Distributed Networked Radio Deployments Using The DELTA Framework

GNU Radio Conference 2024

2024.09.19

Jason Merlo and Jeffrey Nanzer

Michigan State University, East Lansing, MI, USA

What is the DELTA Framework?

“ …a collection of software tools and libraries which enable simple configuration and deployment to short-lived distributed networks of compute nodes, typically connected to radio peripherals running GNU Radio, designed to enable rapid iteration.

DELTA Framework Goals

1. **Scalable development environment capable of handling 10—100s of devices**
   - ✓ Easily deploy code from a central location (e.g., dev laptop/PC) to a network of computers
     - ✓ Handle synchronization in a bandwidth-efficient manner and use incremental builds (preserve build cache between deployments)
   - ✓ Easily monitor and control from a centralized interface (control node)
2. **Scalable software interface**
   - ✓ Maintain GNU Radio companion for simple visual data-flow-centric implementation while enabling scalability
   - ✓ Minimize copy/pasting blocks of code for parallel data channels

DELTA Framework Components

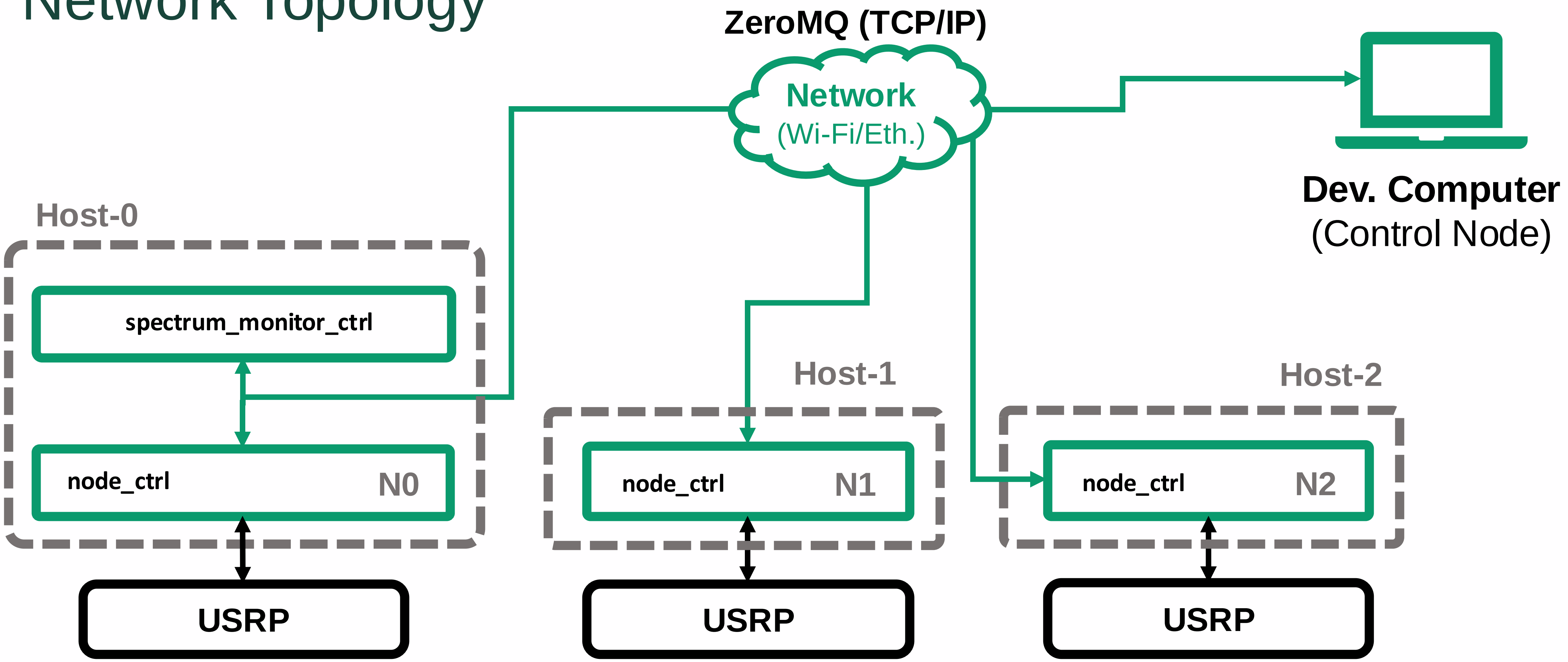
1. **DELTA Project Tool (da_protool)**
   - Lightweight tool (similar to gr_modtool) for creating DELTA Framework projects, and adding controllers
2. **GNU Radio Packages**
   - **gr-delta-utils**
     - Utilities for operating on lists of PDUs to enable scalability while maintaining support for GRC-based program design
   - **gr-delta-coordination**
     - Software tools for wirelessly synchronizing time and phase between radio nodes with high accuracy
3. **DELTA Python Package**
   - Python package for RF signal processing operations on bursty data; acts as the foundation for gr-delta-coordination

