



Secure Firmware Update for IoT Devices

Mohit Kedia, Christos Vasilakis

14th Sept 2021

Requirements for getting secure data



Need to ensure that the data sources are securely deployed, connected, & maintained



Need to ensure that the data is accurate, complete and consistent



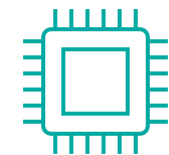
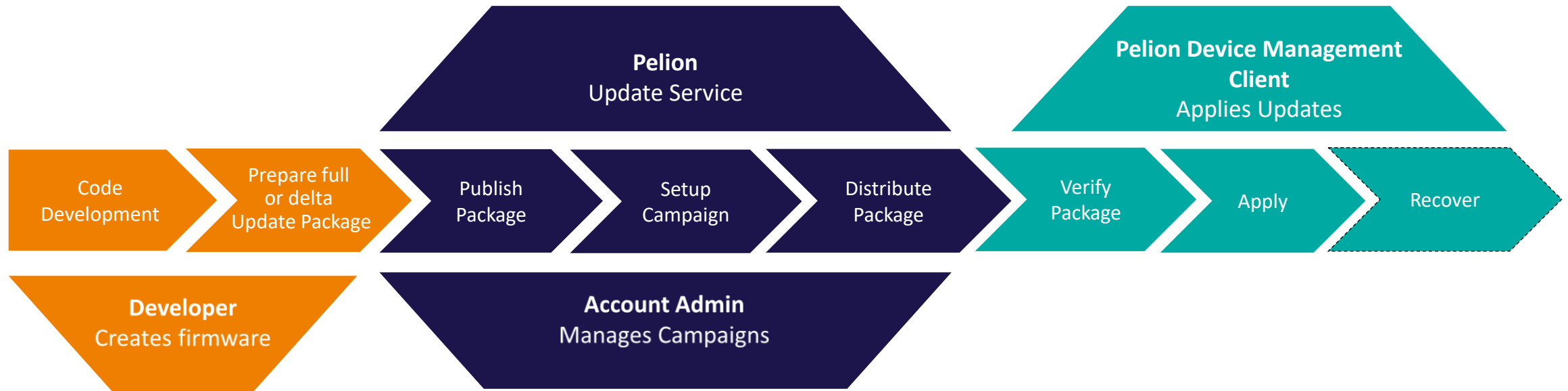
Need to ensure that the transport of the data is done securely to cloud

Remote software update is critical

- Software update is used to:
 - Fix bugs
 - Patch vulnerabilities
- Lack of security in update flow allows to:
 - Replace genuine software version with compromised software
 - Roll back software version to older version that has vulnerabilities or bugs
- As a security measure, it needs to work with secure boot & support millions of devices



Pelion Update Service is Designed for IoT Scale



Pelion Update Protections

ANTI ROLL-BACK



- Prevents installation of insecure/incompatible images
- Updates will only proceed if the version in the firmware manifest is newer than the currently installed firmware

POWER SAFE APPLICATION



- During application, the main image is checked and if corrupt the update will be re-applied
- Power failure at any time during the update will not “brick” the device.
- The device will re-apply half-completed updates on the next boot.

