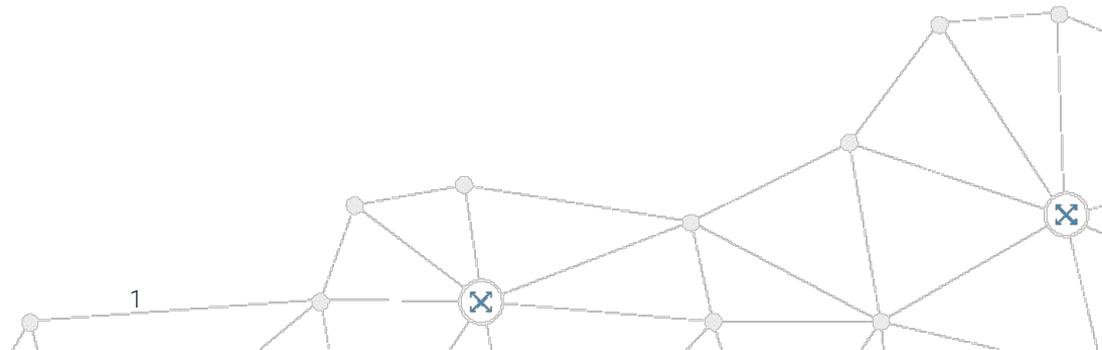Edge Analytics Online Training
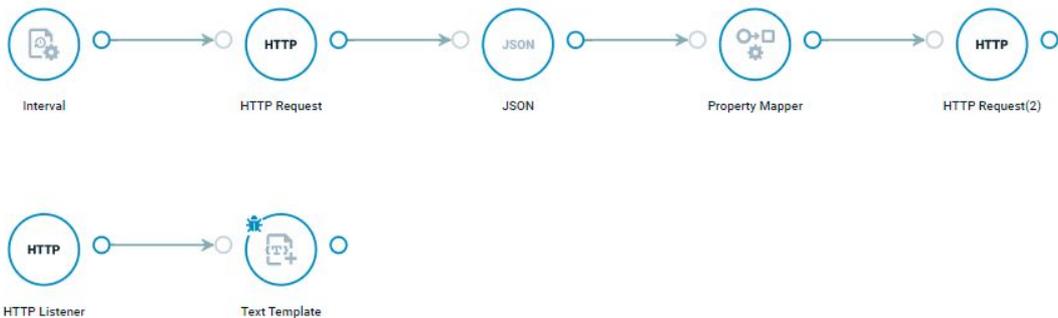
NON STREAMING DATA

**Work with non streaming data.
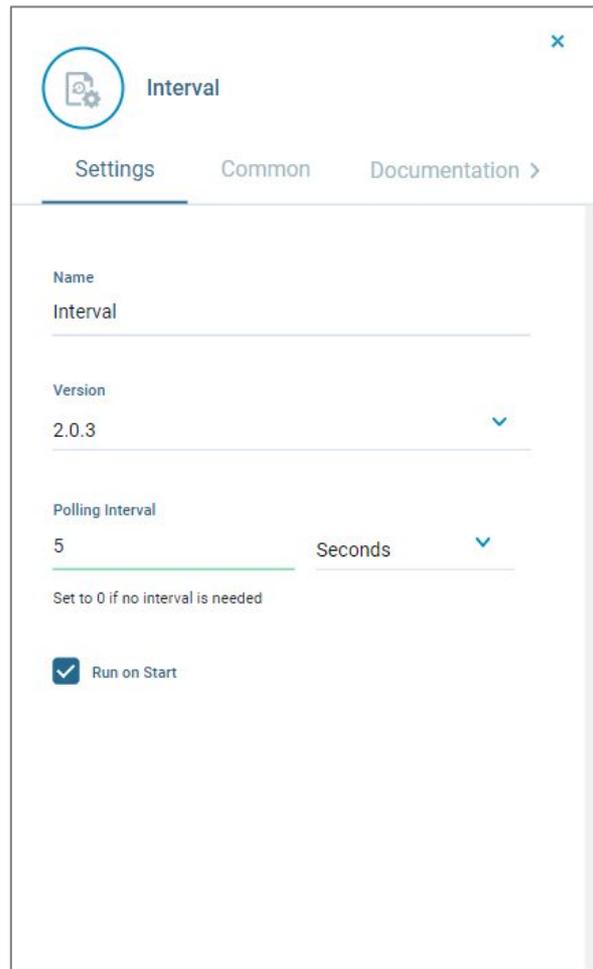Install a local node.**

Session 5
Agenda

- Exercise 4: Non-streaming data
  - Interval module
  - HTTP Request module
  - HTTP Listener module