Packages in green → Available today

Packages in black → To be released

The DELTA Framework // Example

Network Topology



DELTA Framework // Example

Project Creation

```
$ da_protool newproj spectrum_monitor

Creating distributed array project in ./da-spectrum_monitor ...

spectrum_monitor
 ├── config
|    ├── hosts.yml
|    ├── libraries.yml
|   └── radios.yml
 ├── controllers
 ├── docs
└── libs
```

Host configuration information:

- IP address
- Scripts to launch / script args
- Radio resources connected

DELTA Framework // Example

Project Creation

```
$ da_protool newproj spectrum_monitor
Creating distributed array project in ./da-spectrum_monitor ...
spectrum_monitor
 ├── config
|    ├── hosts.yml
|    ├── libraries.yml
|    └── radios.yml
 ├── controllers
 ├── docs
└── libs
```

← Libraries to synchronize and install to all compute nodes:

- Path to library
- Folders to exclude
- Installation commands

DELTA Framework // Example

Project Creation

```
$ da_protool newproj spectrum_monitor

Creating distributed array project in ./da-spectrum_monitor ...

spectrum_monitor
 ├── config
|    ├── hosts.yml
|    ├── libraries.yml
|   └── radios.yml
 ├── controllers
 ├── docs
└── libs
```

Radio configuration info:

- Radio identifier (name, serial number, IP address, etc.)
- Sample rates
- TX/RX Channels to use
- Port assignments
- Initial Gains
- Initial Center Frequencies

DELTA Framework // Example

Controller Creation

```
$ da_protool add monitor_node

Adding controller files to "./controllers/monitor_node_ctrl"

controllers
└── monitor_node_ctrl
    ├── config
    |   └── monitor_node_ctrl.yml.mako
    └── monitor_node_ctrl.grc
```

DELTA Framework // Example

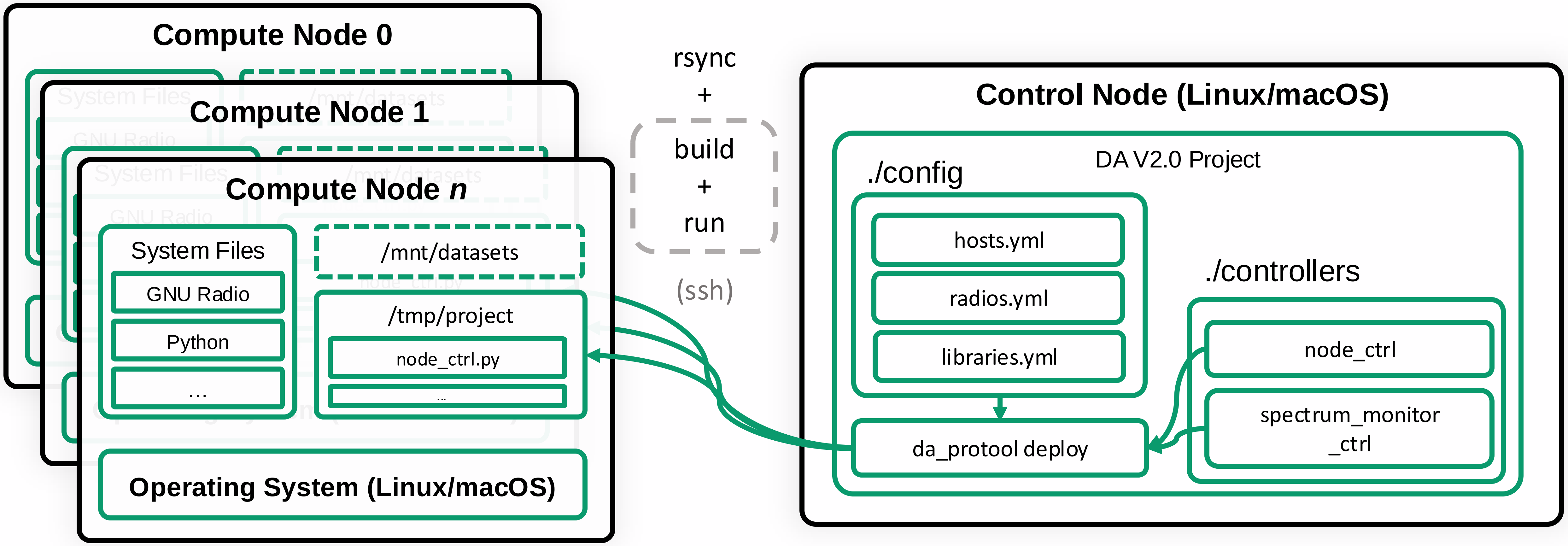
Controller Creation

```
$ da_protool add spectrum_monitor
Adding controller files to "./controllers/spectrum_monitor_ctrl"
controllers
├── monitor_node_ctrl
|   ├── config
|   |   └── monitor_node_ctrl.yml.mako
|   └── monitor_node_ctrl.grc
└── spectrum_monitor_ctrl
    ├── config
    |   └── spectrum_monitor_ctrl.yml.mako
    └── spectrum_monitor_ctrl.grc
```

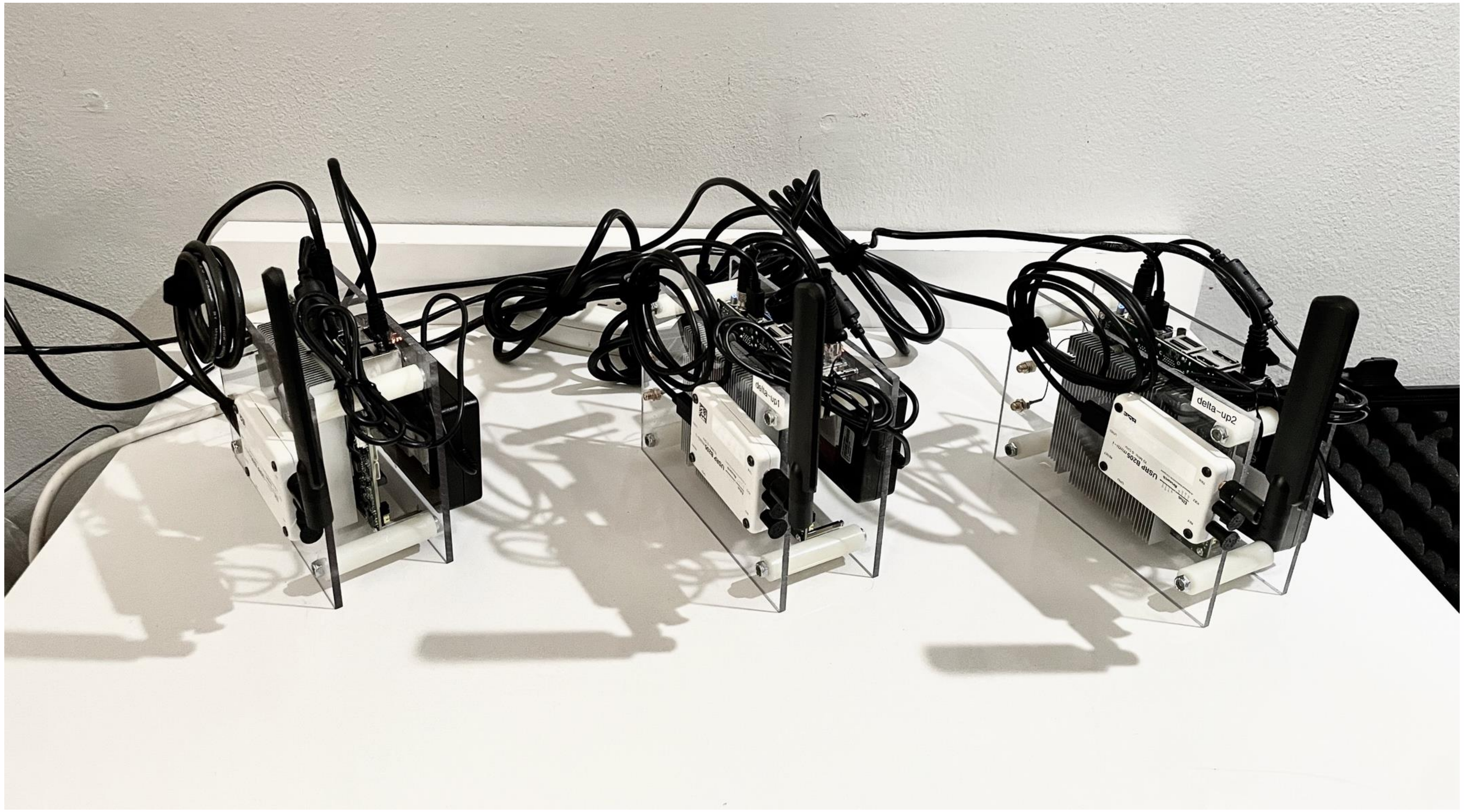
DELTA Framework // Example

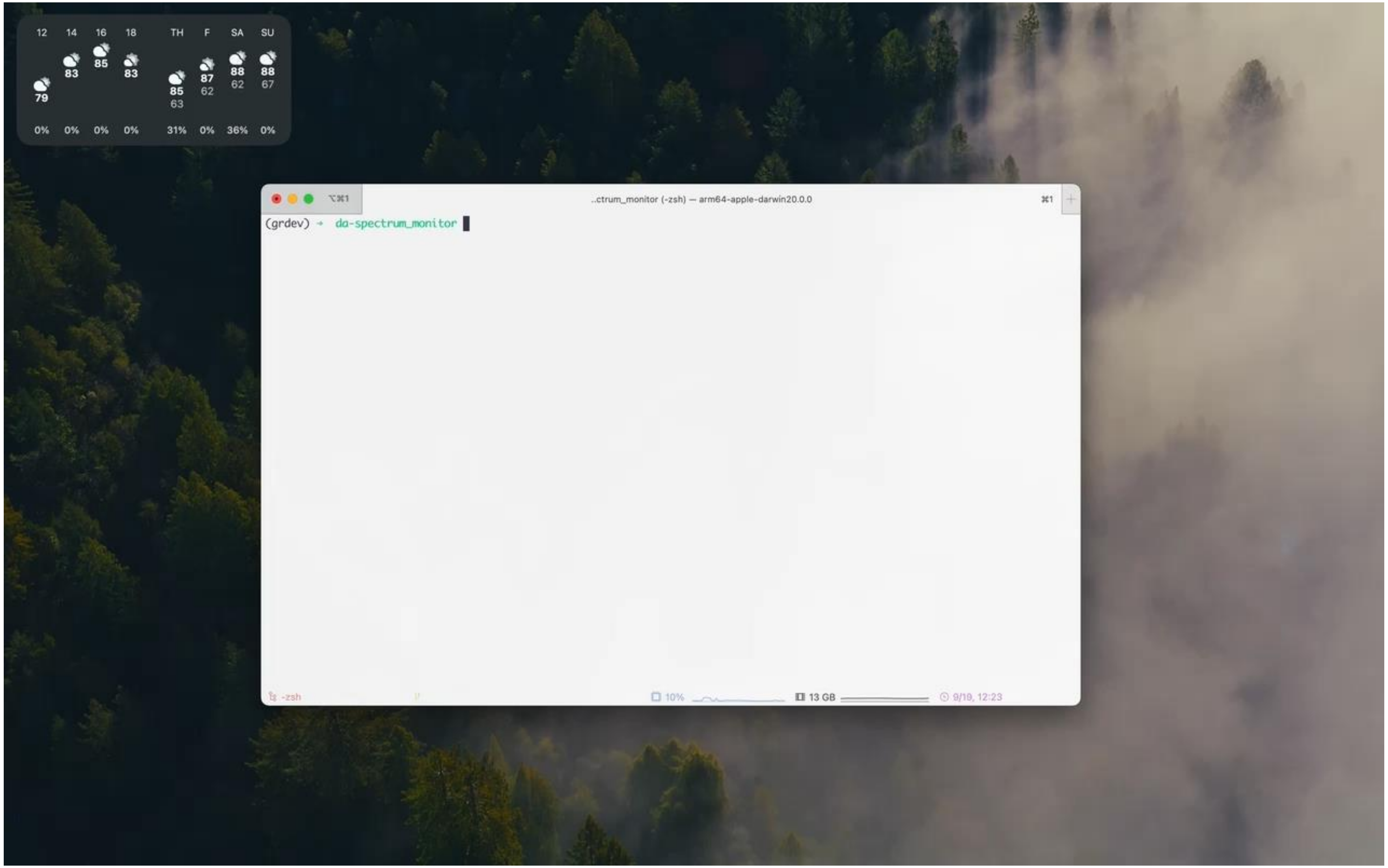
Deployment Process

```
$ da_protool deploy
```



Demo





```
(grdev) → da-spectrum_monitor
```

```
-zsh                    10%        13 GB        9/19, 12:23
```

Thanks!

gitlab.msu.edu/delta/da-framework