Pelion Update Protections

ACCIDENT PROTECTION



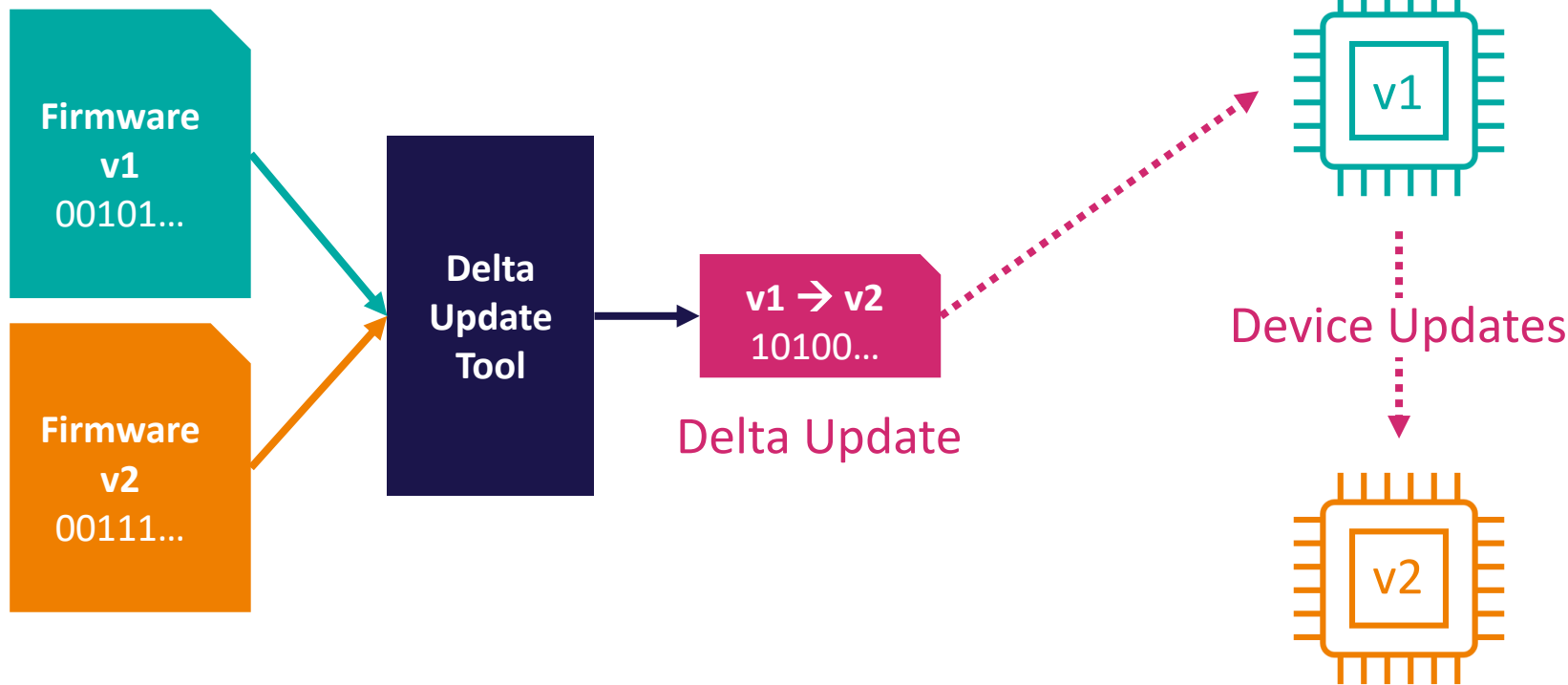
- The firmware manifest describes the devices that the firmware image is built for.
- The device client inspects the manifest and rejects updates that don't match the device.

RESUMABLE DOWNLOADS



- Sometimes power is lost before firmware has finished downloading.
- The device client can resume incomplete downloads when the power is restored.

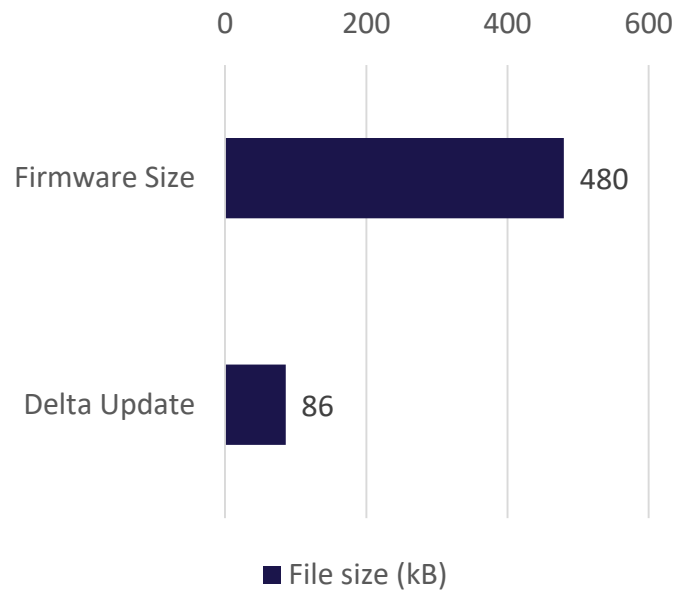
Delta Updates Reduce Bandwidth and Battery Use



- Two versions of firmware are compared for differences
- Only the “delta” is sent to the device for installation
- Large reduction in bandwidth and battery use

