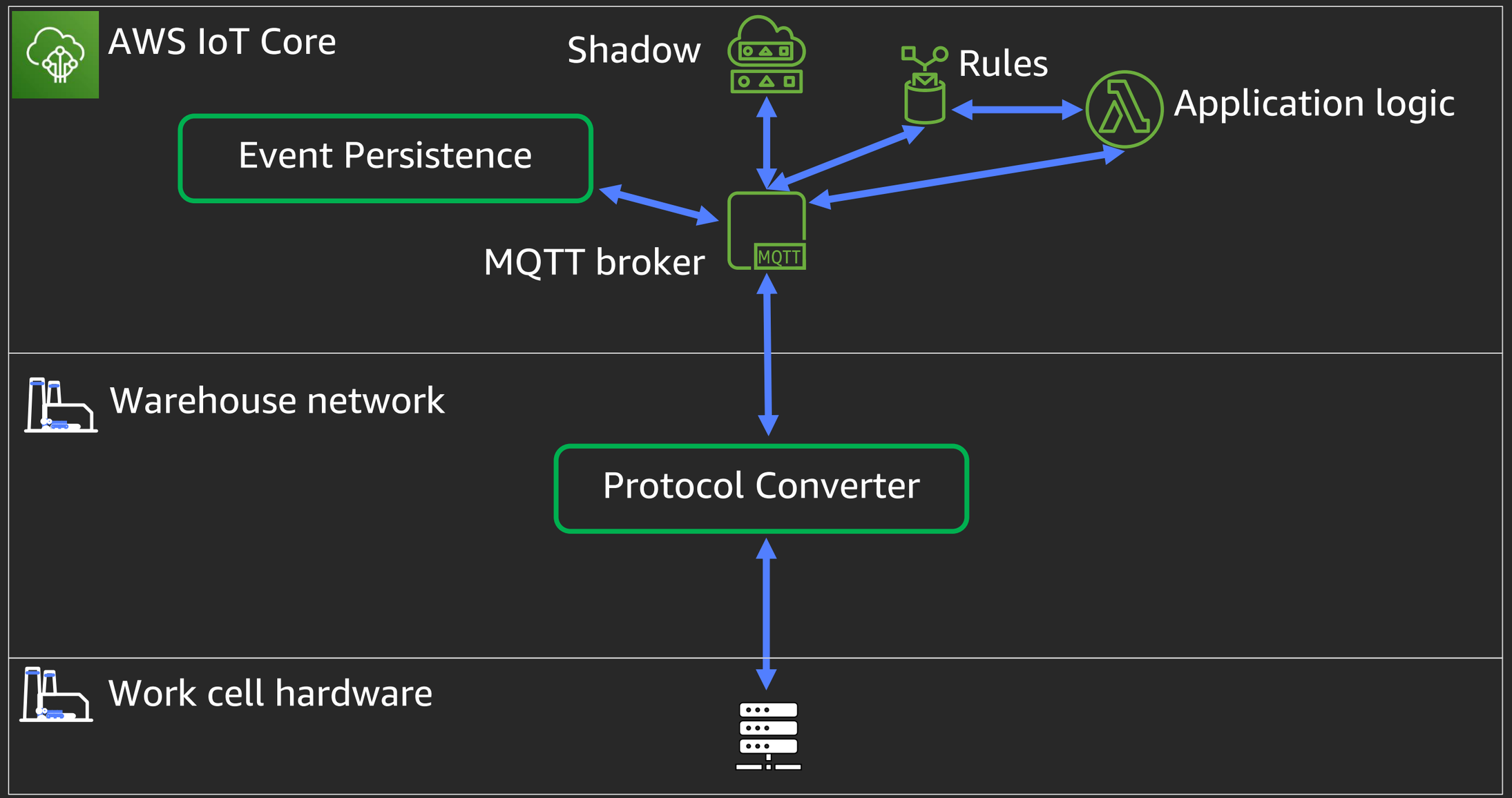
Protocol converter



Event persistence



Brain in the cloud



Translating events to state

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Life of a tote

Arrives at an infeed
Goes from conveyor to arm
Goes from arm to pallet
Pallet is removed
The End

Simple life

Tote arrives at infeed

Tote leaves the infeed

The End

Translator

Custom Protocol:

```
ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000
```

Translator

Custom Protocol:

```
ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000
```

Shadow Update:

```
{ "state":
     { "reported" :
          {
               “INFEED”: “barcode123”
          }
     }
}
```

Translator

Custom Protocol:

```
ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000
```

Shadow Update:

```
{ "state":
     { "reported" :
          {
               “INFEED”: “barcode123”
          }
     }
}
```

Translator

Custom Protocol:

```
ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000
```

Shadow Update:

```
{ "state":
      { "reported" :
            {
                  “INFEED”: “barcode123”
            }
      }
}
```

Translator

Custom Protocol:

```
ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000
```

Shadow Update:

```
{ "state":
      { "reported" :
            {
                  “INFEED”: “barcode123”
            }
      }
}
```

Protocol converter (to cloud)

```
{
  “header”: {
     // header metadata
  },
  “payload” : “ARRIVE,IF01,barcode123,2019-12-14T18:00:00.000”
}
```

Commands to work cell

We could

Shadow Update:

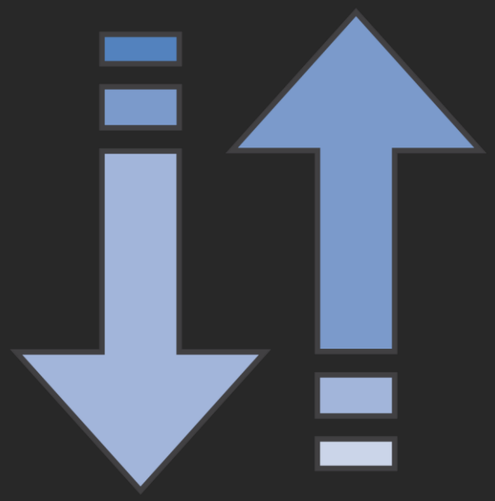
```
{ "state":
      { ”desired" :
            {
                  “INFEED”: null,
                  “DEST”: “barcode123”
            }
      }
}
```

Instead we do this

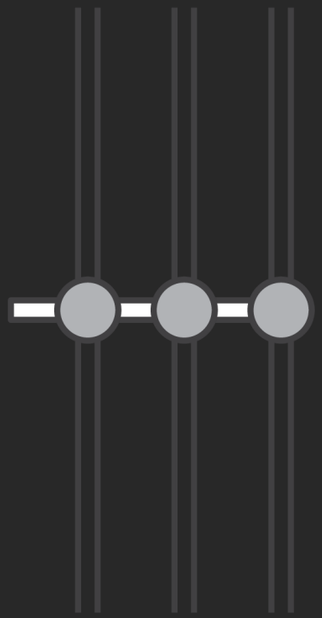
Instead send MQTT:

```
{
   “header”: {
      // header metadata
   },
   “payload” : “MOVE,IF01:barcode123,DEST”
}
```

What are the problems?