- Local Nodes
  - Data directory
  - Dashboard
  - Local Node UI

- Exercise 5: Local Crosser Node
  - Installation
  - Deploy a flow
  - Flow versions

Module
Interval

- Calls the next module on a specified interval
- If you just want a single trigger when the flow starts, set "Run on start" to true and the "Interval" to 0

- For more complex trigger patterns, which can be aligned with wall-clock time, use the *Scheduler* module
- If you want to generate non-empty messages, use the *Data Generator* module

Module
HTTP Request

- HTTP client that can generate external HTTP requests with fixed or dynamic data (from flow messages).

- Supports GET, POST, PUT, PATCH, DELETE.

- The URL and body can be static (settings) or dynamic (message).

- Custom headers can be added.

- Can be an input (GET) or an output (POST, PUT, PATCH, DELETE) module.

- Can use Basic or Bearer authentication.
  - Oauth authentication can be used with the Universal Connector module (not covered in this course)

4

Module
HTTP Listener

- Get data through the internal HTTP server

- Listens on port 9090 by default
    - Can be changed in docker-compose.yml with Docker
    - Can be changed in data/httpconfiguration.json when running as a Windows service

- Incoming data can be filtered on the URL path (routing) and the verbs to accept. Wildcards can be used to match URLs.

- Data can be converted from JSON if *Content-Type* headers are present

5

EXERCISE 4

Use the HTTP Request module as an Input

Use the HTTP Request module as an Output

Receive data through internal HTTP server

Exercise 4
Overview

- Non-streaming data by using APIs
- Make basic requests to get data
- Select the relevant parts
- Send data to HTTP end-points
- Receive data through internal HTTP server

1.	Get Chuck Norris quotes
2.	Get weather data
3.	Select today's forecast and create 'friendly' forecast message
4.	Send forecast data to external end-point
5.	Receive forecast data through internal HTTP server

crosser
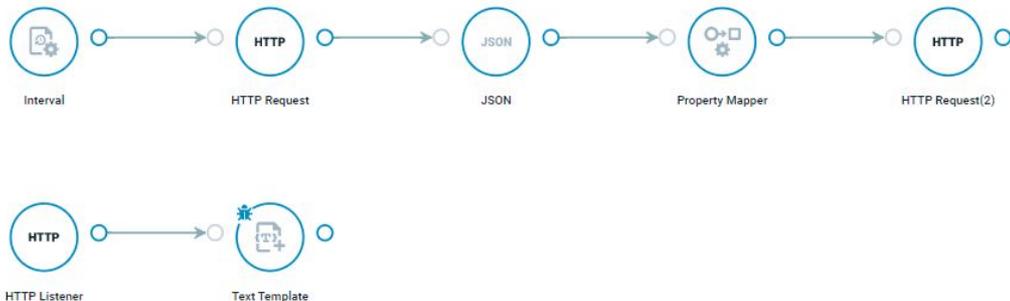
Exercise 4.1
The HTTP Request module as an Input

Get Chuck Norris quotes from external web service

1. Create a new flow called Exercise 4.1
2. Add an Interval module
    - Polling Interval: 0
    - Run on start: Enabled
3. Add a HTTP Request module and check the output:
    - URL: **https://httpbin.org/json**
4. Add a JSON module and check the output:
    - Source Property: data.body
    - Target Property: data

crosser

Exercise 4.2
HTTP Request as output
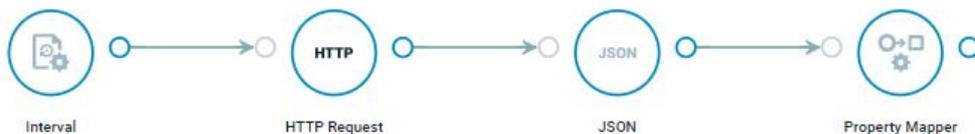
Get a weather forecast from an external web service