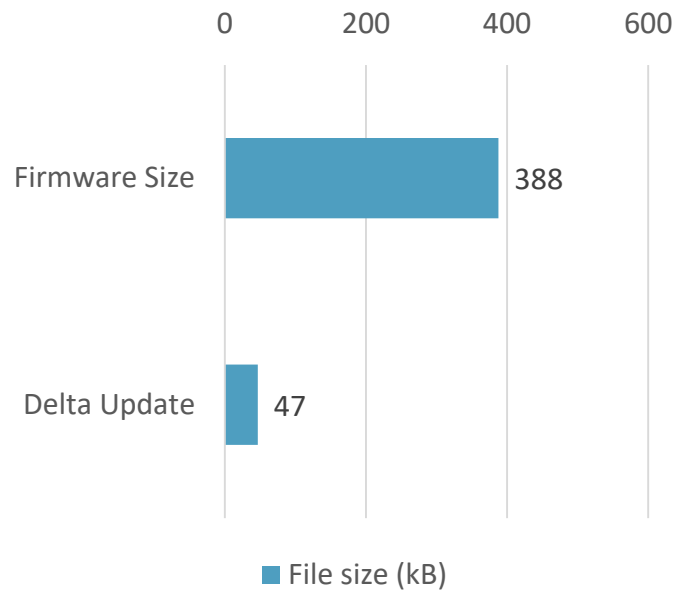
Example Delta Update Results

PDM client 2.0.0 to 2.1.0



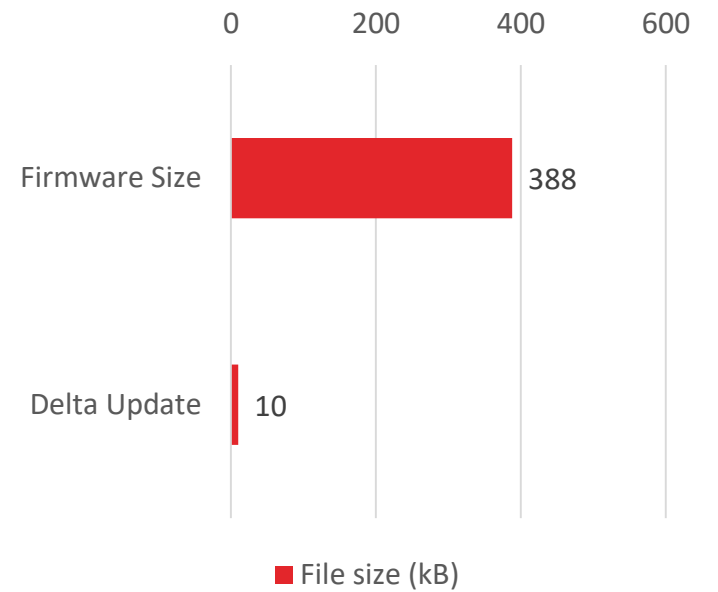
82% reduction

Adding a new driver



87% reduction

Simple string change



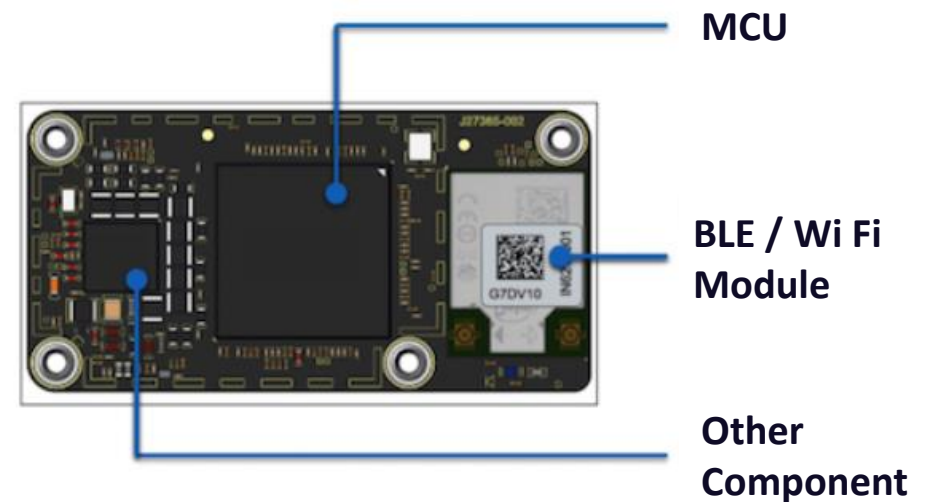
97% reduction

Updating Device Components

