Kubernetes EDGE

Kubernetes
EDGE

Taming Unbounded Resources with Node Feature Discovery

Mark Abrams

What is Node Feature Discovery (NFD)

Kubernetes EDGE

- NFD is simply a fancy labeler
- K8s node labels

Why do I need a labeler

- Mapping workloads to appropriate resources

Resources

Workloads

Why do I need a labeler

- Because we are delivering cattle not pets
- Volume - a large number of devices
  - Efficiency
  - Accuracy
- Resources at the Edge
  - Are not just CPU and RAM
  - Are Unbounded

Unbounded Resources?

- K8s is designed to schedule workloads against resources
- In the data center (first class features of k8s)
  - CPU
  - RAM
  - GPU
- At the edge (not schedulable by default)
  - Unlimited number of I/O devices are possible
    - Sensors
    - Actuators

w/o NFD - Childs Play

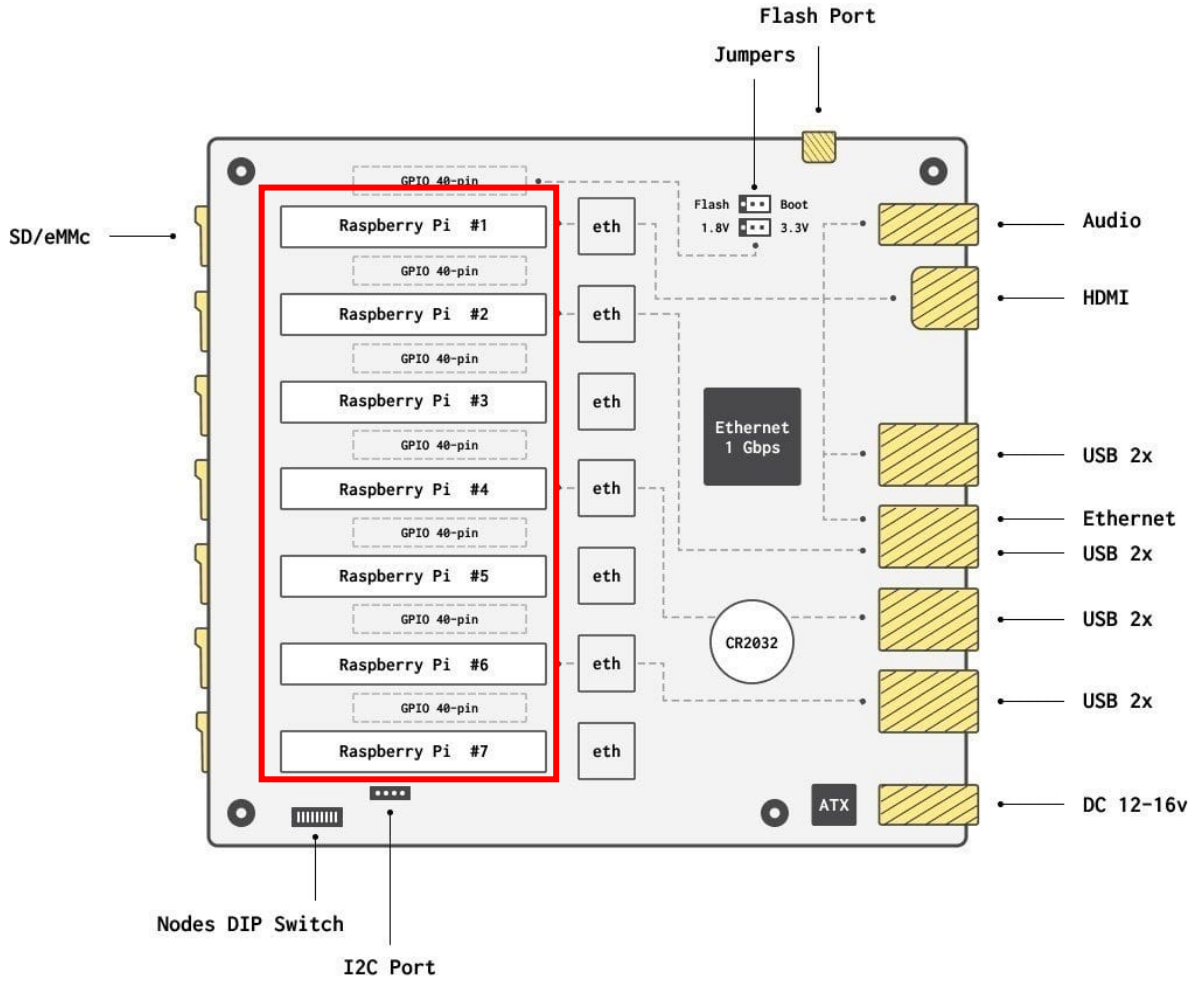
- Do you have any 3’s

- Go fish

Demo Infrastructure - turingpi v1

Kubernetes EDGE

- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device

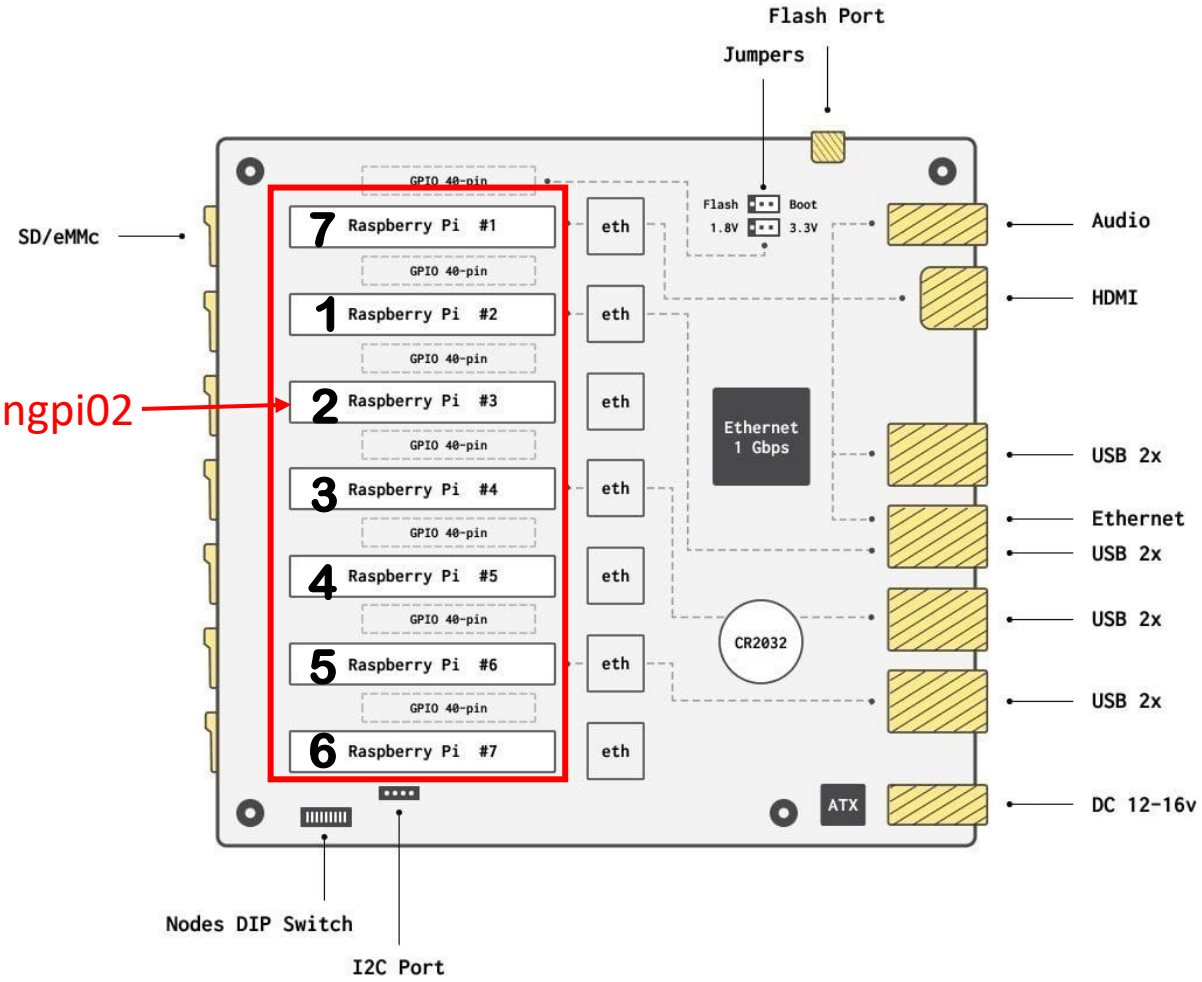