1. Make a copy of the previous Flow and call it Exercise 4.2 ('New Flow from draft' in the tab menu)
2. Update the HTTP Request module:
   - URL: http://www.7timer.info/bin/api.pl?lon=113.17&lat=23.09&product=civillight&output=json
3. Add a Property Mapper module to select today's forecast (the first element in the array):
   - Add a *Move* rule: from data.dataseries[0] to body
4. Check the message coming out from the Property Mapper module

Exercise 4.2
HTTP Request as output

Make an HTTP request to the internal HTTP server

5. Add another HTTP Request module after the Property Mapper module:

    - URL: http://localhost:9090/weatherData
    - Verb: POST

Exercise 4.2
Receive external HTTP requests
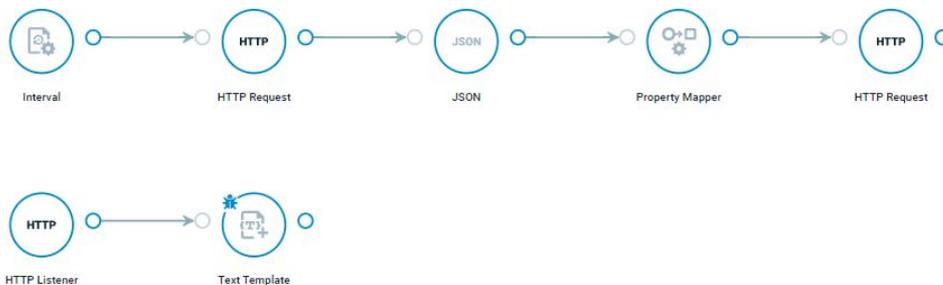
6.　Add a HTTP Listener module:

- Path: weatherData

7.　Add a Text Template module:

1.　Target Property: message
2.　Template: Today it's going to be {data.body.weather} weather with a temperature between {data.body.temp2m.min} and {data.body.temp2m.max} degrees. The wind is expected to be {data.body.wind10m_max} m/s

9.　Check the output of the Text Template module

Exercise 4.2
Validate external requests
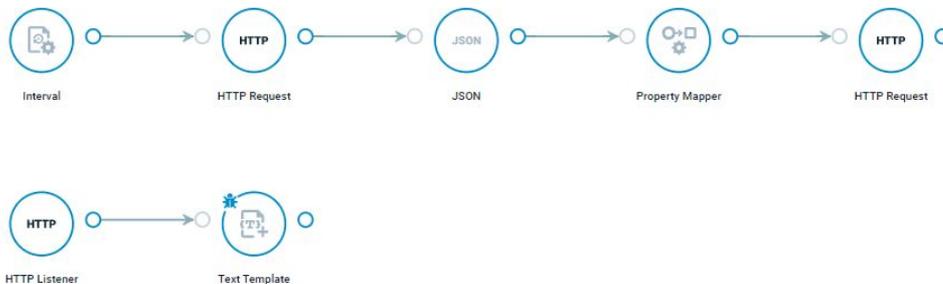
9. In the HTTP Request module, change the path of the URL to something invalid, e.g. 'bin' □ 'bi'

10. Run the Flow and check the output of the HTTP Request module, especially the *crosser.success* property

11. Add a message filter on the JSON module to discard invalid message
    - Source Property: crosser.success
    - Operator: Is True

Note

It is good practice to always check the crosser.success property on the first module after a module that makes external requests, to avoid trying to process messages with invalid data

Exercise 4
Wrap-up

Things to test/consider:

- Why do we need the Property Mapper in front of the HTTP Request module?
- Why did we set the *From* value to 'data.dataseries[0]' in the Property Mapper? What will you get if you change the index to 1?
- Why did we set the *Path* in the HTTP Listener module to 'weatherData' and what happens if you change it?
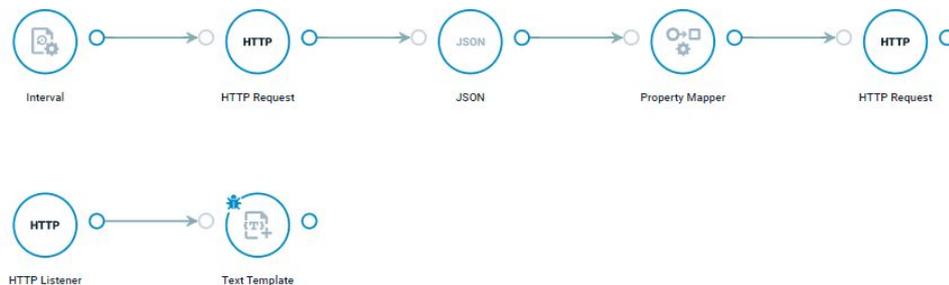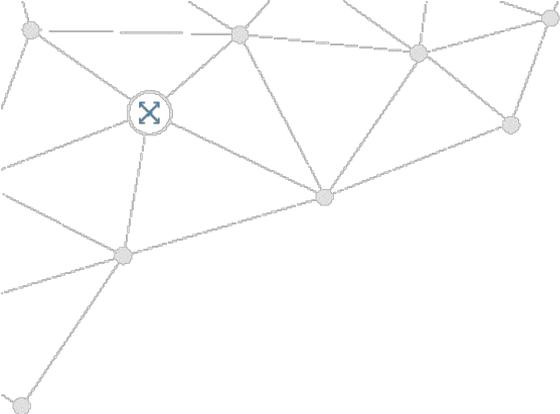
INSTALLING A LOCAL NODE
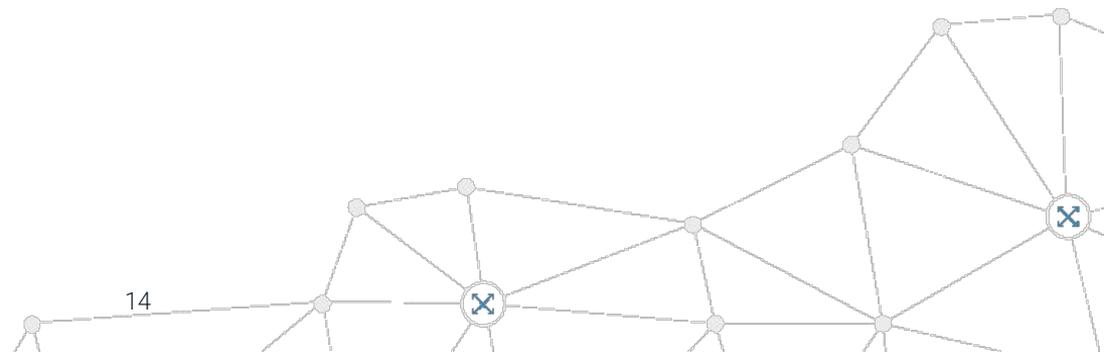
Local Crosser Nodes

- Two options for installation of a Crosser Node
  - As a Docker container (Linux and Windows 10/11)
  - As a Windows service (Windows 10 and Windows Server 2016+)
- You will find how to install a Crosser Node in the documentation [here](). Also covered in the next exercise

crosser

The 'data' Directory

- Docker: `<docker start dir>/data`

- Windows: `<install dir>\Host\data`

- Used for:
  - Node configuration files - *(data/*.json)*
  - Log files - *(data/logs)*
  - Flows - *(data/flows)*
  - Resources - *(data/flowresources)*

- Can be accessed from flow modules as "data/…", eg in file readers and code modules

These files are critical for the operation of the node – Be careful!

crosser

Dashboard
In Control Center

- The Dashboard is based on status information from the Nodes delivered at regular intervals (default every 10 seconds)

- You can see:
  - Connectivity summary
  - Traffic summary over all nodes (current values and graphs)
  - Per node data:
    - Connectivity status
    - Software version
    - CPU load
    - Traffic in/out
    - Flows running/stopped

- Events (more details on *Events* page)
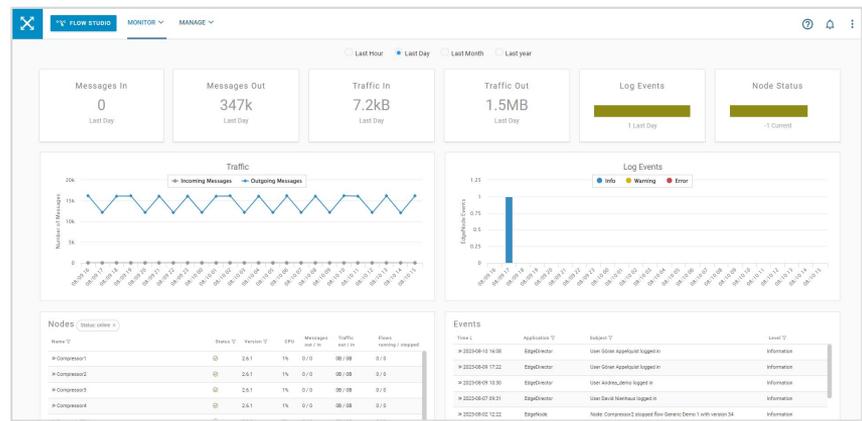
Local Node UI

- Each node has its own dashboard with detailed information
  - http://localhost:9191
- You can see:
  - Resource usage (CPU, memory, exceptions…) for the host and per flow
  - Terminate flows and the host (if enabled)
  - Read and download log files
  - Release notes
  - API documentation
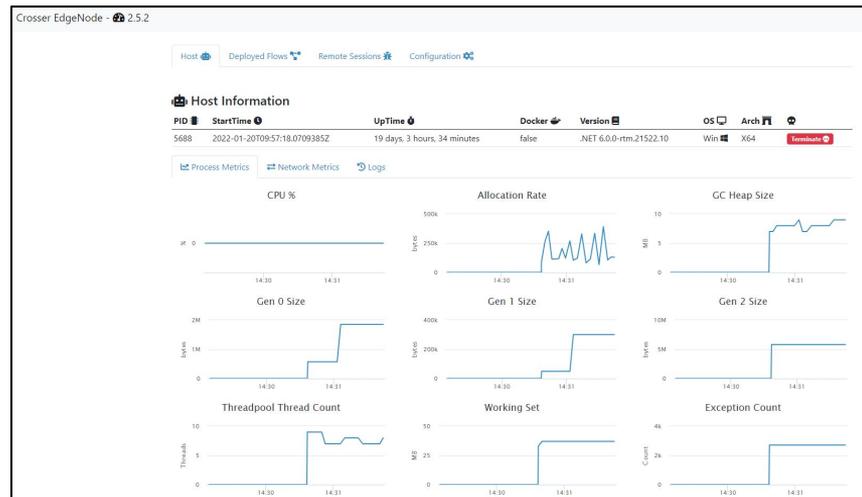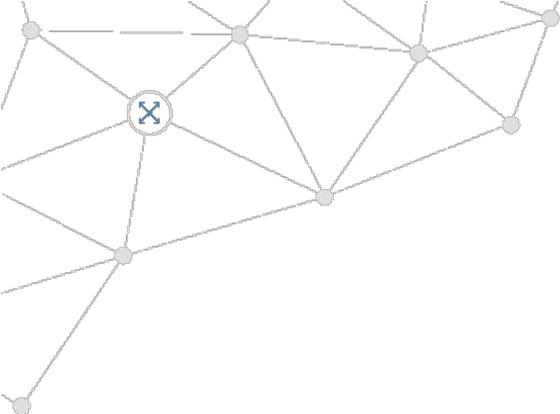
EXERCISE 5.1

Install a local Crosser node

Note: Installing a local Node is not required to complete the remaining exercises
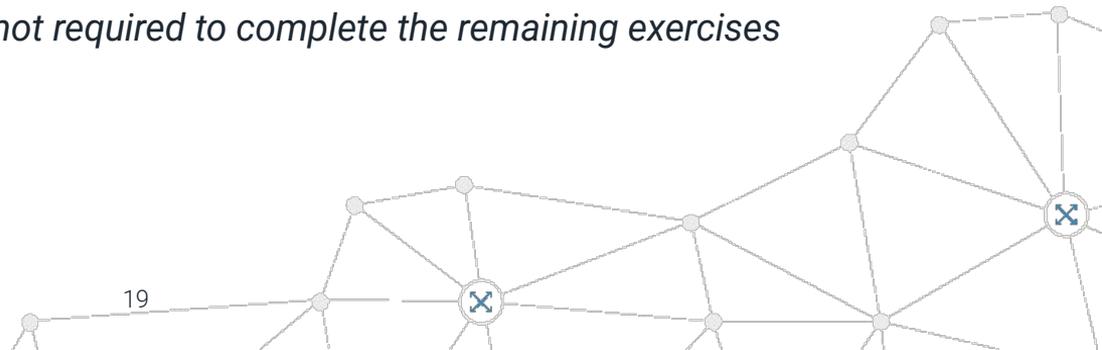
Exercise 5.1
Option 1: Install a Node using Docker

- Prerequisites
    - Docker is installed
    - Docker Compose is installed

- Installation steps (see docs)
    1. Register a node on the *Nodes->Register Nodes* page (copy the *NodeId* and *AccessKey*)
    2. Login to registry.crosser.cloud (credentials in Crosser Cloud) from your local machine
    3. Get the *docker-compose.yml* file (from documentation) and store it in an empty directory on your local computer
    4. Edit the docker-compose.yml file and insert the NodeId and AccessKey from Crosser Cloud (replace placeholders)
    5. Start the Node with 'docker-compose up –d'
    6. Check on the *Nodes* or *Dashboard* pages that the node is active
    7. Test one of your flows on this node
    8. Open the local Node UI at http://localhost:9191

crosser

Exercise 5.1
Option 2: Install a Node as a Windows Service

- Prerequisites
    - Windows 10/11 Pro/Enterprise or Windows Server 2016+
    - Note: On Windows 10/11 you can also use Docker, by installing *Docker Desktop*

- Installation steps (see docs)
    1. Register a node on the *Nodes->Register Nodes* page (copy the *NodeId* and *AccessKey*)
    2. Download the Windows installer from the *Nodes* page, on the *Register Nodes* tab
    3. Extract the file into an empty directory
    4. Open a PowerShell as Administrator and go to the directory where you extracted the installer
    5. Run .\InstallWindowsService.ps1 and enter the *NodeID* and *AccessKey* when asked
    6. Answer 'run' on the last question
    7. Check on the *Nodes* or *Dashboard* pages that the node is active
    8. Test one of your flows on this node
    9. Open the local Node UI on http://localhost:9191

crosser

Exercise 5.2
Introduction

- For this exercise it's good to have a local MQTT client, so that you can see the output of your deployed flow. An easy to use, still powerful client that works on all platforms can be found here: http://mqtt-explorer.com/

- Connect the MQTT client to localhost:1883 before starting the exercise

crosser

Exercise 5.2
## Deploy a flow

1. Create a new flow called Exercise 5

2. Add a Data Generator module and a MQTT Pub Broker module to publish the 'data' object on the internal MQTT broker:
   - Topic: test
   - Source Property: data

3. Run the flow and check that you get some data in the external MQTT client

4. Stop the Flow

5. In the tab menu of your Flow, use the Manage Deployments action. Select your local node and then click on 'Deploy'

6. Note the status of your deployment in the deployment tool and wait until it says 'Started'

7. Verify that you get data in the MQTT client

8. Leave the Flow Studio and open the *Flows page* (this flow will now run until you remove it from the node)

crosser

Exercise 5.3
Flow versions

1. On the *Flows* page, notice that your flow can no longer be modified

2. Add a new version by expanding the Flow row and in the menu to the right (three vertical dots) select 'New Draft from Version' (this can also be done using the tab menu in the Flow Studio)

3. Make a change to your message template in the 'Data Generator', e.g. add some more data or change some names

4. Open the deployment tool and switch to the 'Deployments' tab

5. Select your Node in the list

6. Click on 'Change to this version'

7. Notify the change in your MQTT client

8. Delete the flow from your node, if you don't want to keep it running? You can do this in the deployment tool ('Undeploy'), or by selecting your node on the Nodes page, then select the flow in the panel on the right and click on 'Delete'
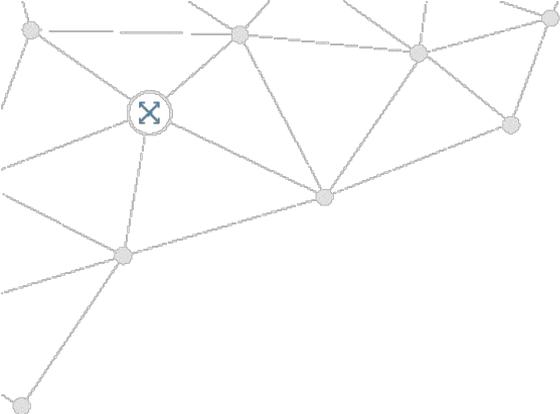
crosser

SESSION – 05 END

Working with non-streaming data

Install a local Crosser Node

Dashboards and the 'data' directory

Flow deployments and versions