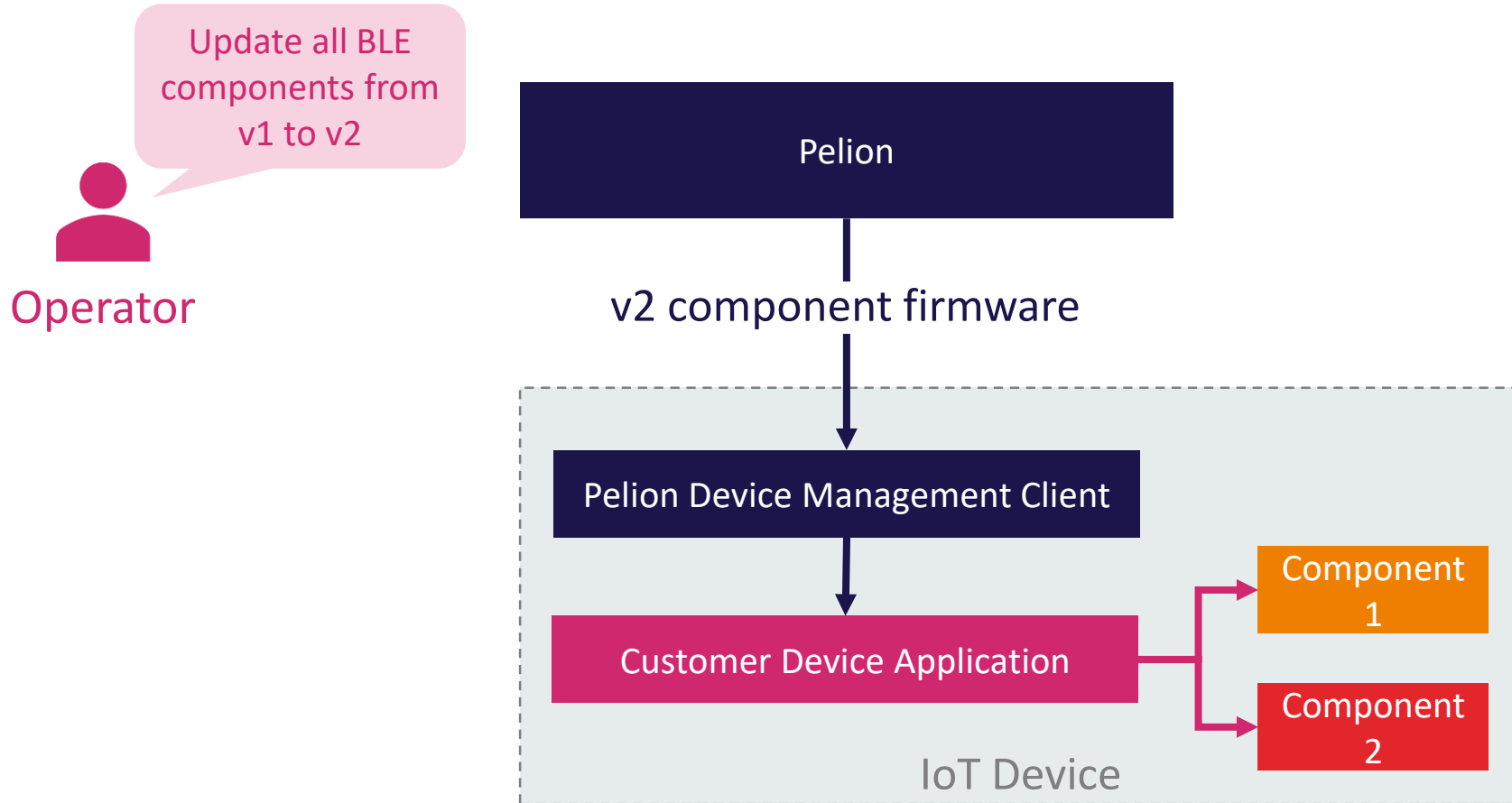
The challenge

- IoT devices may have multiple components
 - Multiple MCUs, comms modules, sensors, etc.
- These components may be provided by different manufacturers
- Each component needs continuous updates for product improvements and security fixes