Demo Infrastructure - turingpi v1

Kubernetes EDGE

- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device

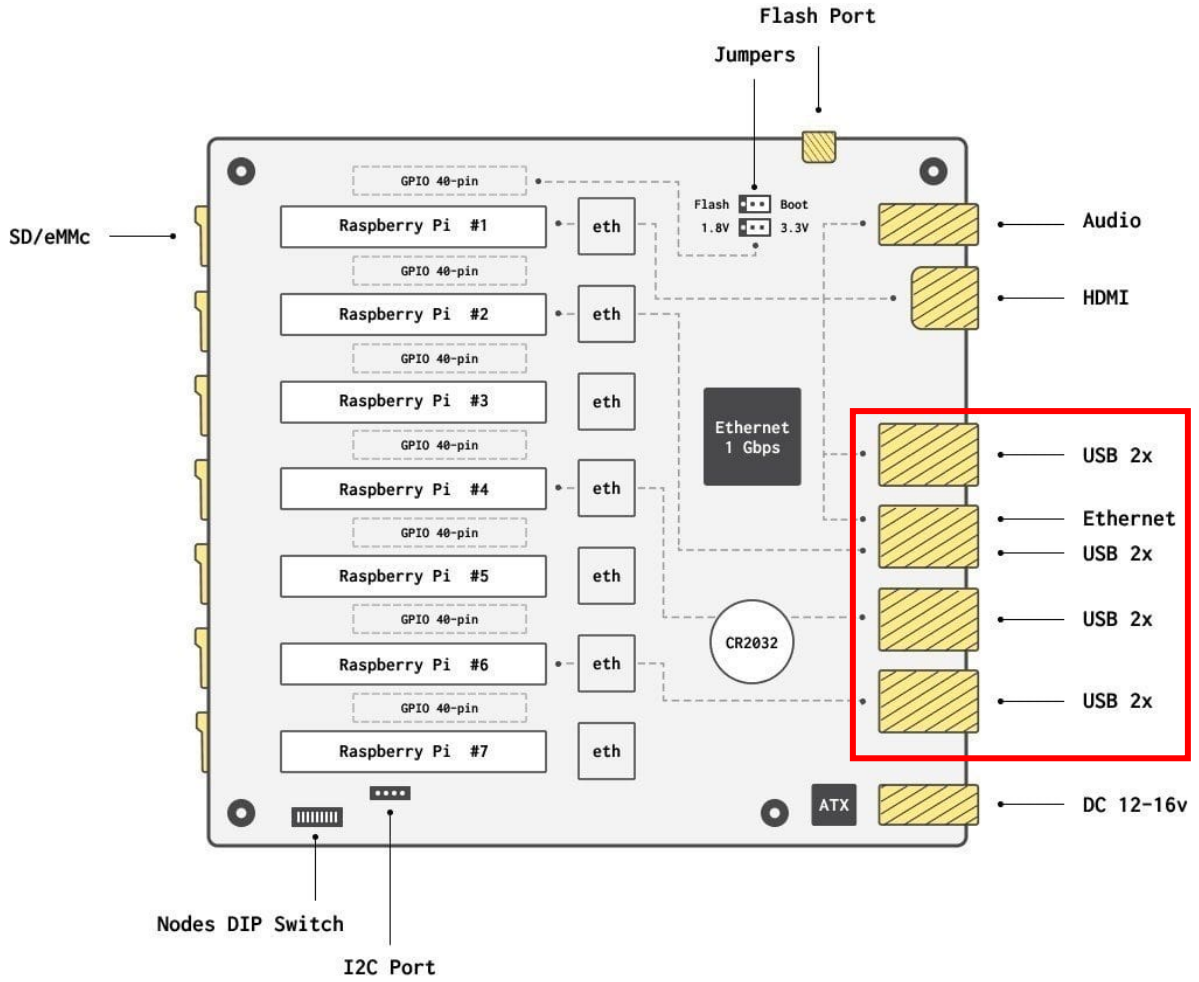
Hostname: turingpi02



Demo Infrastructure - turingpi v1

Kubernetes EDGE

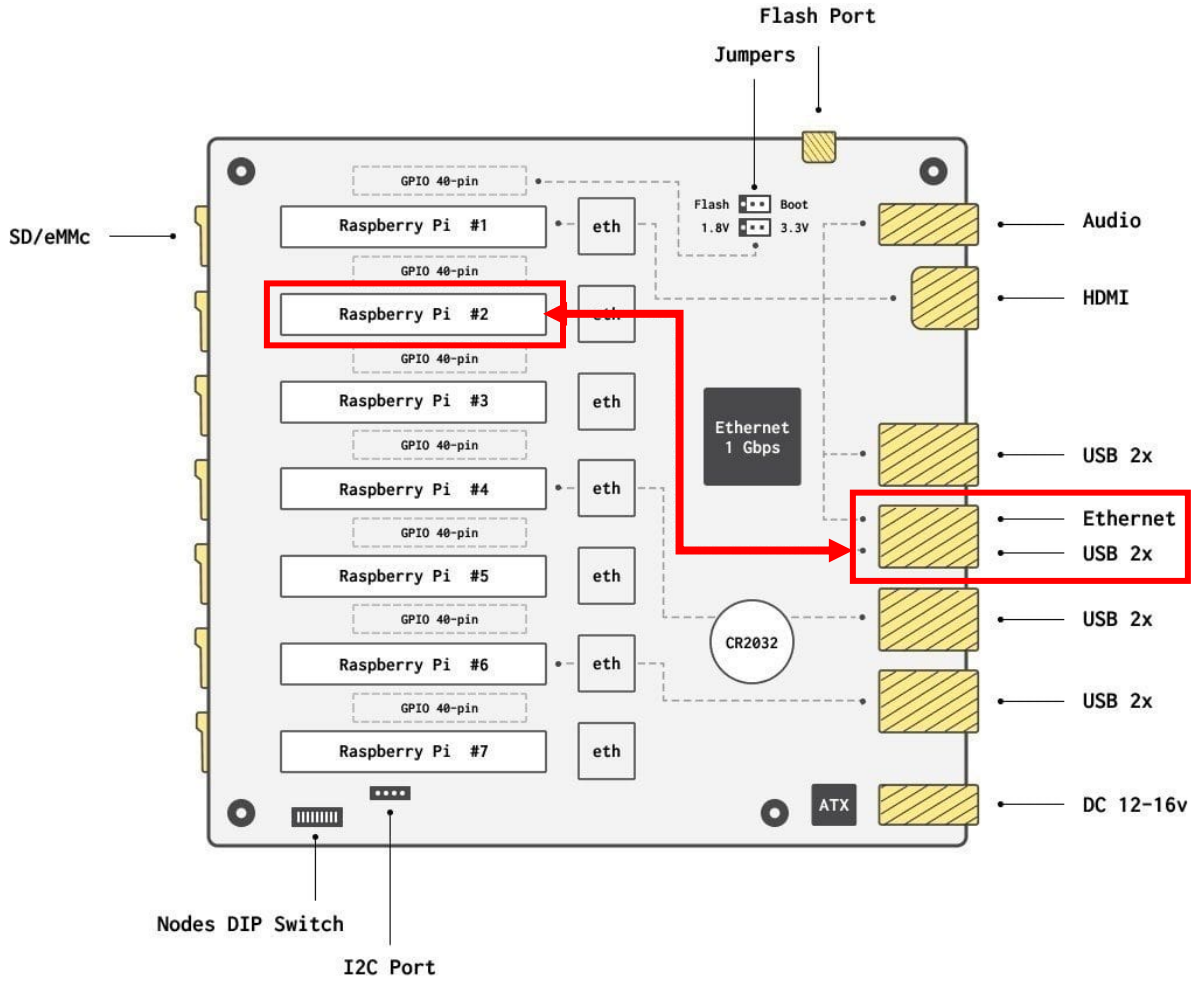
- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device