Latency

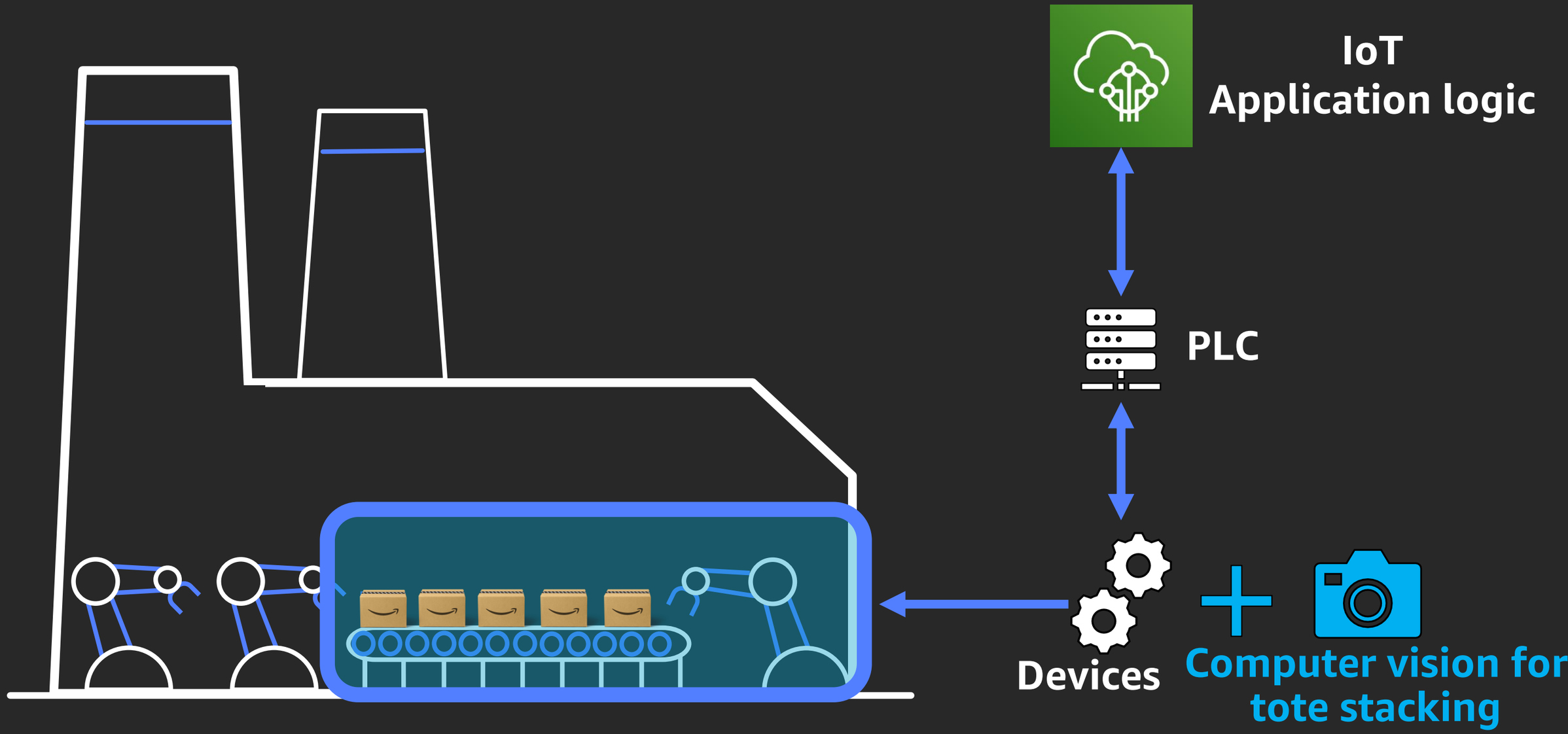


Bandwidth

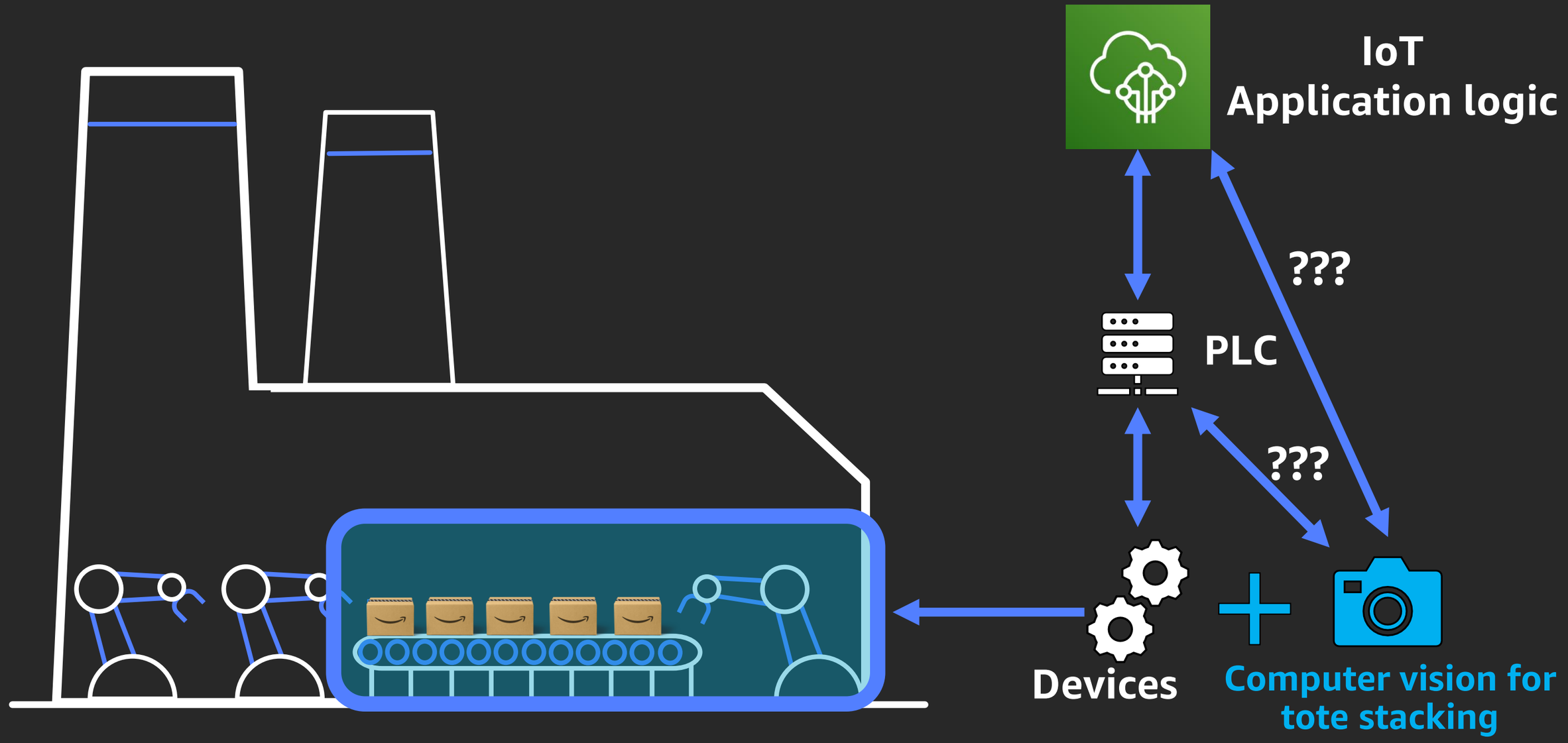


Connectivity

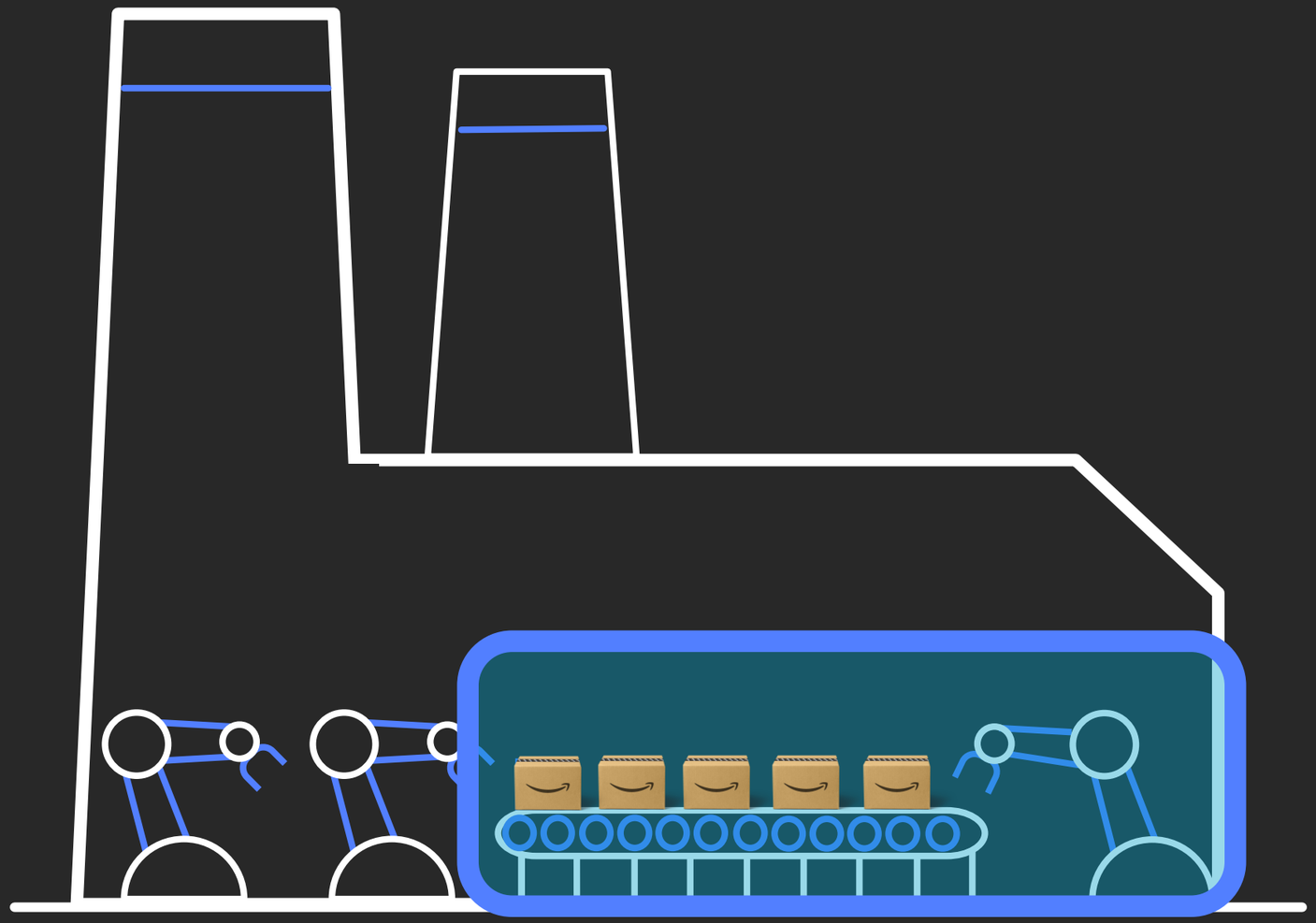
IoT with industrial automation



IoT with evolving industrial automation



IoT with evolving industrial automation





IoT
Application logic



Greengrass
Control Logic



PLC

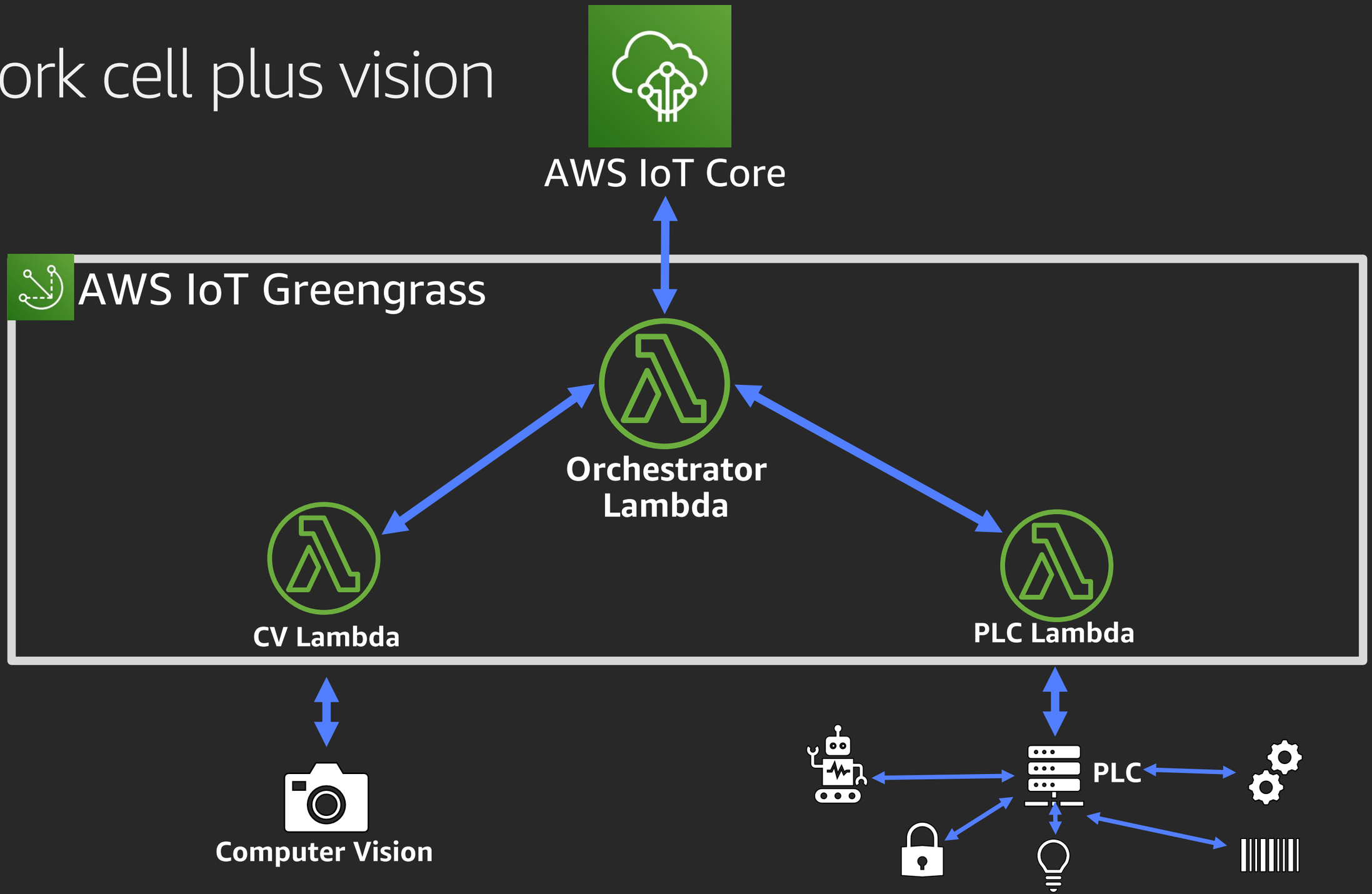


Computer vision for
tote stacking

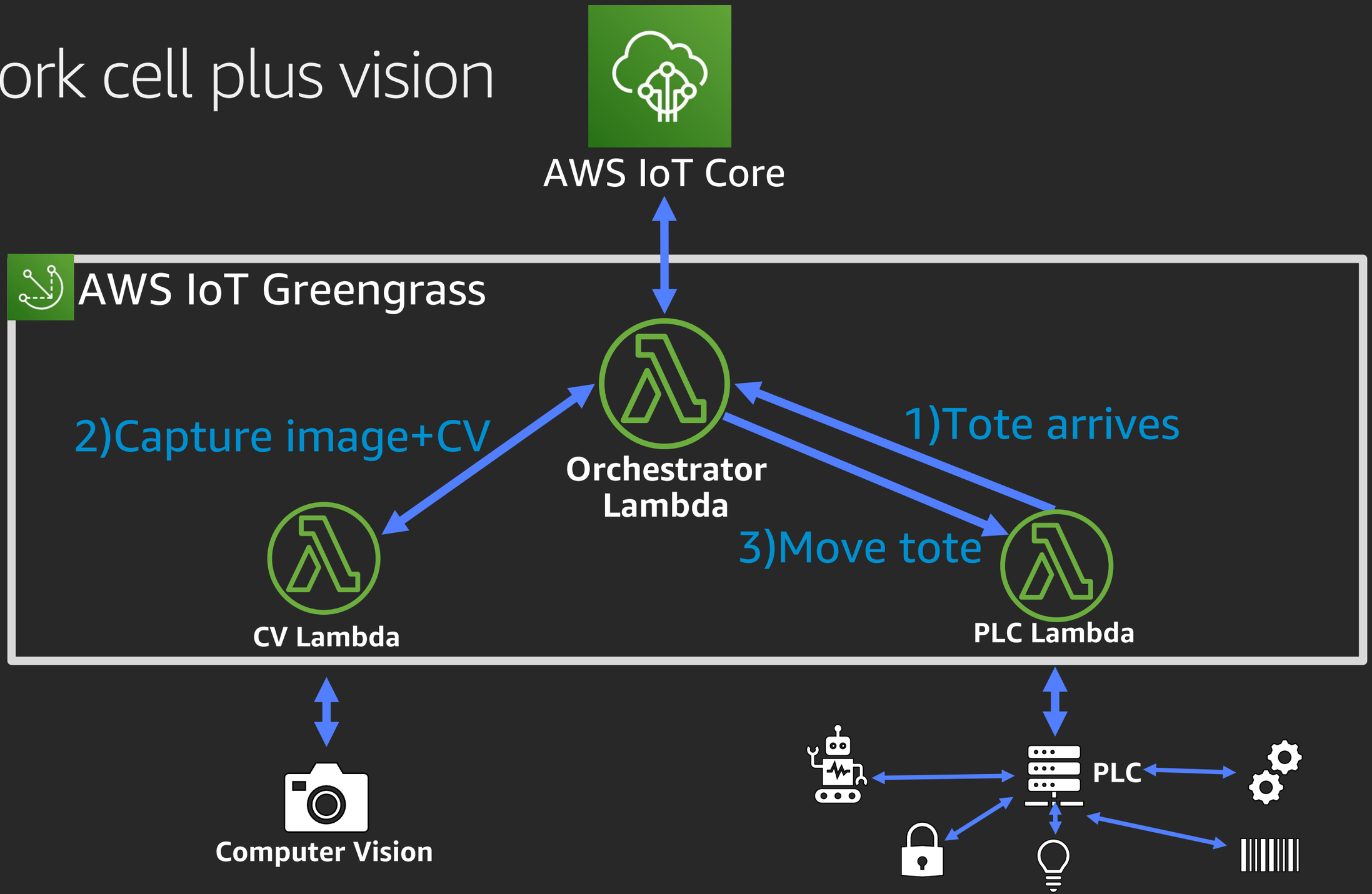


Devices

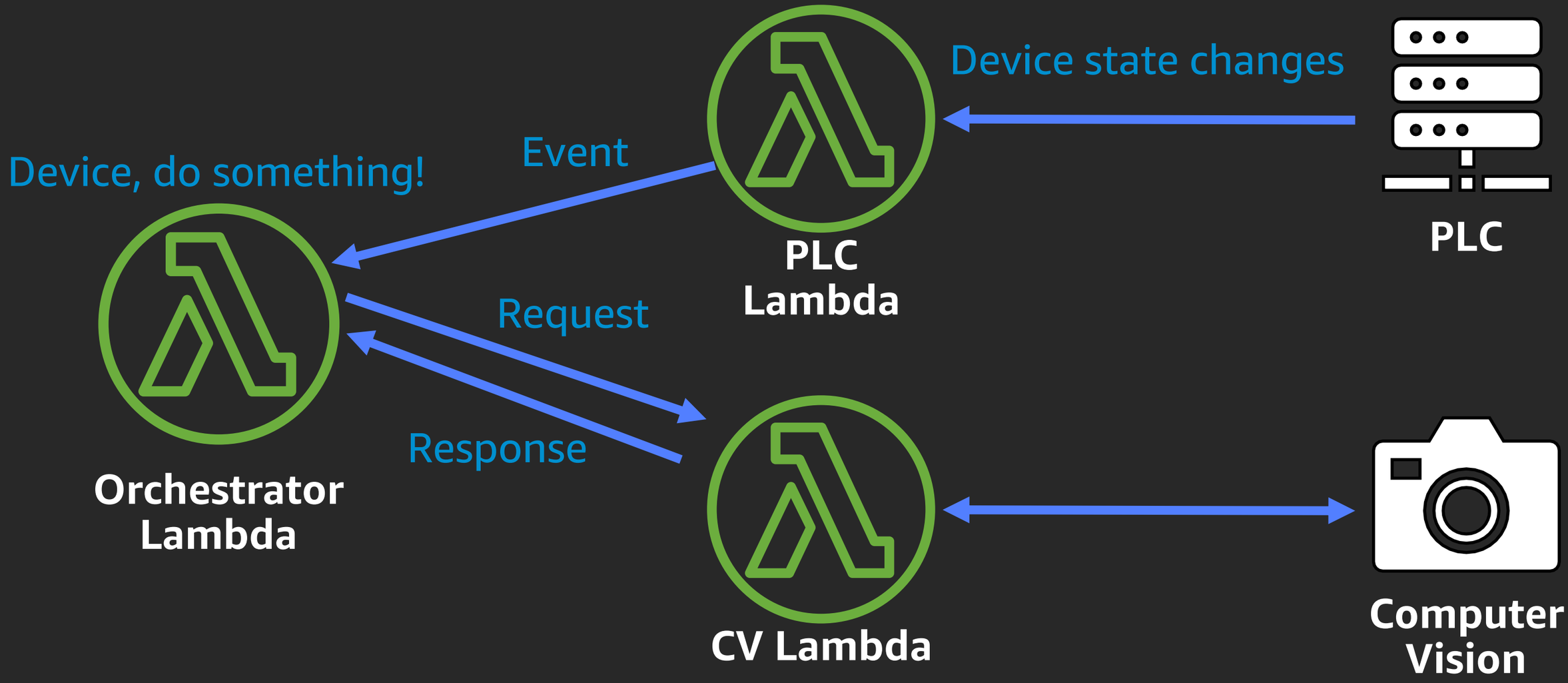
Work cell plus vision



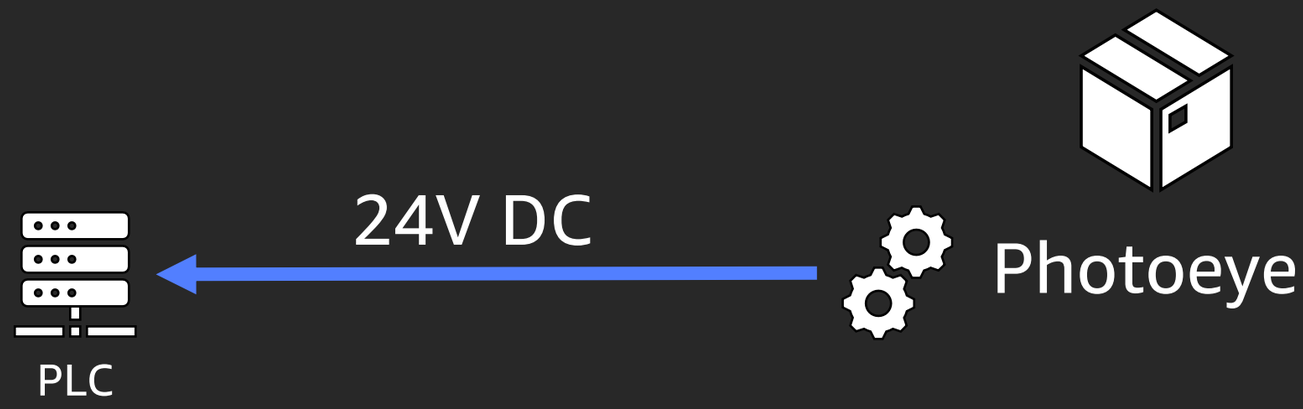
Work cell plus vision



Events and commands

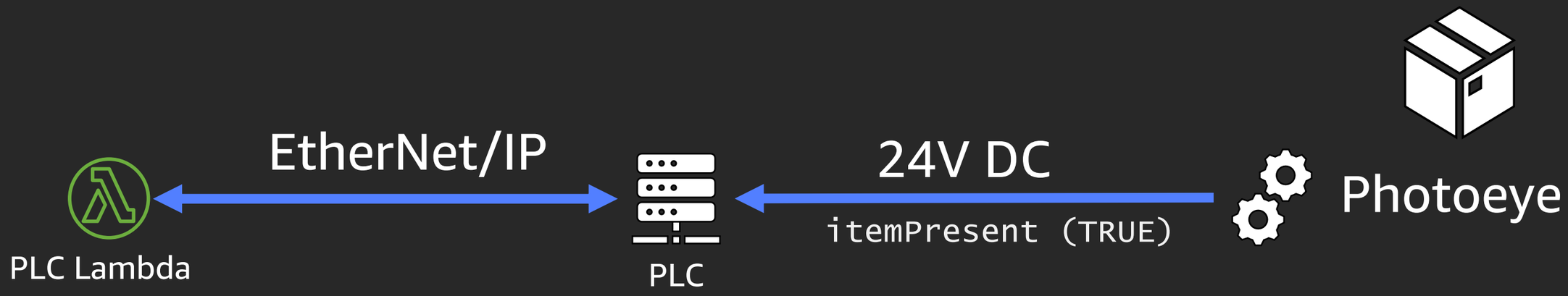


Event: tote arrives



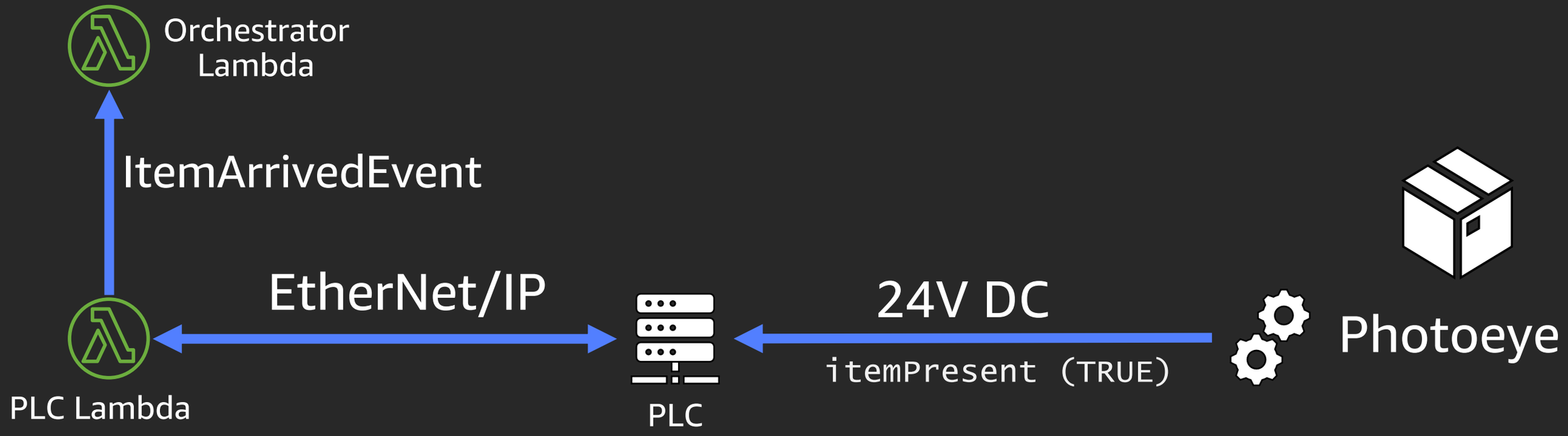

```
DigitalInput.1 -> Debounce filter
Debounce filter output -> itemPresent [BOOL]
```

Event: tote arrives



On a periodic polling loop:
Send EIP ReadTag request to the PLC for all tags we monitor

Event: tote arrives




```
On a periodic polling loop:
Send EIP ReadTag request to the PLC for all tags we
monitor
If itemPresent is TRUE and lastItemPresent is FALSE
  Send new ItemArrivedEvent to the orchestrator
End-If
Set lastItemPresent = itemPresent
```

PLC lambda: sending ItemArrivedEvent

```
String orchestratorArn = …; // Configured


// PLC lambda: on a background polling thread, read tags.
boolean itemPresent = eipClient.readTag(“itemPresent”);
if (itemPresent && !lastItemPresent) {
      // class ItemArrivedEvent extends Message
      ItemArrivedEvent event = new ItemArrivedEvent();
      sendMessageAsync(orchestratorArn, event);
}
lastItemPresent = itemPresent;
```

Sending the message

```
// Base64 encoded JSON context, e.g., { “custom”: { “isMessage”: “true” }}
static final String MESSAGE_CONTEXT = …;
sendMessageAsync(String serviceArn, Message message) {
        String payload = objectMapper.writeValueAsString(message);
        InvokeRequest invokeRequest = new InvokeRequest()
                .withClientContext(MESSAGE_CONTEXT)
                .withInvocationType(InvocationType.Event)
                .withFunctionArn(serviceArn)
                .withPayload(payload);
        // Async “event style” invocation. Does not wait for a response.
        lambdaClient.invoke(invokeRequest);
}
```

Orchestrator lambda: receiving event

```
// Orchestrator lambda handler

lambdaHandler(InputStream inputStream, OutputStream outputStream, Context
context) {

      // Check for isMessage = true in the context

      if (isMessage(context)) {

            Message message = objectMapper.readValue(inputStream,
Message.class);

            messageHandler.handleMessageAsync(message);

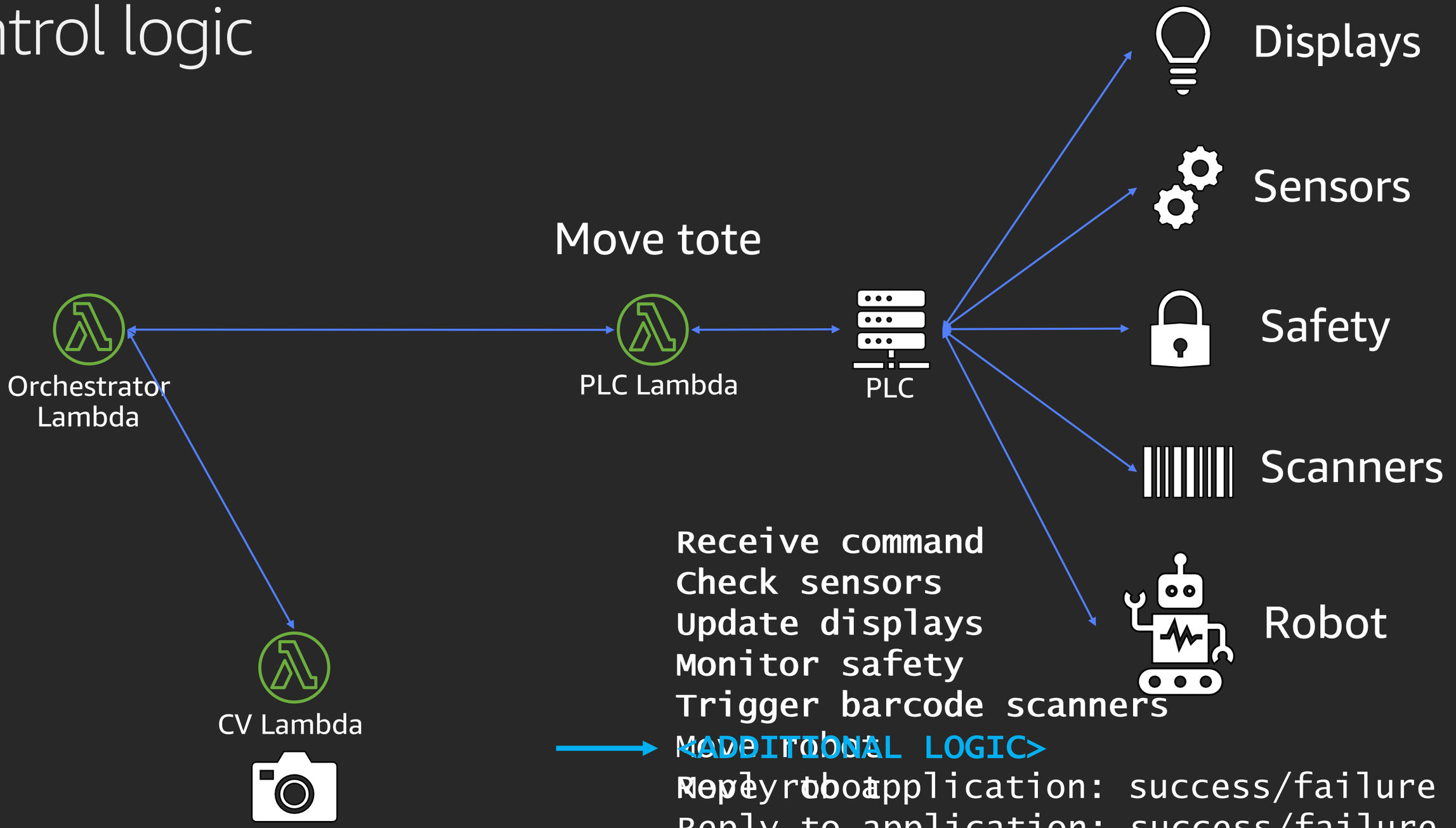
      } else {

            // Handle differently

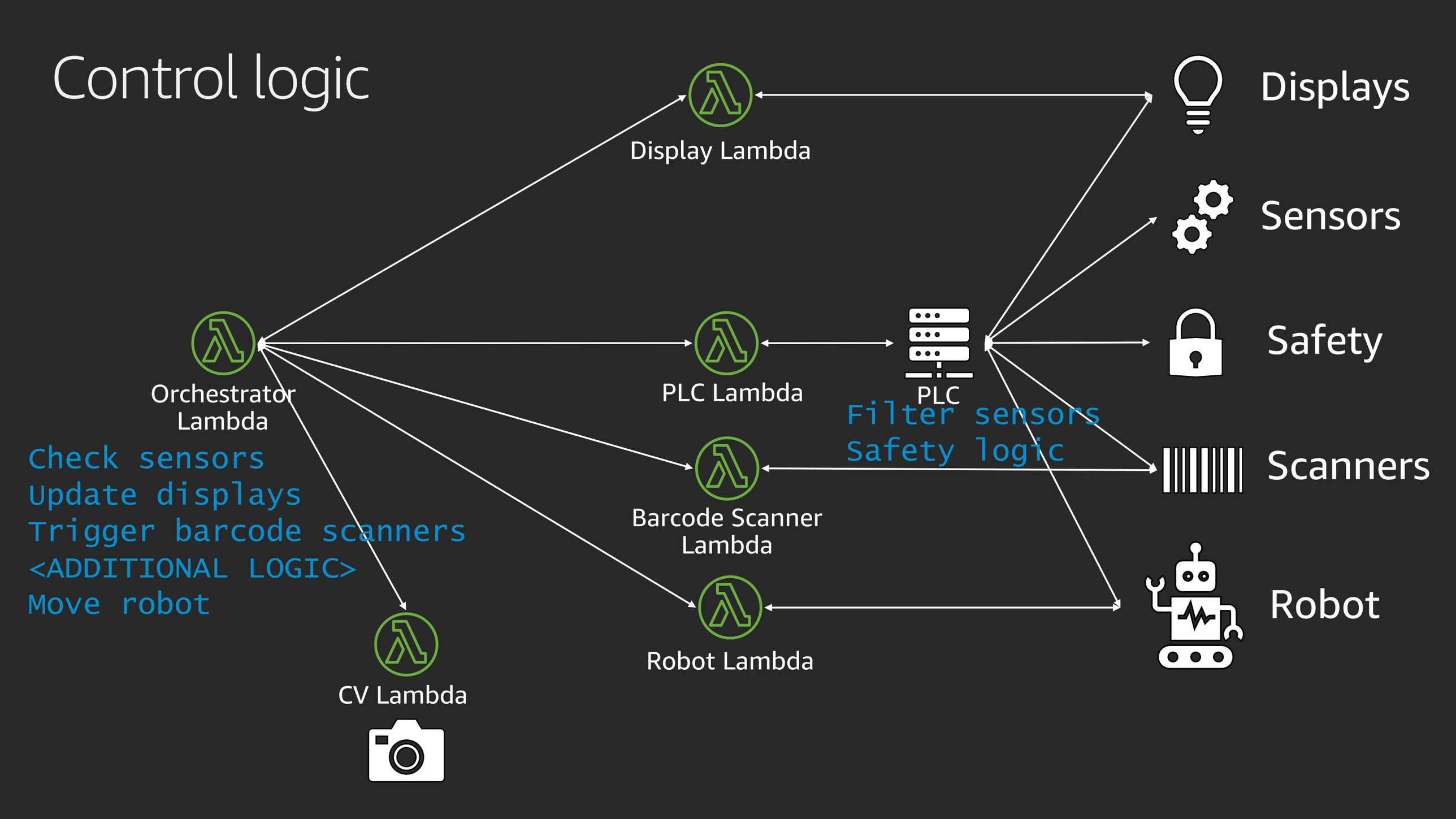
      }

}
```

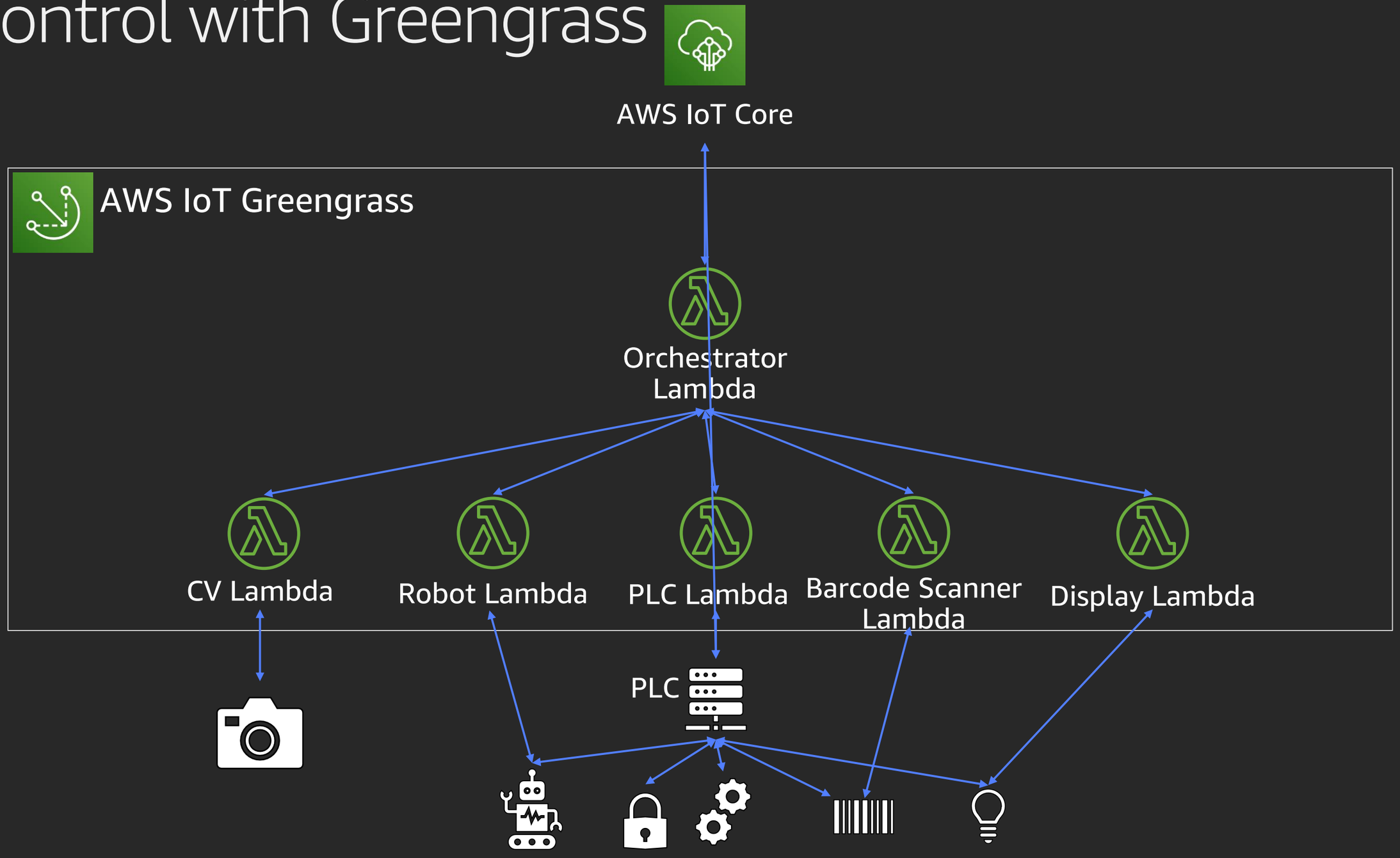
Control logic



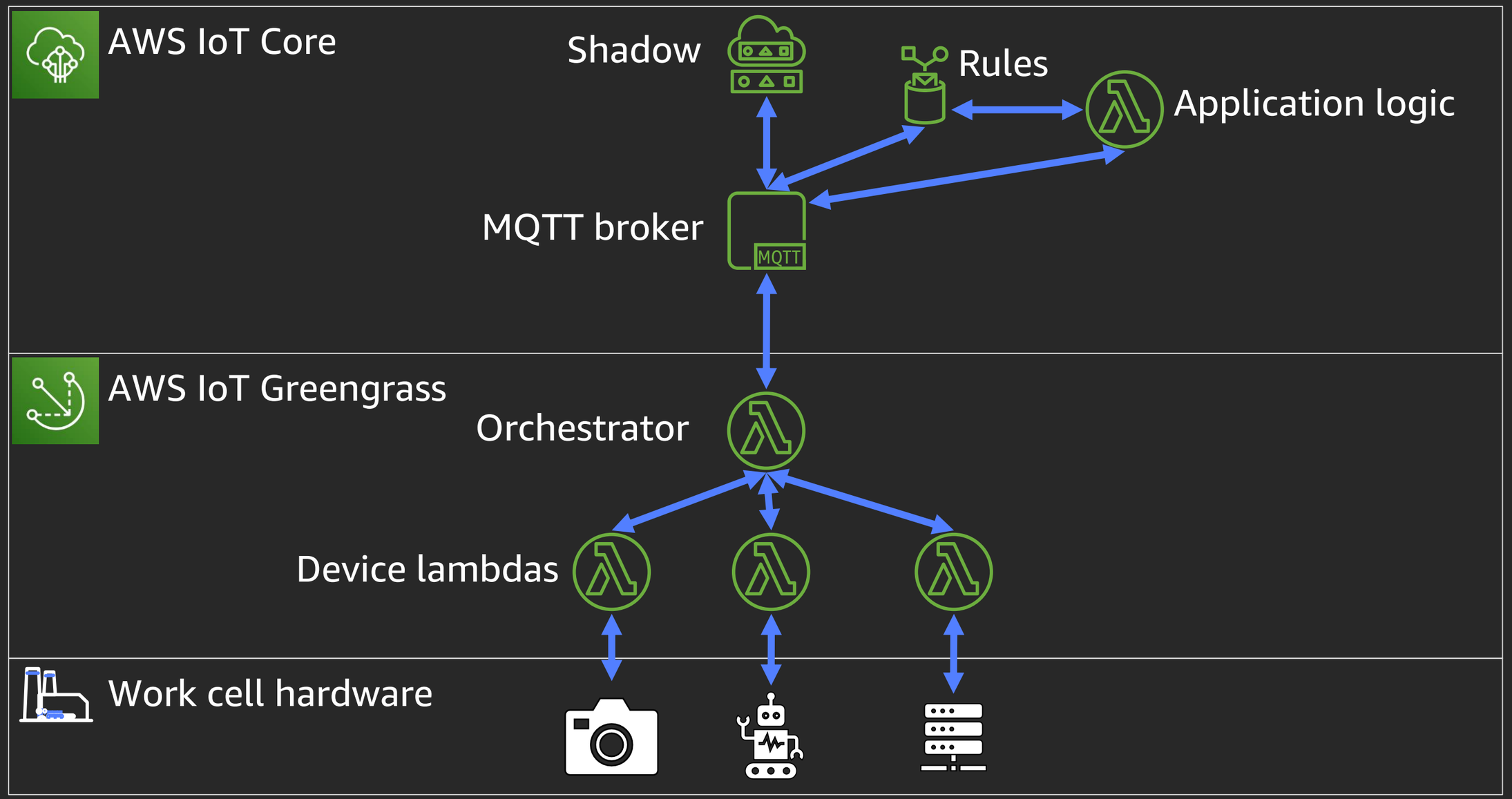
Control logic



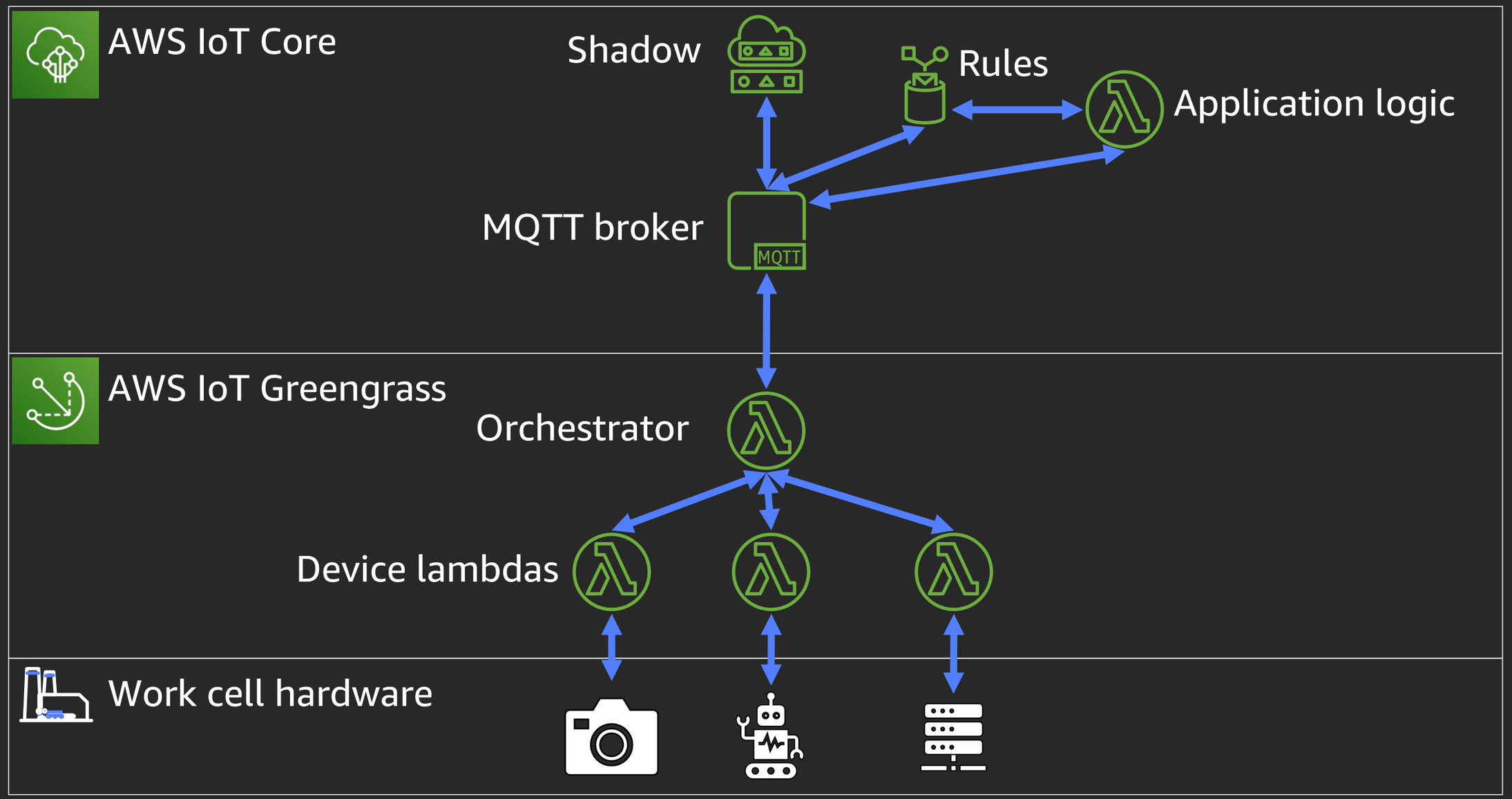
Control with Greengrass



Summary architecture



Summary architecture



Resources

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resources

Amazon Day One blog: *Ever wonder how people and robots team up on your Amazon order?*

https://blog.aboutamazon.com/operations/ever-wonder-how-people-and-robots-team-up-on-your-amazon-order

Track and trace solution

https://aws.training/learningobject/video?id=16652

IoT workshop

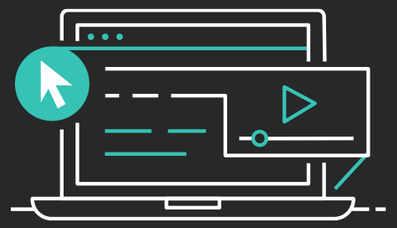
https://tinyurl.com/iot-rudy

Learn IoT with AWS Training and Certification

Resources created by the experts at AWS to help you build IoT skills



Take the free digital curriculum, Internet of Things (IoT) Foundation Series, to build IoT skills and work through common scenarios



25+ additional free digital courses cover topics related to IoT, including:

- AWS IoT Core
- AWS IoT Greengrass
- AWS IoT Analytics
- AWS IoT Device Management
- AWS IoT Events

Visit the Learning Library at https://aws.training

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.