A typical small IoT device



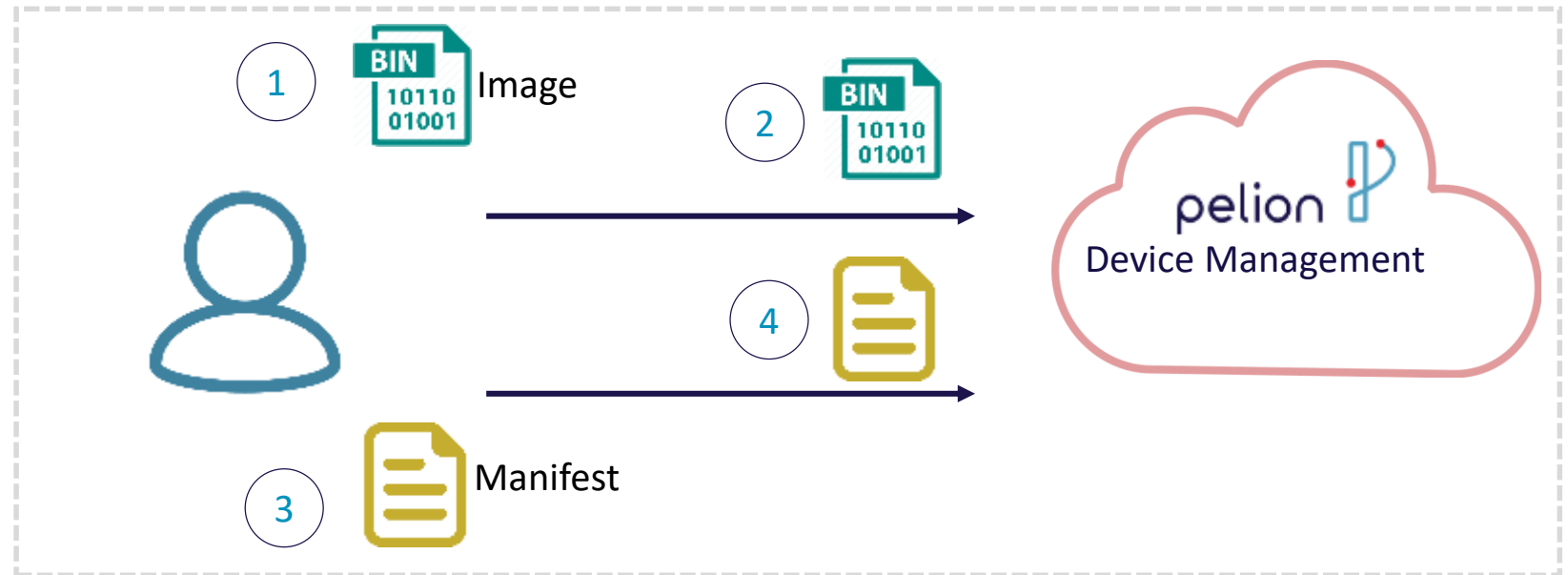
Pelion Delivers Component Firmware to Devices



- Pelion supports firmware campaigns for sub-components
- Pelion Client validates and verifies the data
- Customer app installs the component firmware