Demo Infrastructure - turingpi v1

Kubernetes EDGE

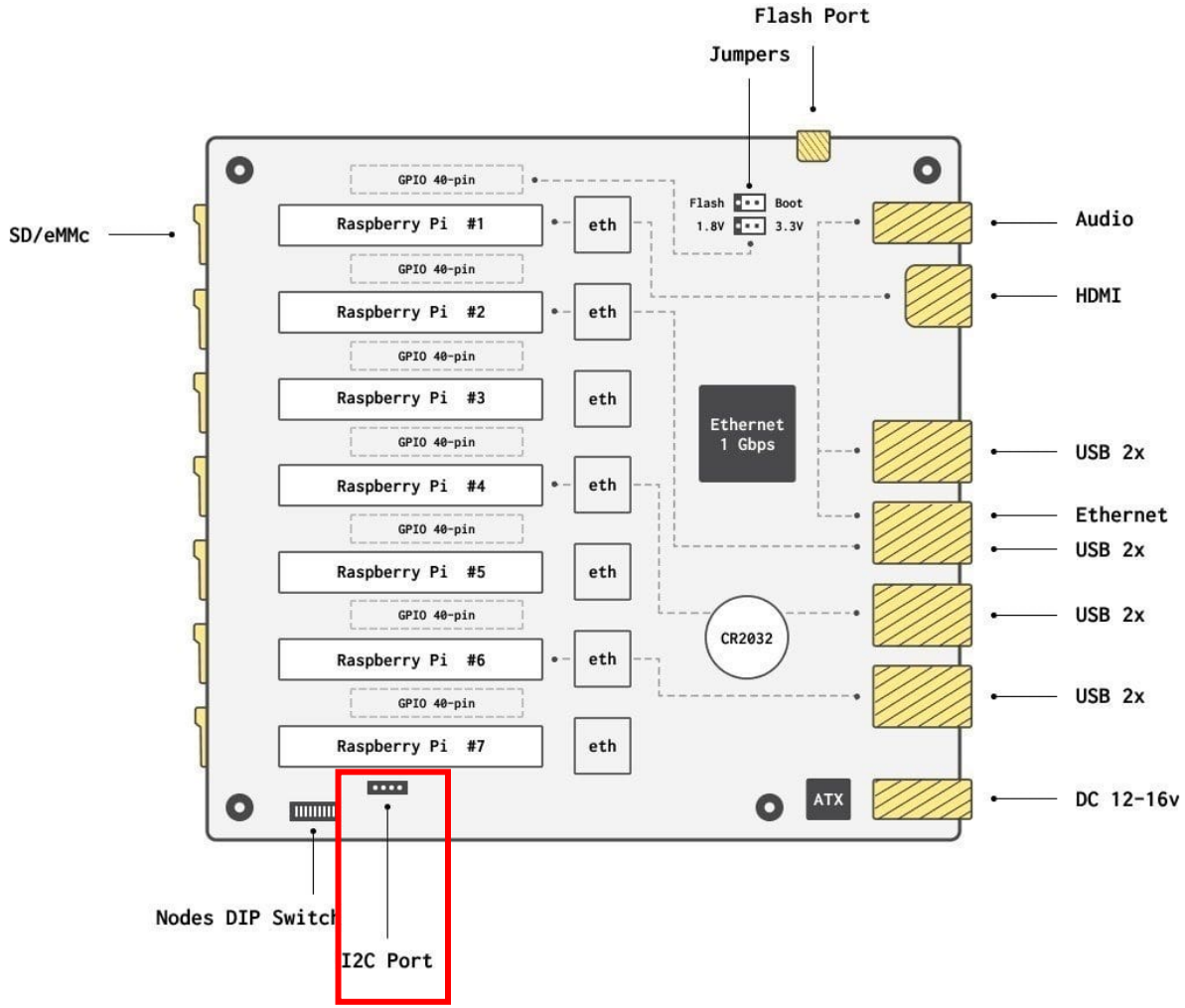
- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device



Demo Infrastructure - turingpi v1

Kubernetes EDGE

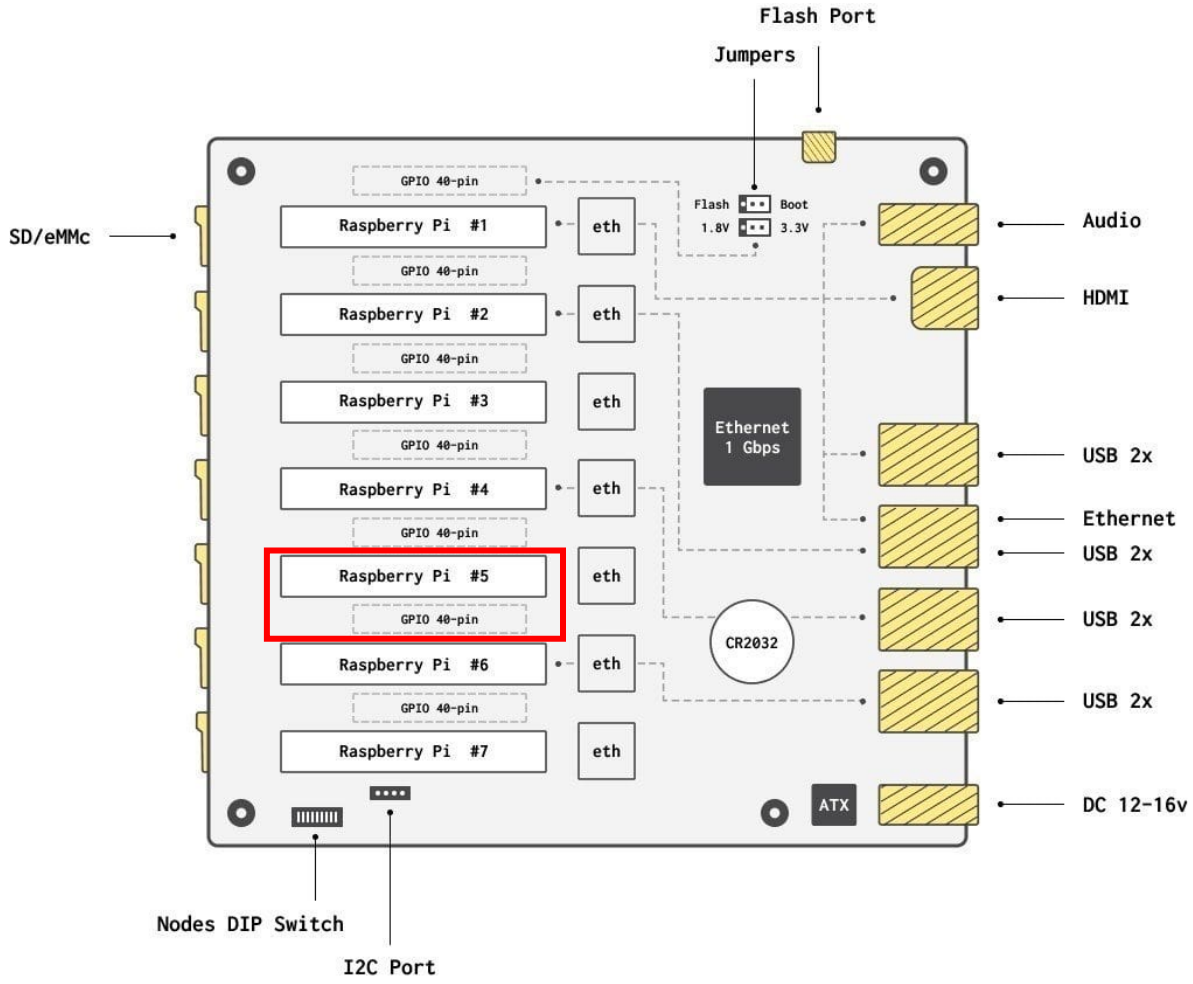
- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device



Demo Infrastructure - turingpi v1

Kubernetes EDGE

- 7 RaspberryPi CM3+ modules
  - 1GB RAM
  - 4 CPU
- 8 USB ports
  - Map to only 4 devices
- I2C interface
  - Maps to all devices
  - Communication between devices
- GPIO
  - 40 pins for each device



Kubernetes
E D G E

Demo

How does it work

Kubernetes
EDGE

- Two Kubernetes Resources
  - NFD Master (Deployment)
  - NFD Worker (DaemonSet)

How does it work

Kubernetes
EDGE

- Operations team responsibilities
  - Know your devices
  - Create configuration defining known resources
  - NFD does the rest

Install Node Feature Discovery

- CRDs
  - K8s templates (yaml config)
  - Helm Chart
  - Operator
- NFD does what it does based on a configuration file
- Configmap is the key to the labeling

Example ConfigMap


```
#core:
#  labelWhiteList:
#  noPublish: false
#  sleepInterval: 60s
#  sources: [all]
#  klog:
#    addDirHeader: false
#    alsologtostderr: false
#    logBacktraceAt:
#    logtostderr: true
#    skipHeaders: false
#    stderrthreshold: 2
#    v: 0
#    vmodule:
#    logDir:
#    logFile:
#    logFileMaxSize: 1800
#    skipLogHeaders: false
```

```
sources:
#  cpu:
#    cpuid:
##     NOTE: whitelist has priority over
blacklist
#      attributeBlacklist:
#        - "BMI1"
#      attributeWhitelist:
#  kernel:
#    kconfigFile: "/path/to/kconfig"
#    configOpts:
#      - "NO_HZ"
#      - "X86"
#      - "DMI"
  usb:
    deviceClassWhitelist:
      - "03"
    deviceLabelFields:
      - "class"
      - "vendor"
      - "device"
```

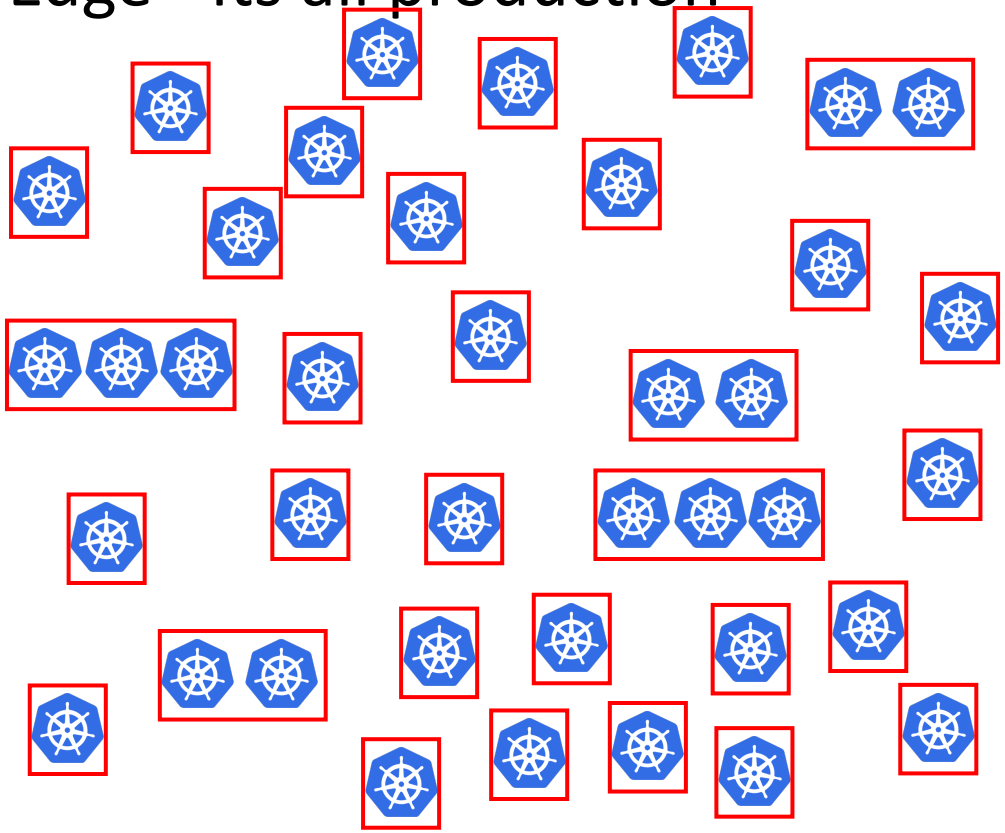
Practical NFD - Architecture Review

- Data Center / Cloud
  - dev
  - test
  - staging
  - prod

Practicatl NFD - Architecture Review

- Data Center / Cloud
  - dev
  - test
  - staging
  - prod

- Edge - Its all production



GitOps for Apps and NFD at the Edge

Kubernetes
EDGE

- Rancher Fleet
- Argo CD
- Flux

Example gitops workflow

1. Install NFD via the gitops pipeline into every cluster
2. Setup NFD configmap and commit to pipeline
3. The system managing the clusters should automatically
    1. recognize changes in downstream cluster node labels and
    2. update the cluster labels accordingly
4. Notify developers
    - of the node labels available
    - and what resource it maps to

- Repeat steps 2 - 4 on changes, hardware updates, etc.

Security

1. Mutual TLS capable between nfd-master and nfd-worker
2. Risk: Allowing Local - user specific features
    1. will execute arbitrary files located in
    2. /etc/kubernetes/node-feature-discovery/source.d on the host
3. Limit features discovered by specifying exact sources
    1. Whilelist/Blacklist features
    2. nfd-worker.conf:
        1. set only specific sources
        2. e.g. sources: [usb,custom,local]

NFD Summary

Kubernetes
EDGE

1. Label nodes in a cluster
2. NFD configuration is cluster scoped
    1. The edge will be thousands of single node or small clusters
    2. External tooling is needed to find groups of the same feature (gitops)
3. Feature discovery can be specified via configuration file
4. Limited features types
    1. Non kernel module features can be added using the *local* option
    2. Compare with Kubernetes device plugins

NFD Summary

Kubernetes
EDGE

1. NFD allows drop in configmaps to another location
    1. Vendors can/should provide configs for their devices
    2. The NVIDIA GPU operator currently takes advantage of NFD
2. NFD is not “just” a labeler.
    1. It is a dynamic labeler
    2. It is a configurable labeler

NFD Summary - Pitfalls

Kubernetes
EDGE

1. Kernel module existence is not a guarantee of functionality
    1. Kernel models can be loaded that have no physical counterpart
    2. K8s health checks may add value here
2. Hot plugging devices works but it is not immediate
    1. Kubernetes will need some cycles to label
    2. Re-labeling may also require redeployment of workloads

Links

- https://kubernetes-sigs.github.io/node-feature-discovery/v0.8/get-started/
- https://kubernetes-sigs.github.io/node-feature-discovery/v0.8/get-started/features.html
- https://www.usb.org/defined-class-codes
- https://k3s-io/k3s
- https://github.com/mak3r/nfd-demo

Kubernetes
EDGE

Thank you

Mark Abrams - Field Engineer and Edge Specialist
SUSE