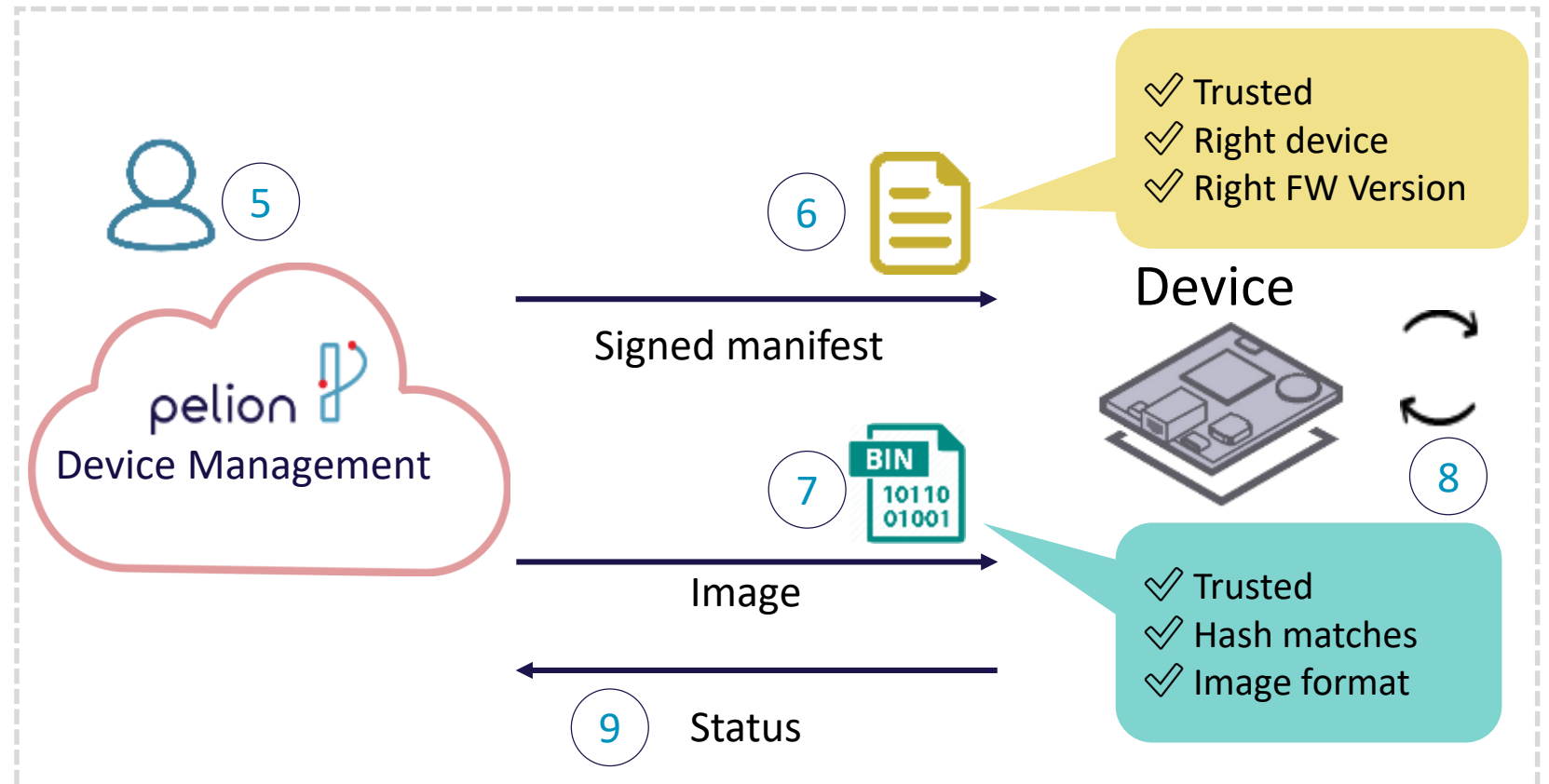
Pelion Update Flow – Creating Images

1. Build new firmware image
2. Upload image to Pelion
3. Create a manifest using the manifest tool
4. Upload the manifest to Pelion



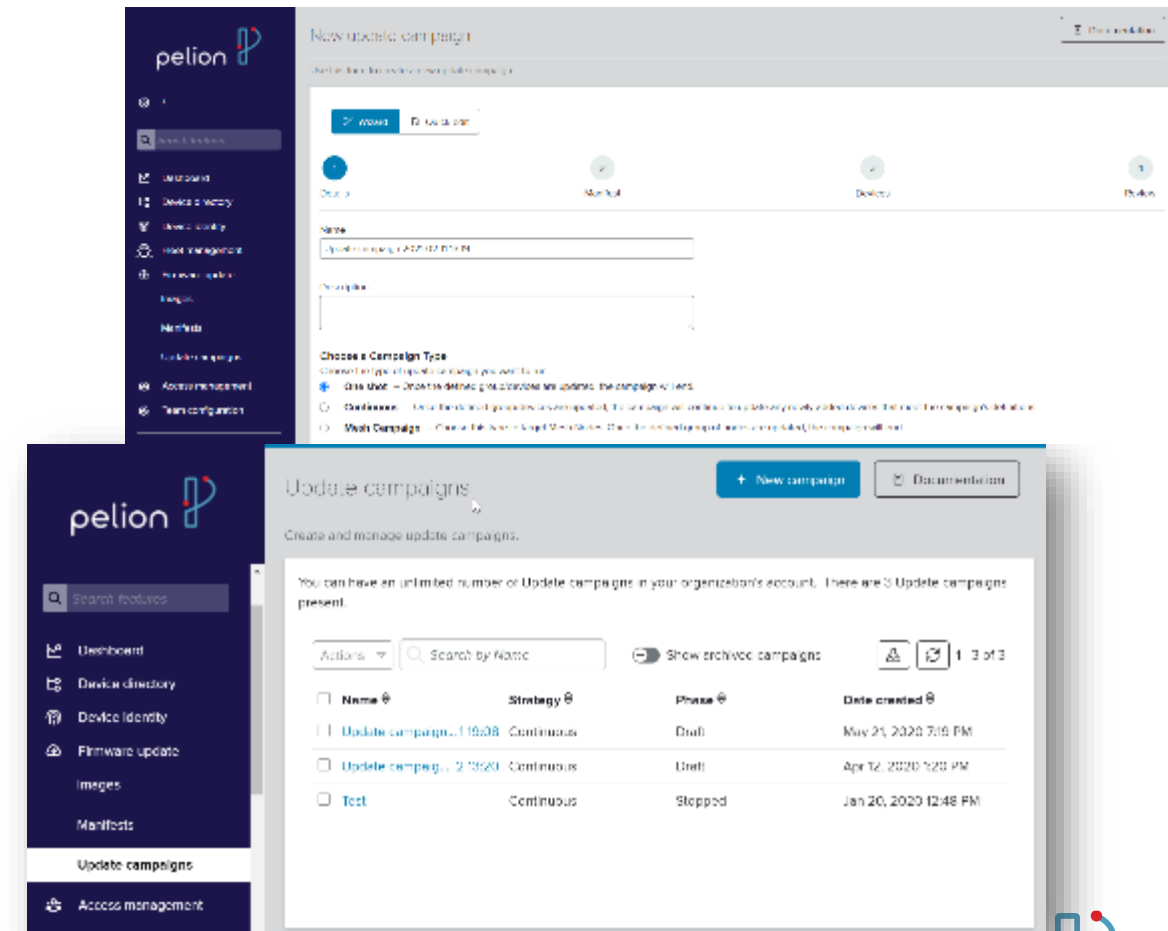
Pelion Update Flow – Deploying Updates

5. Update campaign created via Pelion Portal or API call
6. Device validates manifest
7. Device downloads image and validates
8. Device installs update
9. Device reports success



Update Campaigns

- Firmware update campaigns define which firmware to install onto a group of devices.
- Created through the web portal wizard or by using API calls.
- Powerful attribute-based filters choose which devices to target.
- Operators can see update campaign statistics in the portal.



Device attributes and filters

- Built-in attributes
 - Provided by Pelion.
 - Manufacturer, serial number, device type, etc.
- Custom attributes
 - Provided by you.
 - Arbitrary data that is appropriate to your domain.

The screenshot displays the Pelion Device Management interface. On the left is a navigation sidebar with options like 'Dashboard', 'Device directory', 'Devices', 'Insights', 'Device groups', 'Device events', 'Enrolling devices', 'Device identity', 'Firmware update', 'Access management', 'Team configuration', 'Billing reports', 'Job management', 'Profile', 'Help', 'Language', and 'Privacy'. The main area is titled 'Devices' and shows a list of devices with columns for 'Name', 'State', 'Type', 'Execution', 'Date creat', and 'Date boot'. A filter is applied: 'Lifecycle status is equal to Enabled'. A dropdown menu is open over the filter, listing various attributes such as 'Deployed state', 'Deployment', 'Description', 'Device ID', 'Device key', 'Device name', 'Etag', 'Endpoint name', 'Endpoint type', 'Enrollment timestamp', 'Enrollment mode', 'Firmware checksum', 'Gateway host name', 'Last operator suspension category', 'Last operator suspension date', 'Last operator suspension description', 'Last system suspension category', 'Last system suspension date', 'Last system suspension description', and 'Lifecycle status'. The 'Device ID' attribute is highlighted in the dropdown. To the right, a 'Device details' panel is open for a specific device, showing its ID and a tabbed interface with 'ATTRIBUTES' selected. This panel lists both built-in attributes (like 'pycstat_update_fiber') and custom attributes (like 'account_id', 'auto_update', 'bechling_suspension_date', 'bechlingged_timestamp', 'cc_id', 'component_attributes', 'connector_expiration_date', 'created_at').

Continuous vs One-Shot Campaigns

One-Shot



- A one-shot campaign performs one search to find devices that match the filter.
- The matching devices are updated.
- New devices are ignored.

Continuous



- Continuous campaigns keep searching for devices that match the filter.
- Whenever a match is found, an update is triggered.
- Useful for updating devices when they first connect.

E.g. “*update any device where version < 3.7*”.



Thank You
Obrigado

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה