

Pikalert®: NSF -NCAR's Road Weather Hazard Prediction System

Tom Brummet
Brummet@ucar.edu
Software Engineer, Research Applications Laboratory
NSF NCAR



NSF - NCAR

- The NSF National Center for Atmospheric Research (NCAR) is a U.S. research center dedicated to understanding the atmosphere, Earth system and Climate.
 - Sponsored by the U.S. National Science Foundation (NSF) and Managed by the University Corporation for Atmospheric Research (UCAR)
 - <https://ncar.ucar.edu/>
- The Research Applications Laboratory (RAL) is one of 8 Labs at NCAR.
 - “RAL conducts directed research and engineering to discover solutions to problems relevant to society and facilitates the transfer of our information, expertise, and technology to the public and private sectors.”
 - <https://ral.ucar.edu/>

Pikalert[®] Introduction

- **What is Pikalert?**
 - Pikalert is an Enhanced Maintenance and Decision Support System (MDSS) that blends advanced road weather forecast data with observations to predict weather related roadway hazards.
 - Hazards are produced for the tactical time (next 5 minutes) as well as hourly out to 72 hours.

Pikalert[®] Background

- How did Pikalert Originate?
 - Initial development on Pikalert started circa 2008 as part of a project funded by the Federal Highway Administration (FHWA) and Research and Innovative Technology Administration (RITA). The goal of the project was “to promote safety, mobility and the environment of the nation’s surface transportation system through a new connected vehicle initiative.”-- This spawned the development of the Vehicle Data Translator
- Pikalert started as a single application called the ‘Vehicle Data Translator’ (VDT) that would ingest and processes mobile data along with ancillary weather data (e.g., radar) in order to create advanced road segment data or hazards, such as slickness potential.
 - IE. wipers on would indicate rain
- Ongoing support from several State Departments of Transportation has enabled us to grow Pikalert into a comprehensive Maintenance Decision Support System (MDSS).
 - Alaska, Colorado, Iowa, Oregon, Wyoming & Nevada
- <https://ral.ucar.edu/solutions/products/pikalert>

Pikalert[®] Evolution

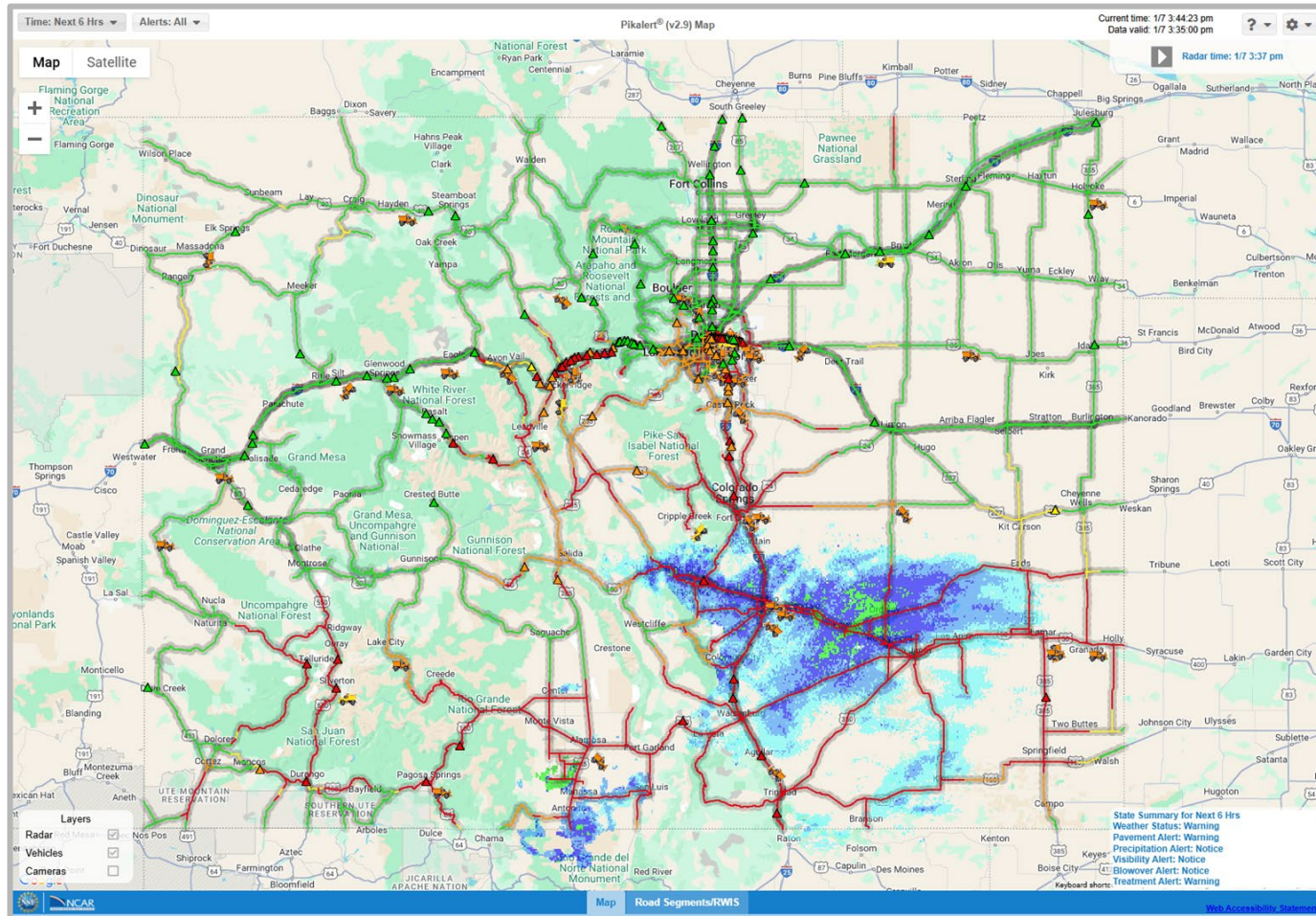
Following the development of the VDT, various other components were developed and integrated in order to build out a full MDSS system.

Primary components include--

- **DICast Forecasting Component**
 - The DICast Forecast System was integrated in order to provide highly accurate weather and road forecasts (more info later)
- **Road Weather Hazard (RWH)**
 - RWH was designed to ingest the tactical output from VDT as well as weather forecast and treatment data in order to produce weather related hazards on individual road segments (e.g., visibility, precipitation, wind)
- **Road Weather Assessment (RWA)**
 - RWA was designed to follow VDT and RWH and summarize the road weather alert and treatment information which is ultimately used in the display and (in some instances) delivered to the consumers.
- **Interactive Web display**

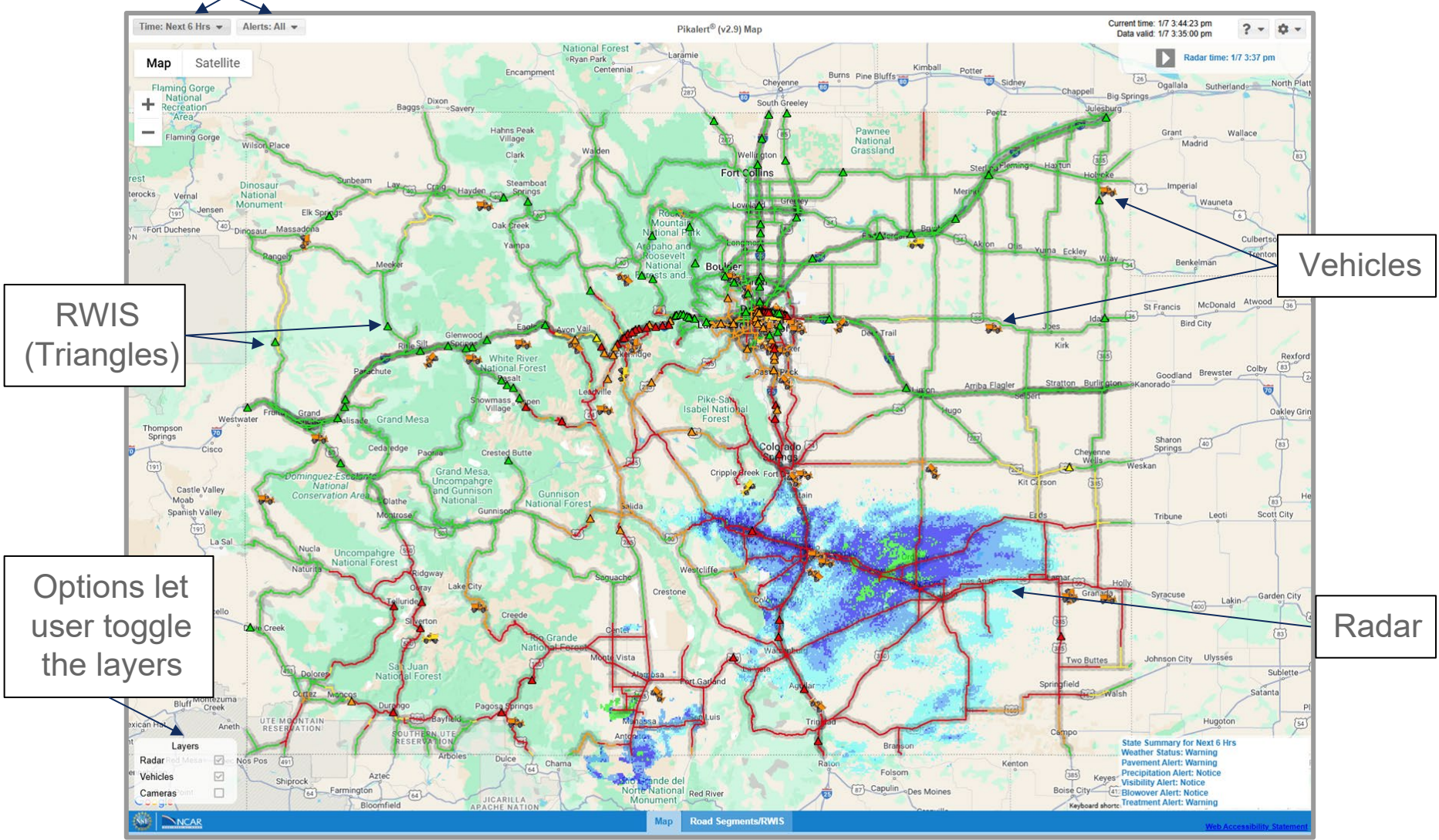
Pikalert® Web Display

<https://emdss.pikalert.org/Pikalert/?&state=colorado>

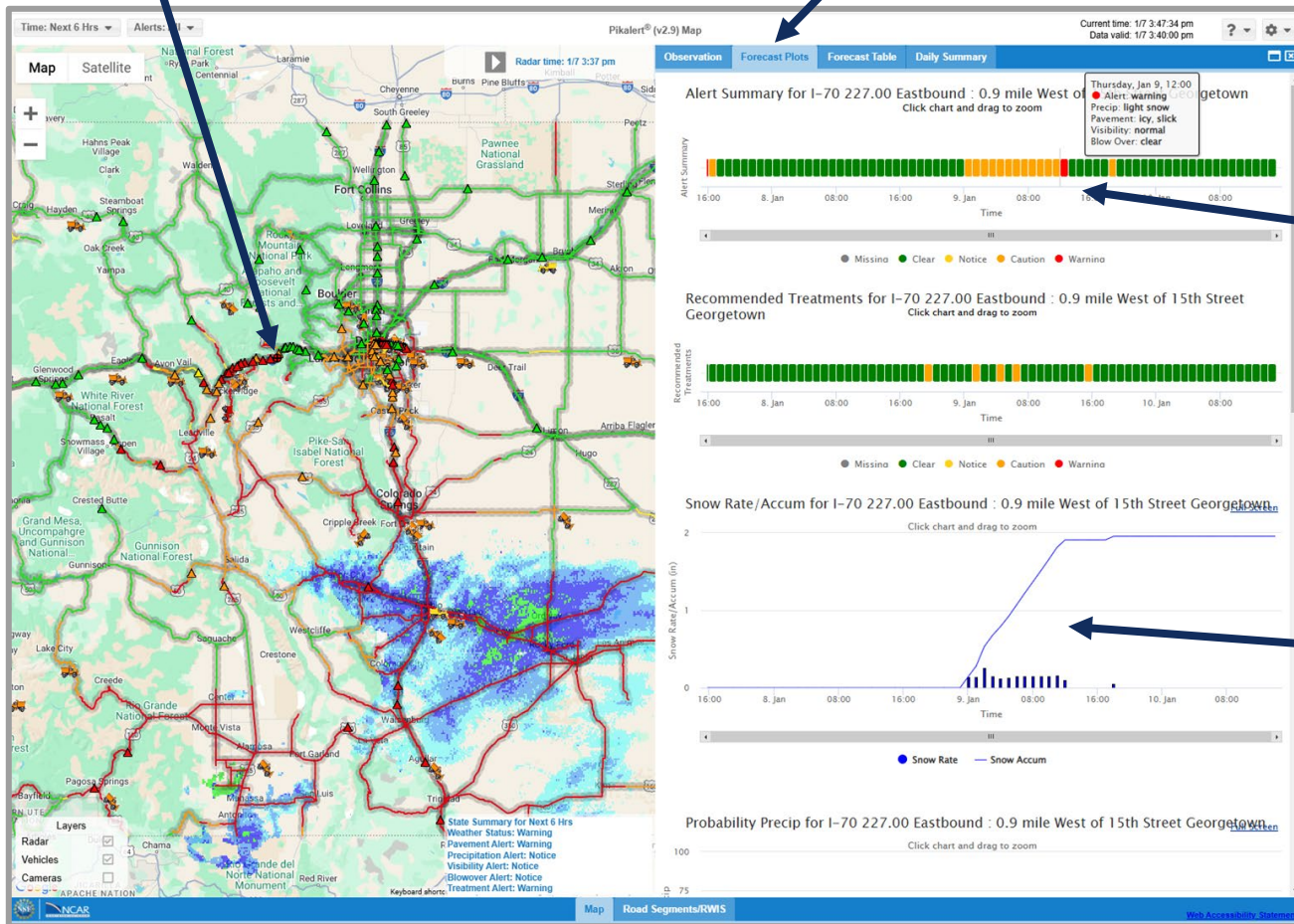


Options let user change the alert type and time

Pikalert® Web Display



Pikalert® Web Display

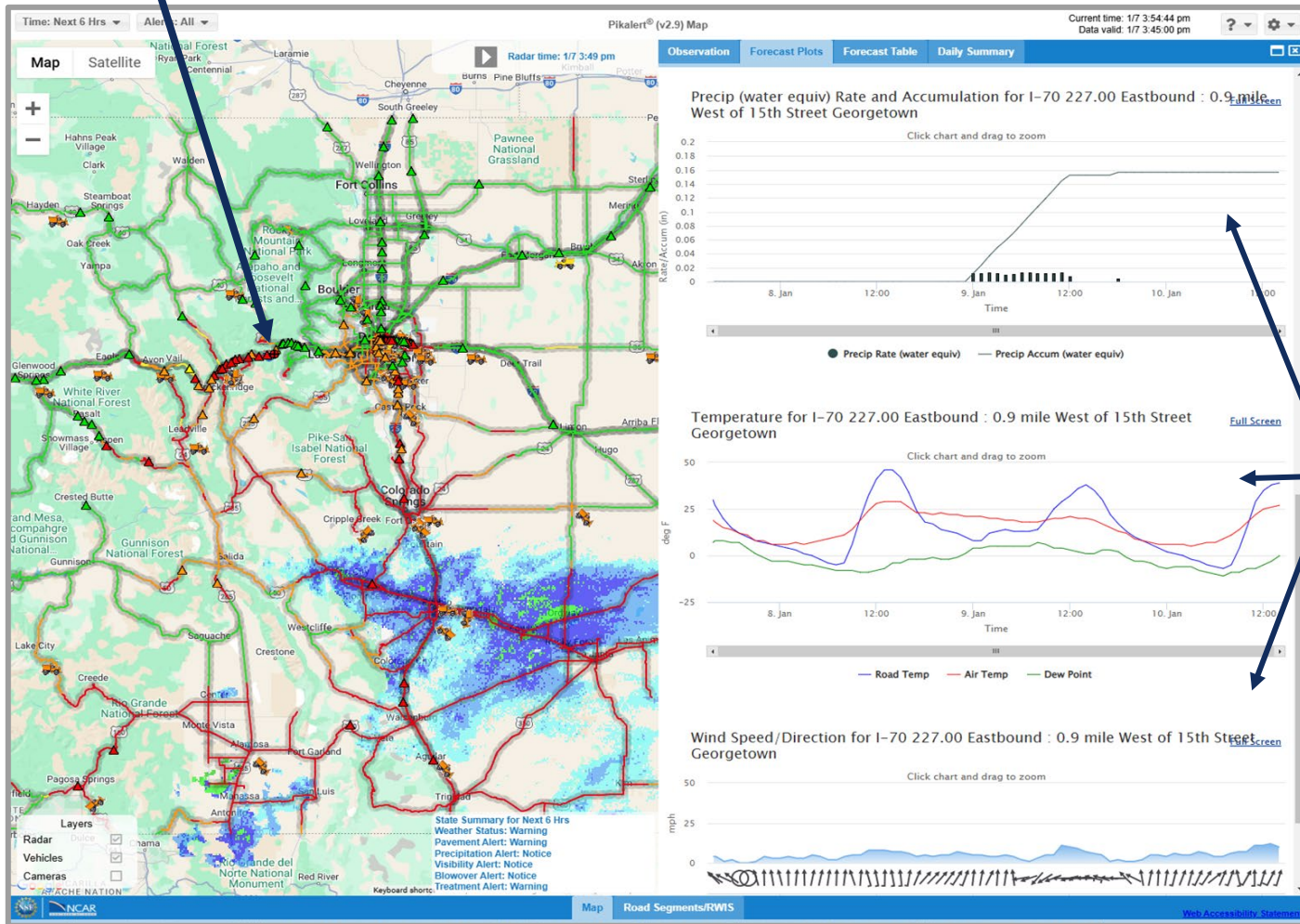


Clicking a segment or RWIS allows the user to get a more detailed picture of the forecast and alerts.

In this case, there is a 'warning' of light snow and icy roads forecasted for Jan 9th, 12:00 MST.

A plot showing the 3 day forecast of **snow rate** and **snow accumulation** is also displayed

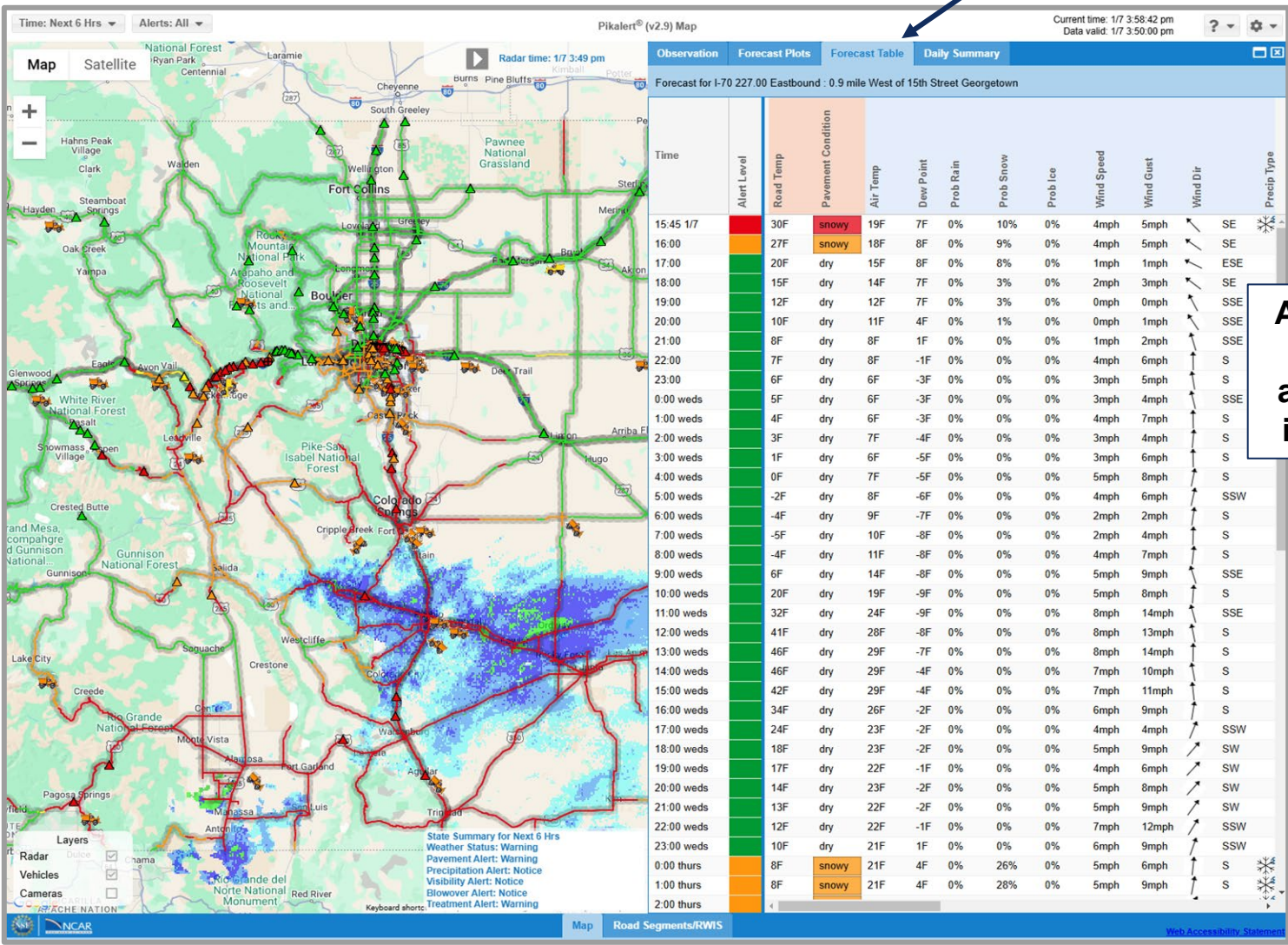
Pikalert® Web Display



Further plots showing 3-days forecasts of . . .

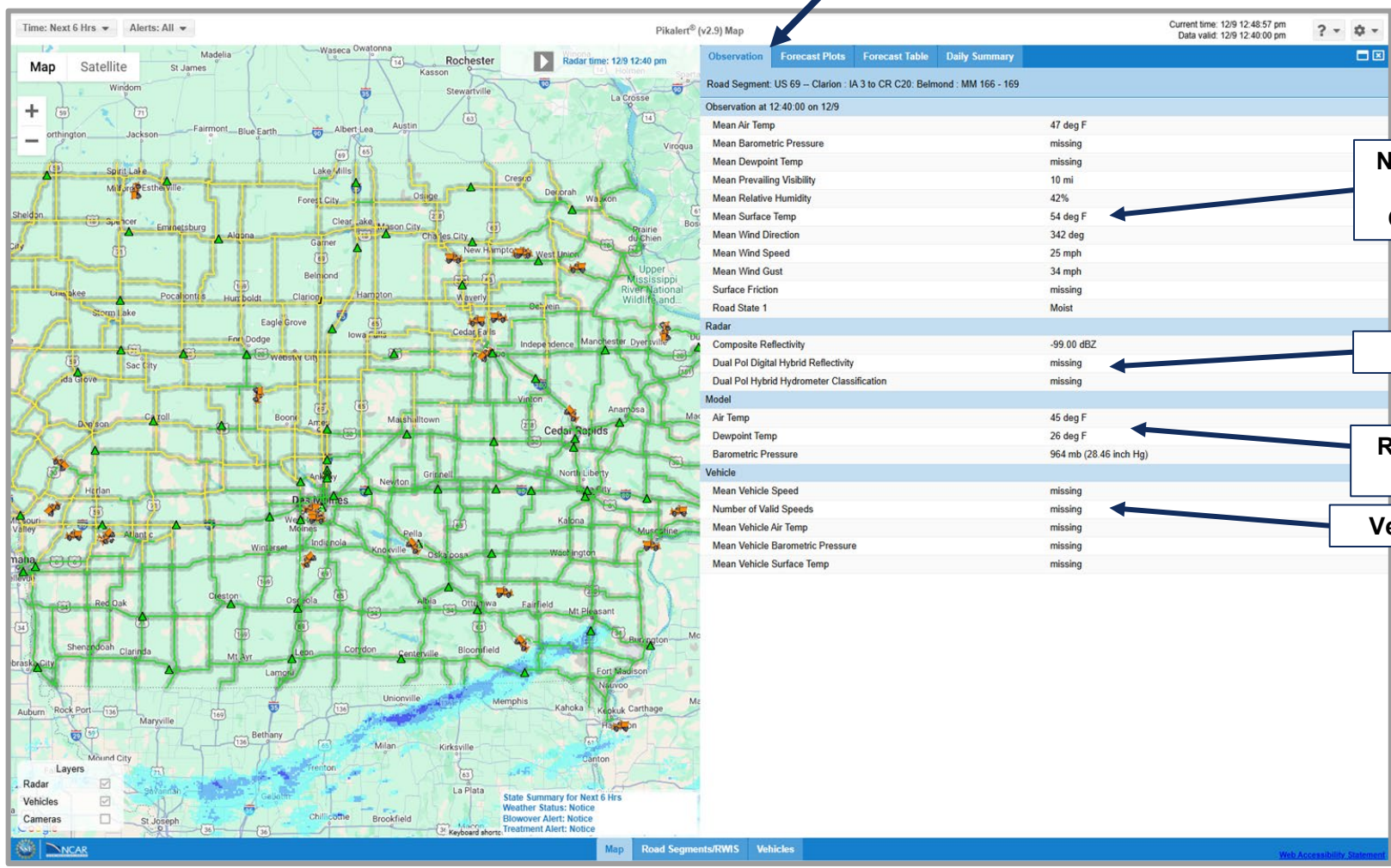
- Precip Rate / Accumulation
- Air Temp, Road Temp and Dew Point
- Wind Speed / Direction

Pikalert® Web Display

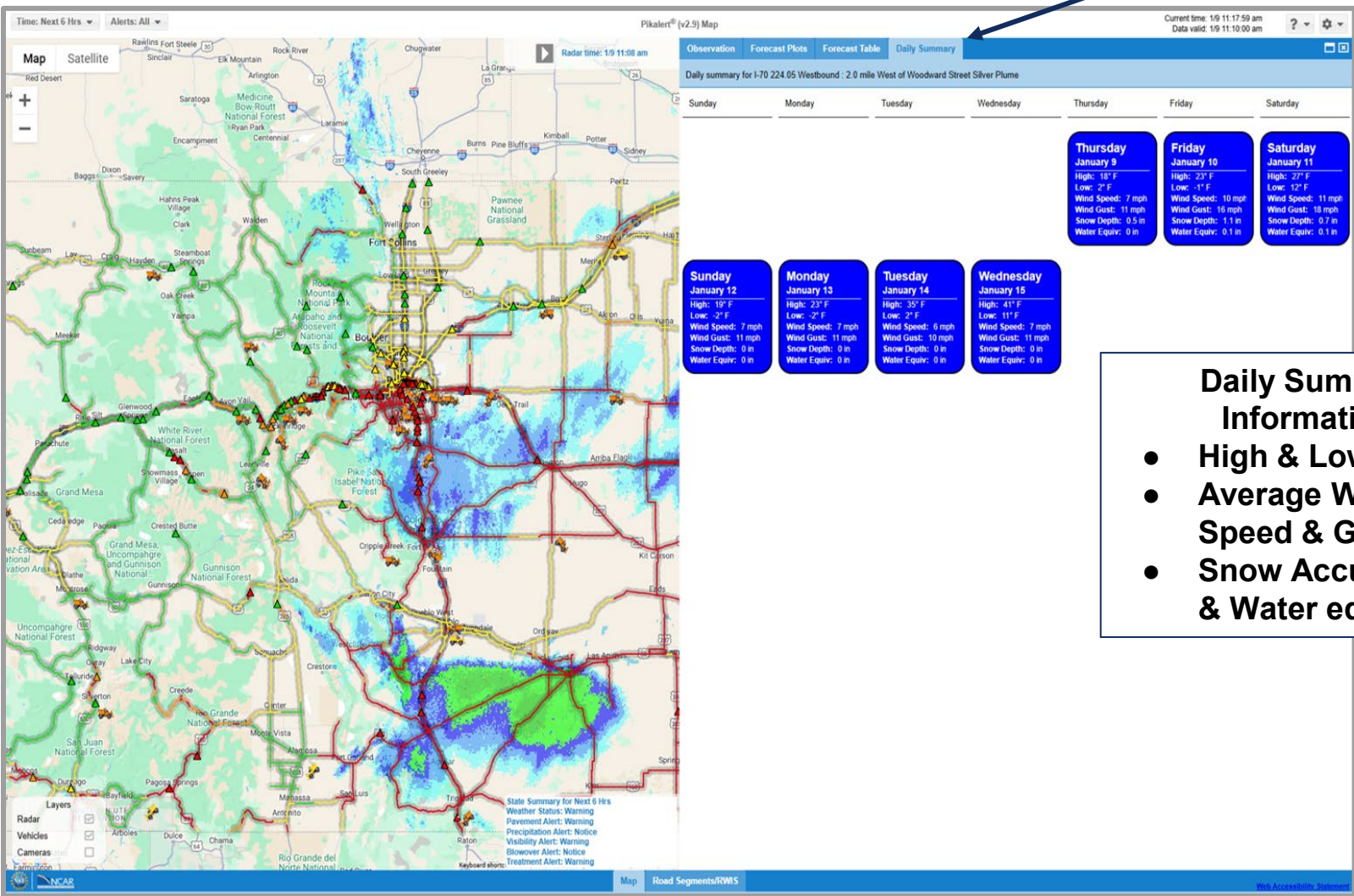


All of the forecast information can also be displayed in a table format.

Pikalert® Web Display

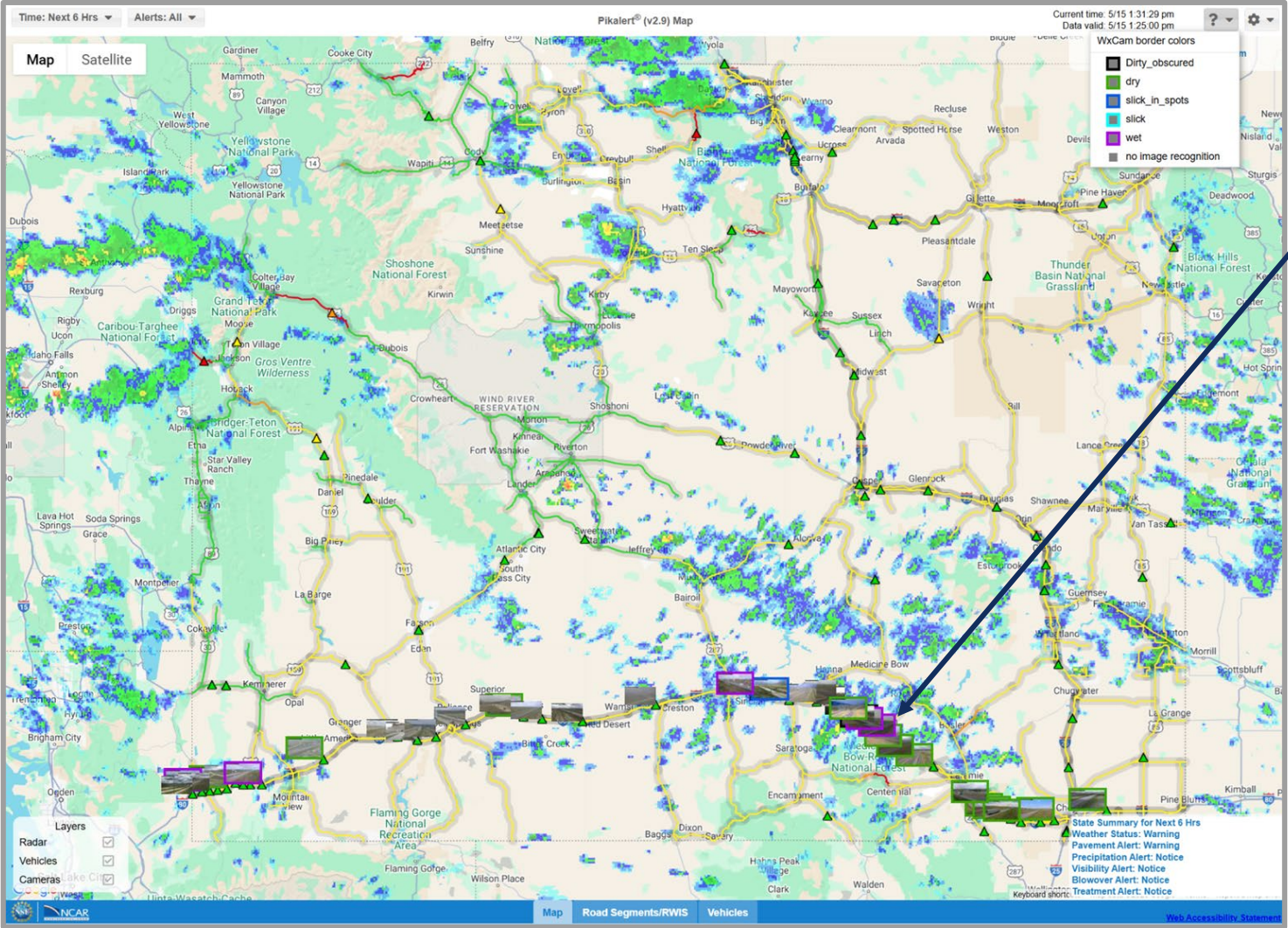


Pikalert® Web Display



- Daily Summary Information:**
- High & Low Temp
 - Average Wind Speed & Gust
 - Snow Accumulation & Water equivalent

Pikalert® Cameras

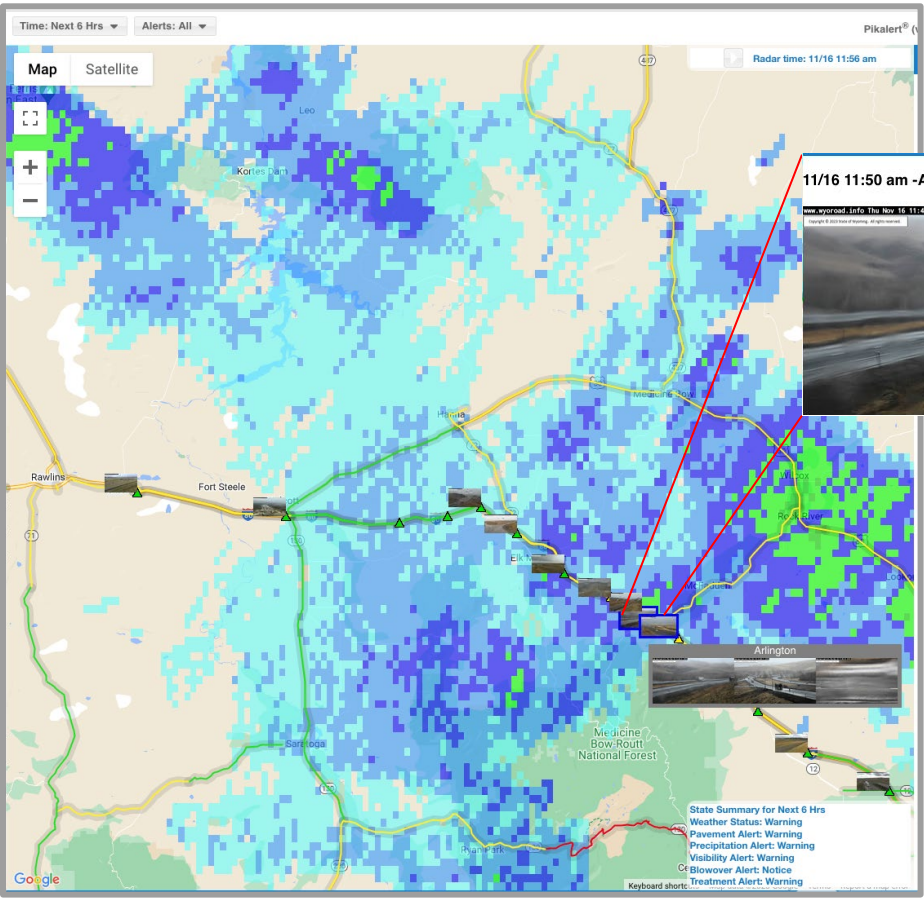


**Cameras at
RWIS locations**

Road Surface
Condition (RCS)
Categories

Dry
Wet
< 50% Covered
> 50% Covered

Pikalert® Image Recognition



Road Surface Condition (RSC) Classification
(Identified by the border color around specific images)

RSC determined by image recognition is used as another observation that can influence the near-term forecasts/alerts

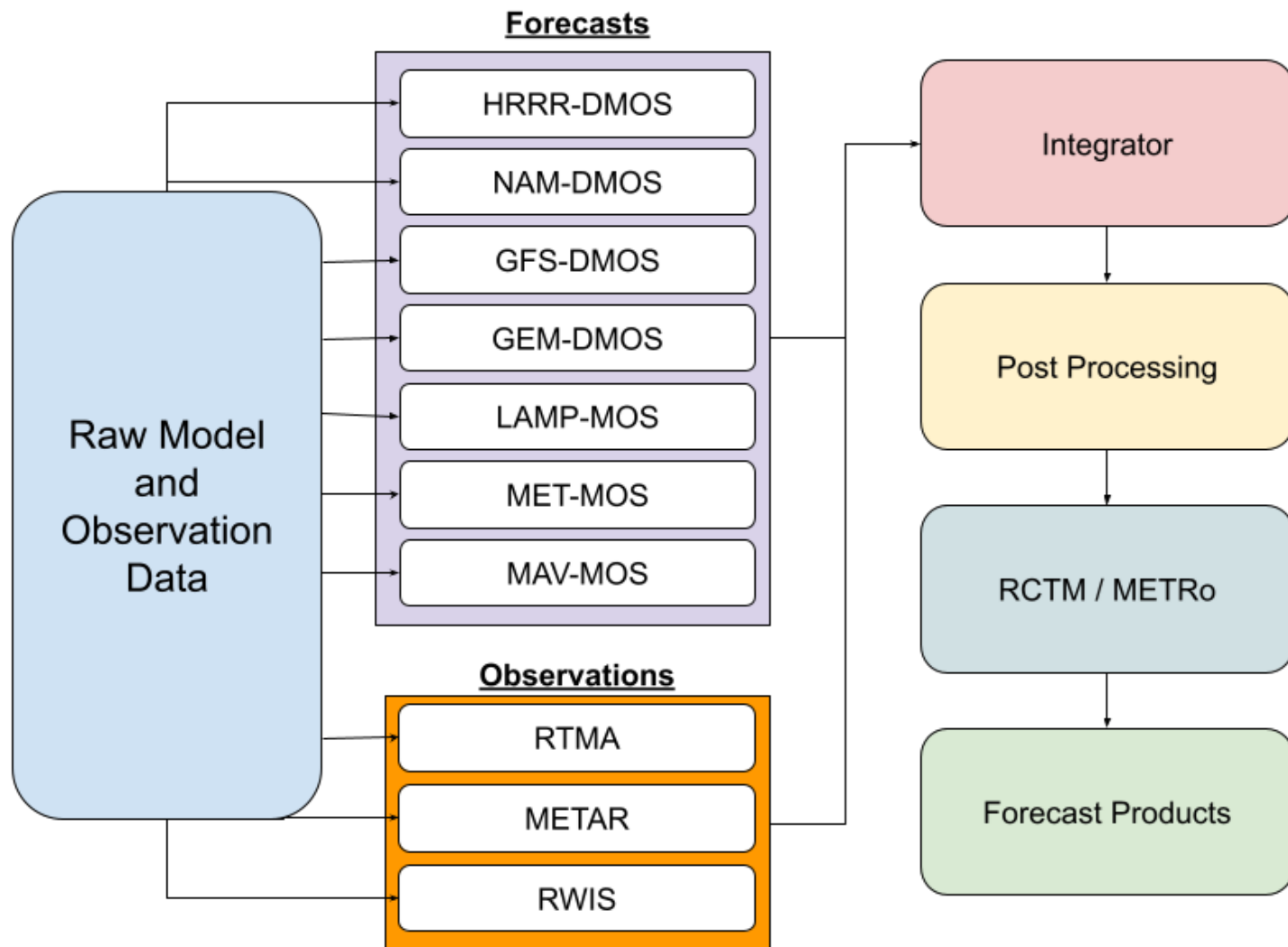
DI Cast

- DI Cast -- 'Dynamic Integrated Forecast system'
 - Statistical post processing engine that provides accurate forecast data for the Pikalert road segments.
- DI Cast integrates several Numerical Weather Prediction (NWP) models and blends them into a single forecast, weighing the models based on recent performance (against the observation).
 - Forecasts are produced for both surface stations as well as road segments

DICast: RCTM

- The Road Condition and Treatment Module (RCTM) runs on top of DICast and leverages the 'METRo' model to produce road specific forecasts (road temperature, snow build up, road surface condition, etc...).
- METRo-- “Model of the Environment and Temperature of the Roads”
- More info on RCTM and METRo later

DI Cast



Pikalert[®]: Inputs

- Data Driven System
 - “Garbage in = Garbage out”
- Observations--
 - Road Weather Information System (RWIS)
 - Real-Time Mesoscale Analysis (RTMA)
 - Surface observation stations (Metar, Synoptic reports)
 - Radar & Satellite data
 - Camera Imagery
 - Vehicle observations (Speed, Braking Status/ABS, Wiper Status, etc. . .)
- Input Forecast data--
 - Atmospheric data (Temperature, wind, precipitation, etc. . .)
 - Surface data (Road Temperature, Road Surface Condition, treatment recommendations)
- Need the ability to scale this infrastructure to new domains
 - IE. Scale up or down the system based on sponsor requirements

Pikalert[®] In The Cloud

- It was determined that leveraging cloud computing would allow us to easily scale the system and be more flexible to the needs of our sponsors.
- Amazon Cloud Tools:
 - Amazon Web Services (AWS) Console
 - Management dashboard with all relevant AWS info (if you can find it)
 - EC2 (Elastic Compute Cloud)
 - Acts like an in-house machine
 - S3 (Simple Storage Service)
 - Used to archive large volumes of data
 - Amazon Rekognition (image recognition)
 - Used to determine road surface status from roadside cameras
 - CloudWatch
 - Used to monitor system health and alerts

AWS Console



Console Home [Info](#)

[Reset to default layout](#)[+ Add widgets](#)

Recently visited [Info](#)

- Billing and Cost Management
- CloudWatch
- Amazon Rekognition
- S3
- EC2
- Route 53
- Lambda
- Simple Notification Service
- IAM
- Support
- Simple Queue Service
- EC2 Global View

[View all services](#)

Applications (0) [Info](#)

[Create application](#)

Region: US East (N. Virginia)

Select Region

us-east-1 (Current Region)

< 1 >

Name | Description | Region | Originati. | Star

No applications
Get started by creating an application.

[Create application](#)[Go to myApplications](#)

Welcome to AWS

- Getting started with AWS**
Learn the fundamentals and find valuable information to get the most out of AWS.
- Training and certification**
Learn from AWS experts and advance your skills and knowledge.
- What's new with AWS?**
Discover new AWS services, features, and Regions.

AWS Health [Info](#)

- Open issues
0 Past 7 days
- Scheduled changes
0 Upcoming and past 7 days
- Other notifications
0 Past 7 days

[Go to AWS Health](#)

Cost and usage [Info](#)

Current month costs
\$509.60
↑ 16% compared to last month for same period

Forecasted month end costs
\$990.54
↓ 0% compared to last month's total costs

Savings opportunities

[Enable Cost Optimization Hub](#)

Cost (\$)



■ Savings Plans for Compute usage
■ EC2 - Other ■ Rekognition ■ S3
■ EC2 - Compute ■ Others

[Go to Billing and Cost Management](#)

AWS EC2

- Pikalert & DICAST systems run on a 'm6i.2xlarge' (ALL STATES)
 - 8 virtual CPU
 - 32G Memory
- This serves as our operational 'machine' where the entire system is run.
 - Acts just like any inhouse machine
- EC2s are flexible to scaling. Unlike an inhouse machine, we can quickly add more CPU or RAM
- Pricing:
 - On Demand: \$0.384 per hour (~\$290 per month)
 - We have a AWS reserved instance savings plan, which locks in our EC2 for a few years, but reduced our costs to~**\$53 per month**
 - Cost for storage volume is significant. Current Pikalert EC2 has a 3TB EBS volume that costs~**\$245 per month**

AWS S3

- AWS S3 serves as our data archival and/or transfer tool
 - Costs: ~\$0.023 GB per month (\$23 per TB per month)
- S3 provides various storage 'areas' that can be used to decrease pricing. IE. 'S3 Glacial Deep Archive' is ~1\$ per TB per month, but you can only access your data once or twice a year.
- For Pikalert, a lot of our 'infrequently accessed' data uses 'S3 Intelligent- Tiering', which pushes your data to 'deeper' and cheaper archives as the data ages off
 - This makes it slower and *slightly* harder to access, but has a significant impact on costs
- Pikalert costs ~\$220 per month for nearly 20TB of total storage
- Small costs associated with data transfer

AWS Rekognition

- AWS Rekognition is cloudbased computer vision service.
- Within Pikalert, Rekognition is used to determine road surface condition (primarily snow coverage) based on roadside images.
- In order to be used for the road condition task, thousands of images were manually labeled (dry, wet, partially covered, fully covered). The model was retrained on this dataset in order to predict these classes.

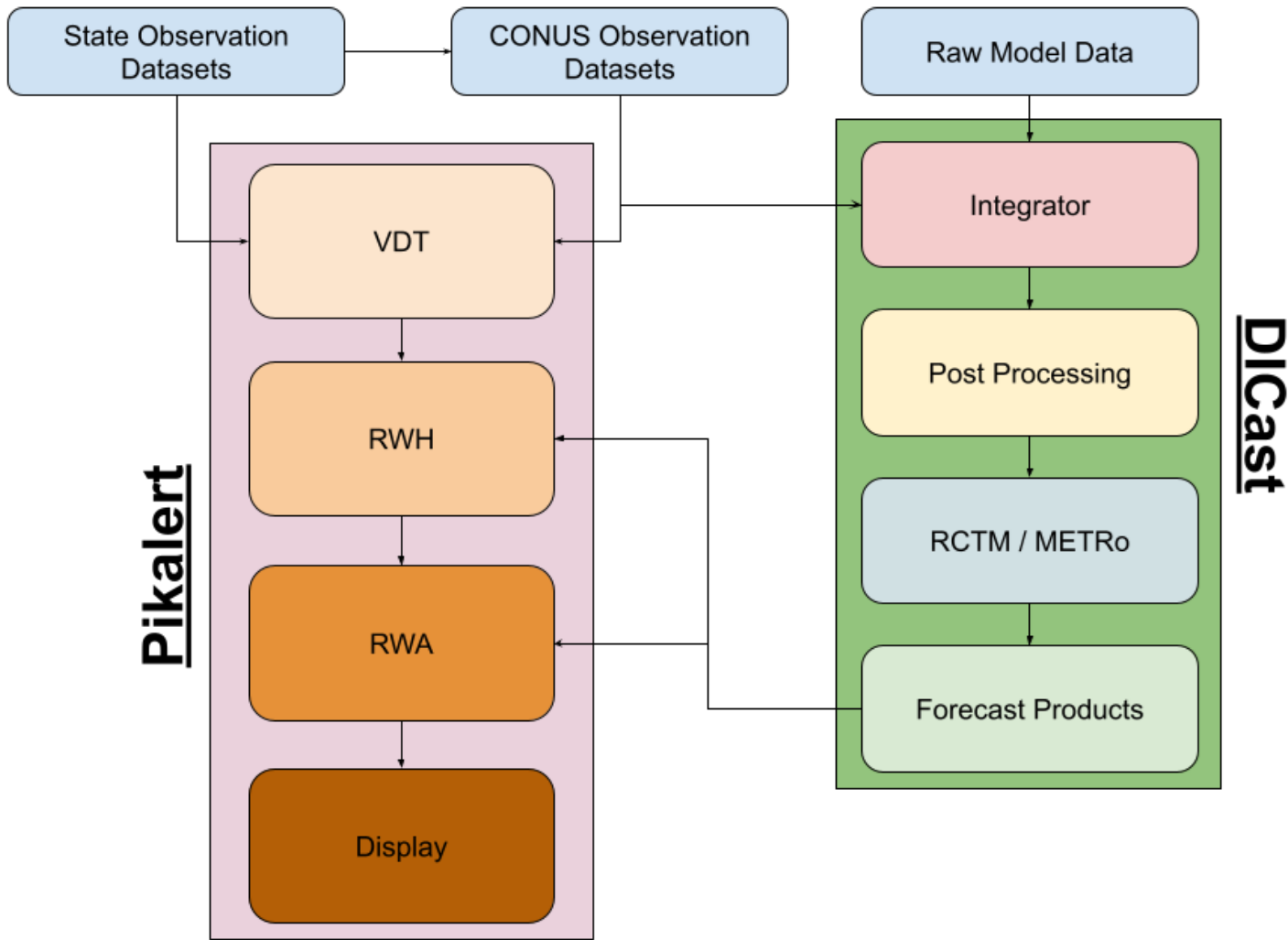
AWS Rekognition Costs

- Costs are determined based on model 'up time'
- Current realtime logic --
 - a. Download images from state DOTs
 - b. Spin up model (~3-5 min)
 - While model is spinning up, copy specific images to S3 (and perform other image processing tasks)
 - c. Run model on images in S3 (few seconds)
 - d. Shut down model
 - This is critical in order to keep costs low
- This is done every 10 minutes (daylight hours) for 8 sites in Oregon and 20+ sites in Wyoming
- \$8 per day -- ~\$250 per month
 - a. Additional costs associated with training new models

AWS CloudWatch

- AWS Cloudwatch collects, visualizes and analyzes metrics, logs and events in real time.
- Within Pikalert, this is used to-
 - Detect system anomalies, such as high costs or CPU usage
 - Monitor system performance (is our EC2 meeting our needs?)
 - Monitor Rekognition 'up -time'
 - Alerts were set up on the operational Rekognition models that will notify us if the model runs for longer than expected.
 - Cloudwatch can trigger an event which will call an AWS Lambda function to turn off a Rekognition model if it has run for longer than some predefined time.
- Negligible costs with the current Pikalert infrastructure

System Overview



System Code & Languages

- Most of the ‘heavy-lifting’ processes are written in C++ for efficiency and speed.
 - Python ‘wrappers’ are used to put together command line arguments (data inputs, configuration files, etc)
- Most of the data processing tools are written in Python, because memory management and speed isn’t as big of a concern.
- Static files in various formats
 - .json, .py, .yaml (cloud specific), .cfg (asc), .params (asc)
- System is driven by a crontab
 - processes triggered at specific times

Example Crontab

```
#####
#
# OREGON Processing
#
#####
# Get the raw oregon image and rwis data and do some processing
#
*/5 * * * * ep.py -L /d1/vii/system/lock/run_or_data_ingest -E 270 run_or_data_ingest.py >& /dev/null ; ep.py -L /d1/vii/system/lock/run_or\
_rwis_to_madis -E 270 run_or_rwis_to_madis.py >& /dev/null
*/10 * * * * ep.py -L /d1/vii/system/lock/run_or_image_data_ingest -E 570 run_or_image_data_ingest.py >& /dev/null

#
# Copy the Oregon data to s3
#
59 23 * * * ep.py -L /d1/vii/system/lock/run_or_images_to_s3 -E 1800 run_or_images_to_s3.py >& /dev/null
15 00 * * * ep.py -L /d1/vii/system/lock/run_or_rwis_to_s3 -E 1800 run_or_rwis_to_s3.py >& /dev/null
20 00 * * * ep.py -L /d1/vii/system/lock/run_or_files_s3 -E 3600 run_or_files_to_s3.py >& /dev/null

#
# Copy the Oregon image label output to s3
#
55 23 * * * S3_utils_archive.py OR_img_labels_to_s3 -l /d1/vii/oregon/data/log/OR_image_labels_to_s3 >& /dev/null

#
# Copy the Oregon rwh,vdt,seg_stats data to s3
#
5 03 * * * ep.py -L /d1/vii/system/lock/run_or_vdt_rwh_to_s3 -E 7200 run_or_rwh_vdt_to_s3.py >& /dev/null

# Run the dual-pol radar conversion to netCDF
00-55/05 * * * * ep.py -L /d1/vii/system/lock/run_or_Nids2NetCDF -E 300 run_or_Nids2NetCDF.py >& /dev/null ; ep.py -L /d1/vii/system/lock/r\
un_combine_HHC_or_radar_nc -E 300 run_combine_HHC_or_radar_nc.py >& /dev/null

# Run OR processing
03,08,13,18,23,28,33,38,43,48,53,58 * * * * ep.py -L /d1/vii/system/lock/ep_run_or_process -E 290 ep_run_or_processes.py >& /dev/null

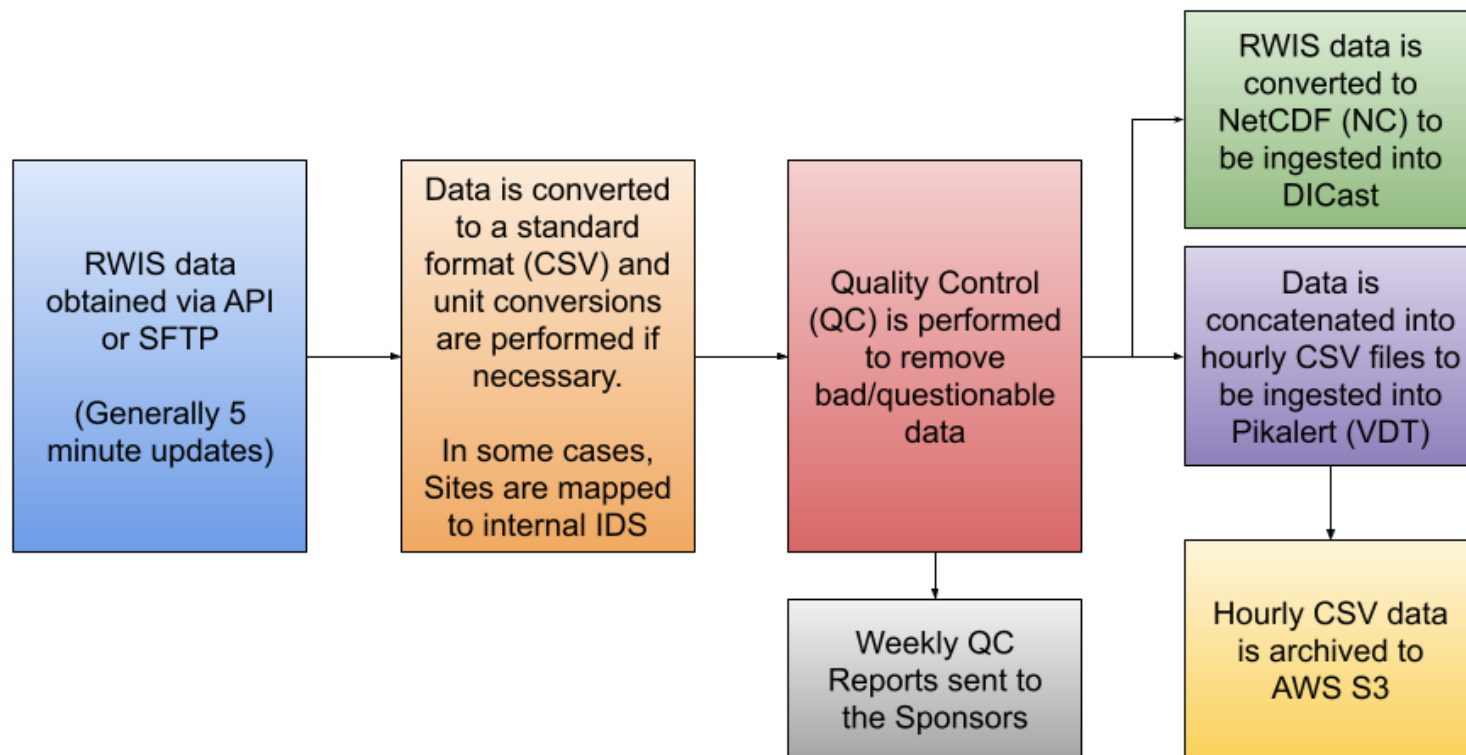
#
# Scrub some s3 areas
#
10 0 * * * ep.py -L /d1/vii/system/lock/run_or_s3_scrub -E 1800 run_or_s3_scrub.py >& /dev/null
```


Pikalert observation processing

- Each state provides some or all of the following observations
 - RWIS (Wx & Road observations)
 - 5 minute updates
 - Camera images
 - 5-10 minute updates
 - AVL data (Vehicle / snowplow observations)
 - 5-10 minute updates
 - Other (road closures, incidents, etc)
- Each obs feed goes through a process to standardize the data
 - Data conversion to format which can be ingested into the system

Pikalert RWIS processing

- Each state either provides an API or sends the data to us via SFTP



Pikalert image processing

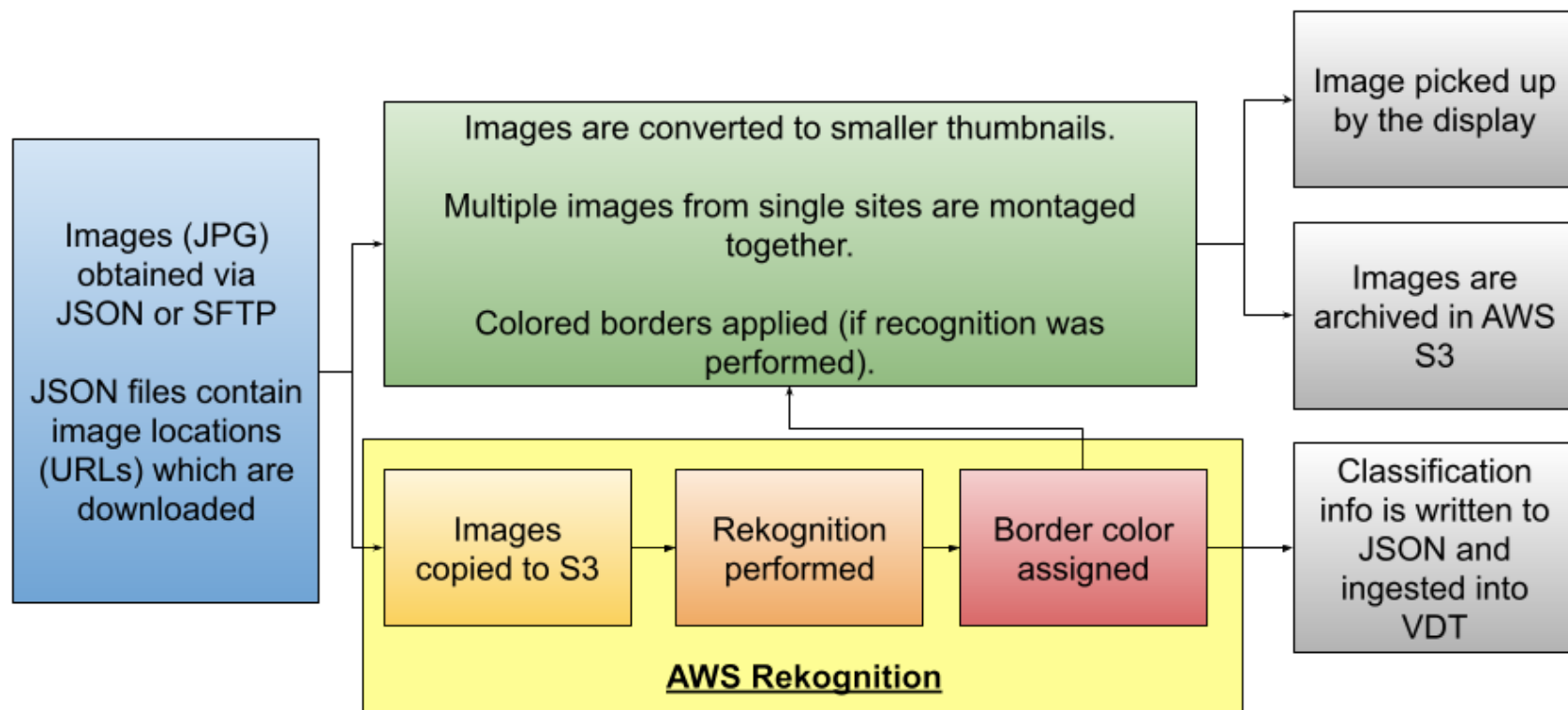
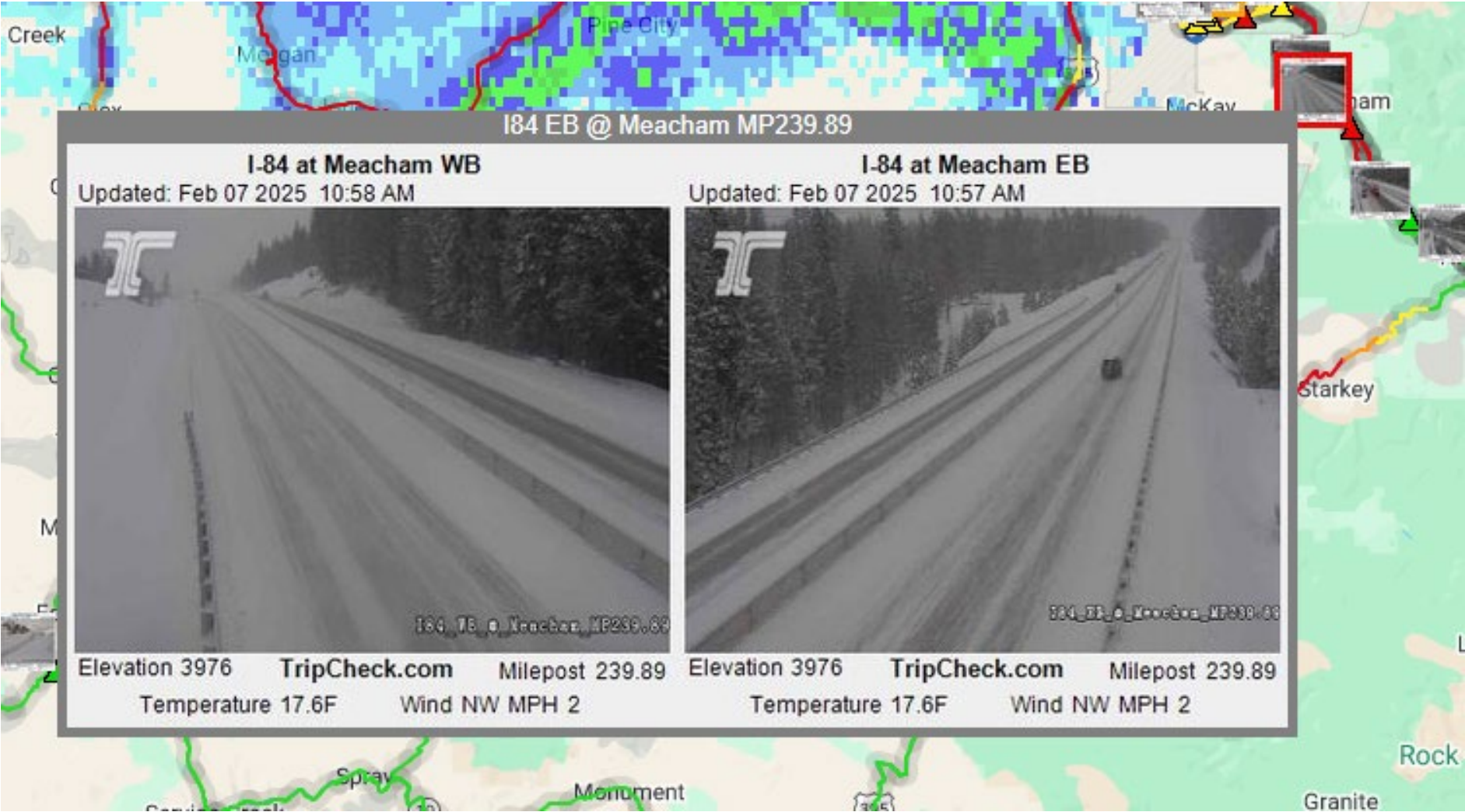
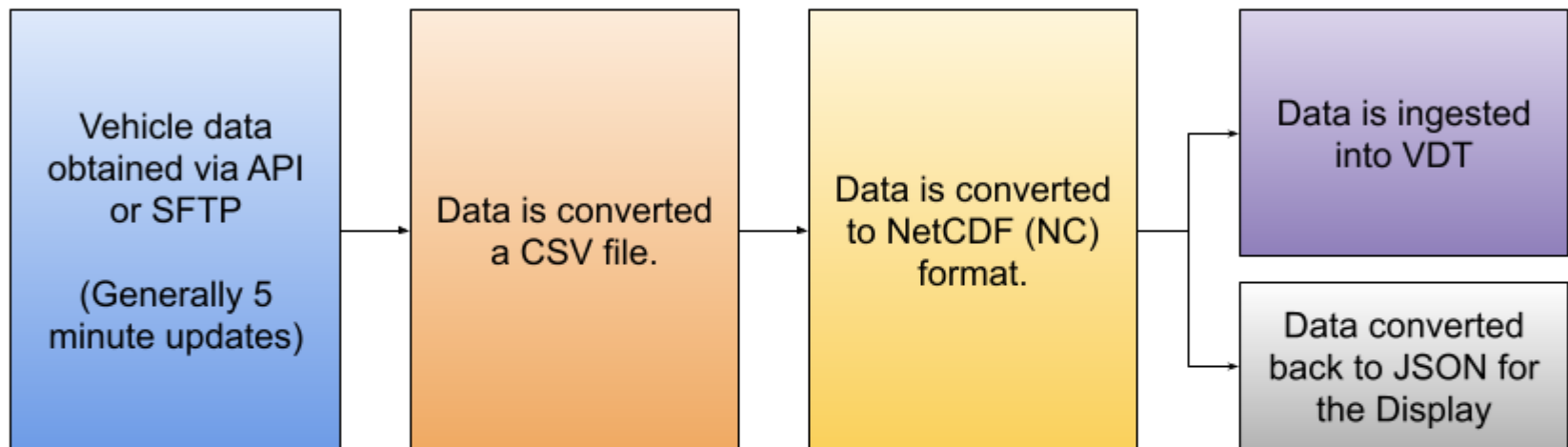


Image Examples



Pikalert Vehicle Obs Processing



- Current state sponsors provide data for (on average) less than 50 vehicles.
 - This data is often incomplete or lacking useful information
- Oregon and Alaska are working to provide much more data (more than 1,000 vehicles by 2026).

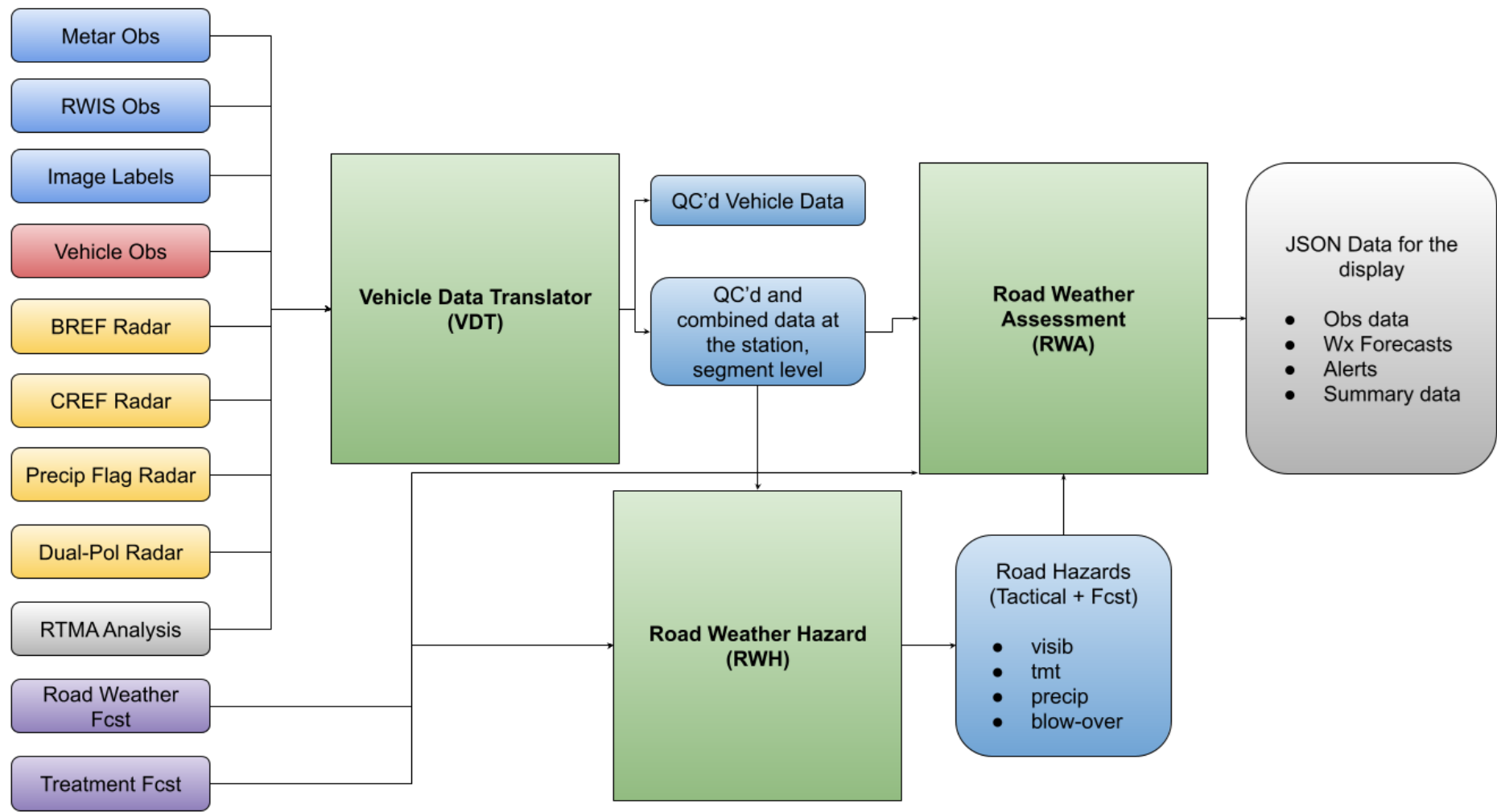
Raw JSON

```
],
"features": [
  {
    "attributes": {
      "OBJECTID": 2,
      "LABEL": "A33676",
      "XPOSITION": -93.596726649,
      "YPOSITION": 41.657891089,
      "HEADING": 93.2,
      "VELOCITY": 3.107,
      "ROADTEMP": null,
      "AIRTEMP": null,
      "SOLIDMATERIAL": null,
      "LIQUIDMATERIAL": null,
      "PREWETMATERIAL": null,
      "SOLIDRATE": null,
      "LIQUIDRATE": null,
      "PREWETRATE": null,
      "FRONTFLOWSTATE": 9999,
      "RIGHTWINGFLOWSTATE": 9999,
      "LEFTWINGFLOWSTATE": 9999,
      "UNDERBELLYFLOWSTATE": 9999,
      "SOLIDRUNTOTAL": null,
      "LIQUIDRUNTOTAL": null,
      "PREWETRUNTOTAL": null,
      "SOLIDSETRATE": null,
      "LIQUIDSETRATE": null,
      "PREWETSETRATE": null,
      "BLASTMODE": null,
      "TOTAL_TRUCK_COUNT": 2,
      "ACTIVE_MATERIAL": "NONE",
      "ROUTE_NAME": "NE 54TH AVE E",
      "LOGDT": 1744700159000,
      "MODIFIEDDT": 1744718159000,
      "MODIFIEDDT UTC OFFSET": "-05:00",
      "REST_UPDATED": 1744718354000,
      "REST_UPDATED UTC OFFSET": "-05:00",
      "COST_CENTER": null,
      "MAINT_DISTRICT": null,
      "GARAGE_NAME": null
    },
    "geometry": {
      "x": -93.596726649,
      "y": 41.657891088999996
    }
  },
  {
    "attributes": {
      "OBJECTID": 3,
      "LABEL": "A35982",
      "XPOSITION": -91.20077202,
```

Processed JSON

```
{
  "data_time": "202504151200",
  "districts": [
    {
      "data_time": "202504151200",
      "display_name": "iowa",
      "district_name": "iowa",
      "max_lat": 43.82699966430664,
      "max_lon": -90.03399658203125,
      "min_lat": 40,
      "min_lon": -96.40299987792969,
      "vehicles": [
        {
          "auto_brake": "-9999",
          "elevation": "-9999",
          "heading_deg": "241.2",
          "humidity": "-9999",
          "id": "A35982",
          "lat": "41.2770",
          "lon": "-91.2010",
          "obs_time": "1744718160",
          "pressure": "-9999",
          "road_temp_f": "-9999.0",
          "speed_mph": "7",
          "spreader": "-9999",
          "temp_f": "-9999.0",
          "traction_control": "-9999",
          "wiper_status": "-9999"
        },
        {
          "auto_brake": "-9999",
          "elevation": "-9999",
          "heading_deg": "93.2",
          "humidity": "-9999",
          "id": "A33676",
          "lat": "41.6580",
          "lon": "-93.5970",
          "obs_time": "1744718159",
          "pressure": "-9999",
          "road_temp_f": "-9999.0",
          "speed_mph": "3",
          "spreader": "-9999",
          "temp_f": "-9999.0",
          "traction_control": "-9999",
          "wiper_status": "-9999"
        }
      ]
    }
  ]
}
```

Pikalert High Level Diagram



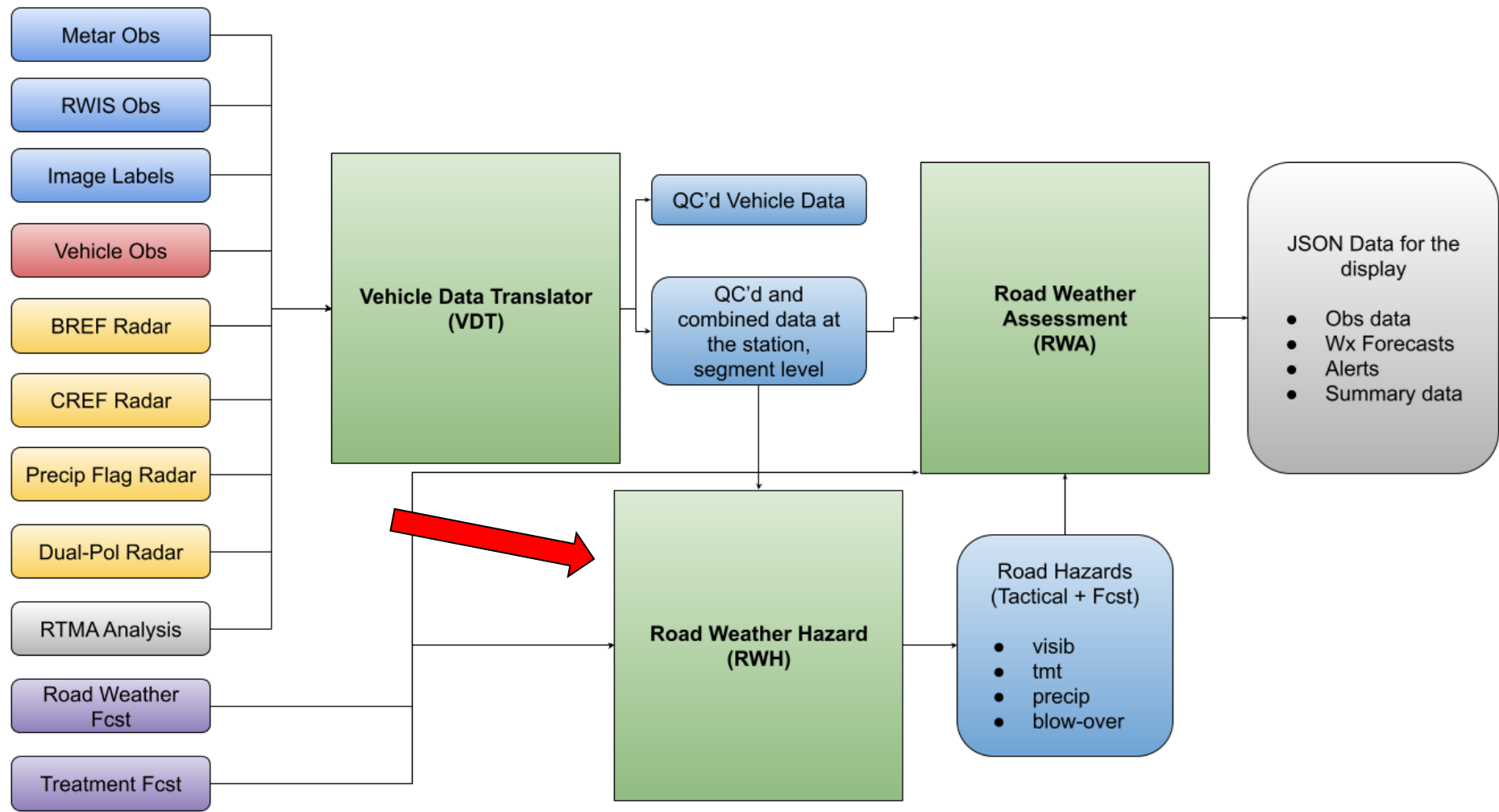
Pikalert VDT

- VDT is the application that started Pikalert. It was created back in 2008 to use vehicle data as well as standard weather observations to derive road and atmospheric hazard condition assessments.
- C++
- Relevant vehicle observations include. . . .
 - Braking info
 - ABS activity (Automated Braking System)
 - Brakes on/off (down to the wheel)
 - Vehicle heading & speed
 - Acceleration info (in both x/y directions)
 - Lights (including fog lights and brights/high beam)
 - Stability / Traction Control
 - Steering Angle / Rate of change
 - Wipers

Pikalert VDT Cont.

- As additional weather data becomes available, it is integrated and distilled down to the road segments within the VDT application
- Each state has specific parameters that determine what obs are used for each segment--
 - Radius of influence (ROI): Used to determine if a surface observation is close enough to influence the segment
 - Separate ROI for image label data
 - Time cutoff: Used to determine if the observation was recent enough to influence the segment
 - Maximum distance to Radar / RTMA grid cells
 - QC parameters
 - Used to make sure that all relevant data is in agreement in order for it to influence the segment

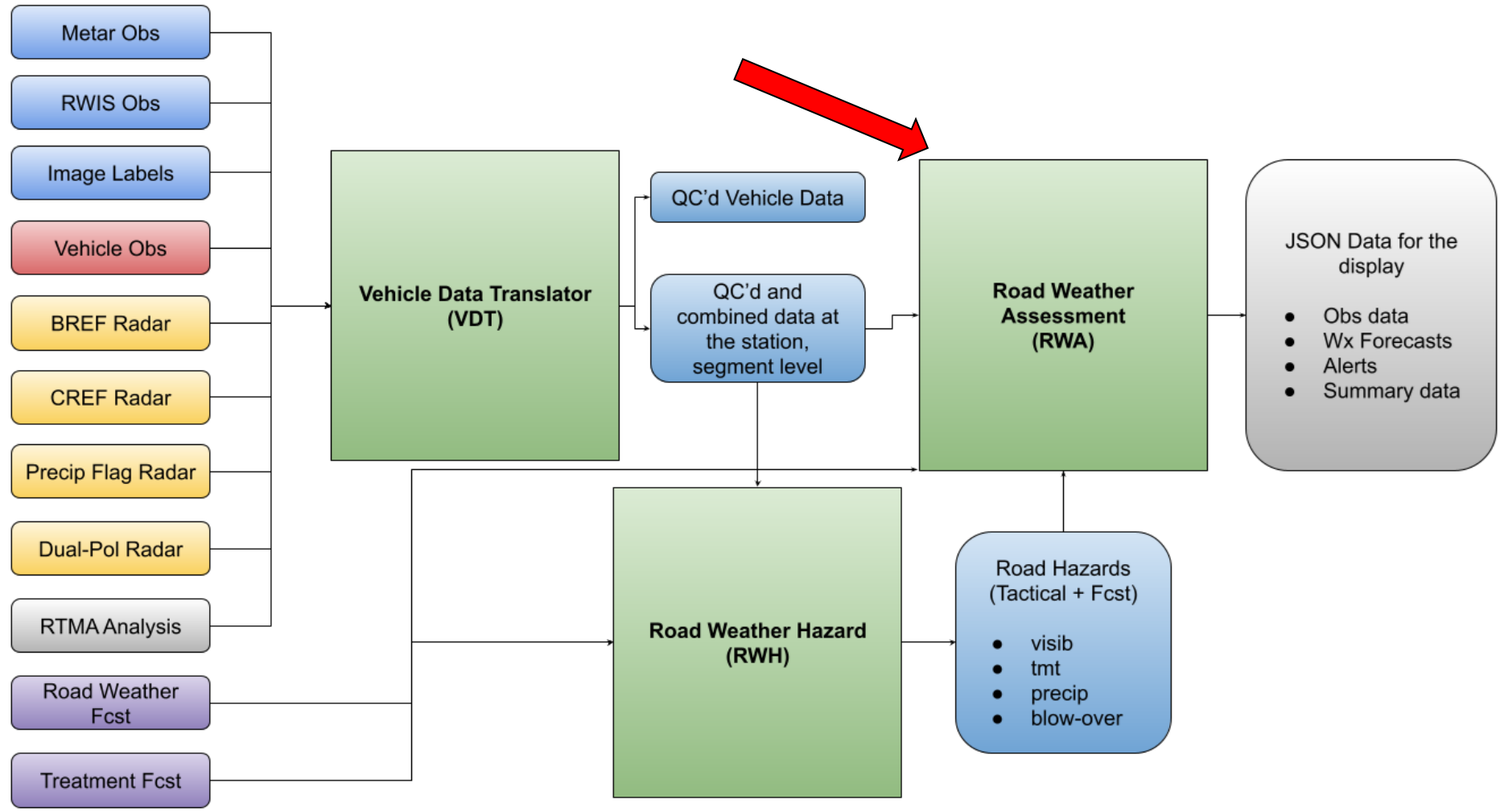
Pikalert



Pikalert RWH

- The Road Weather Hazard application was created to ingest the segment level data (from VDT), merge that with D1Cast weather and treatment forecasts, and produce segment level hazard predictions
- C++
- Hazards
 - Precip Type and Precip Intensity
 - Pavement Condition
 - dry, wet, snow covered, ice covered, hydroplane, black ice
 - Visibility
 - Normal, low, heavy rain, heavy snow, blowing snow, fog, haze, dust, smoke
 - Pavement Slickness (normal or slick)
 - Blow Over potential
 - Regular vehicle, light high-profile, loaded high-profile, pickup w/ trailer

Pikalert



Pikalert RWA

- After some time it was determined that a display was necessary in order to visualize the forecasts and alerts, as well as to serve as a sanity check. The Road Weather Assessment application was created in order to serve the display with the necessary data.
- C++
- RWA Ingests the output from VDT and RWH and produces a suite of JSON files which are ingested by the display software.
- Output from RWA is delivered to specific sponsors who wish to ingest it into their own visualization software.
 - This data is most often uploaded to an AWS S3 bucket which the sponsor has access to.

Pikalert Archival Processes

- A vast majority of the Pikalert and DICAST data is archived to AWS S3 for various purposes, including . . .
 - System backup
 - Case studies
 - Machine learning
- S3 backups are primarily managed via python scripts, most often run once per day.

```
#!/usr/bin/env python

import os
import sys_path

BUCKET = 'pikalert-oregon'

#
# Specify the profile used for executing sync commands (leave empty string('') for default profile)
#
PROFILE = ''

DIR_MAP = {
    #
    # Image label files (json)
    #
    "/d1/vii/oregon/data/processed/image_labels" : {'s3_folder' : 'img-rec-labels',
                                                    'scrub' : None}
}
```

Pikalert Archival Processes Cont.

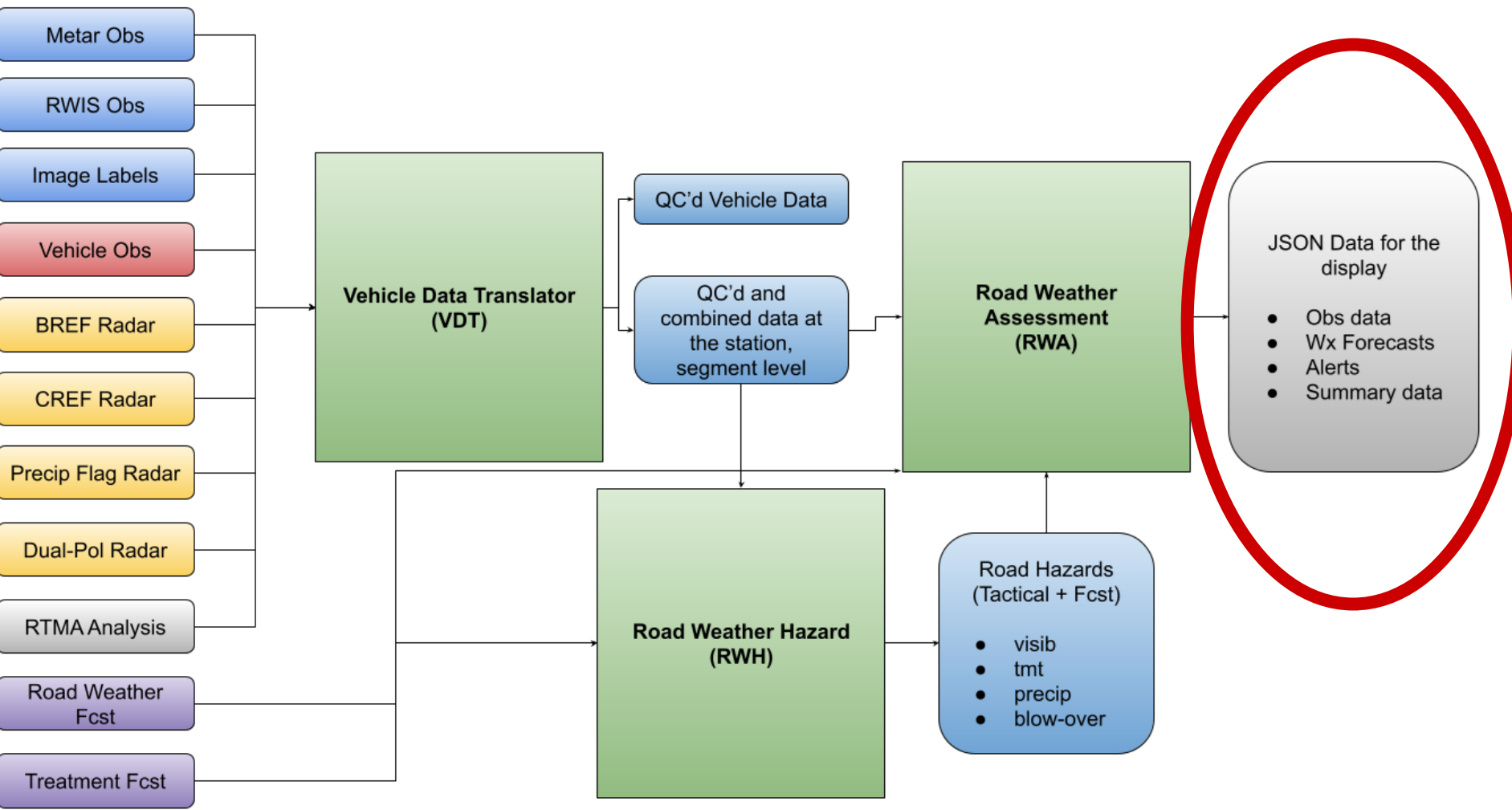
- Data that isn't often accessed is stored using AWS Intelligent Tiering, which ages data into 'deeper' archives, making it slower to access but much cheaper to store.
- Scrubbing policies can be set up on entire buckets using AWS S3 lifecycle rules.
 - For finer control, python scripts are used to remove specific files or folders after a certain date.
- S3 bucket's can also be 'mounted' to the EC2, allowing you to interact with them as if it was just another folder on the machine.
 - "ls", "cp", etc. . . .

Pikalert Display Software

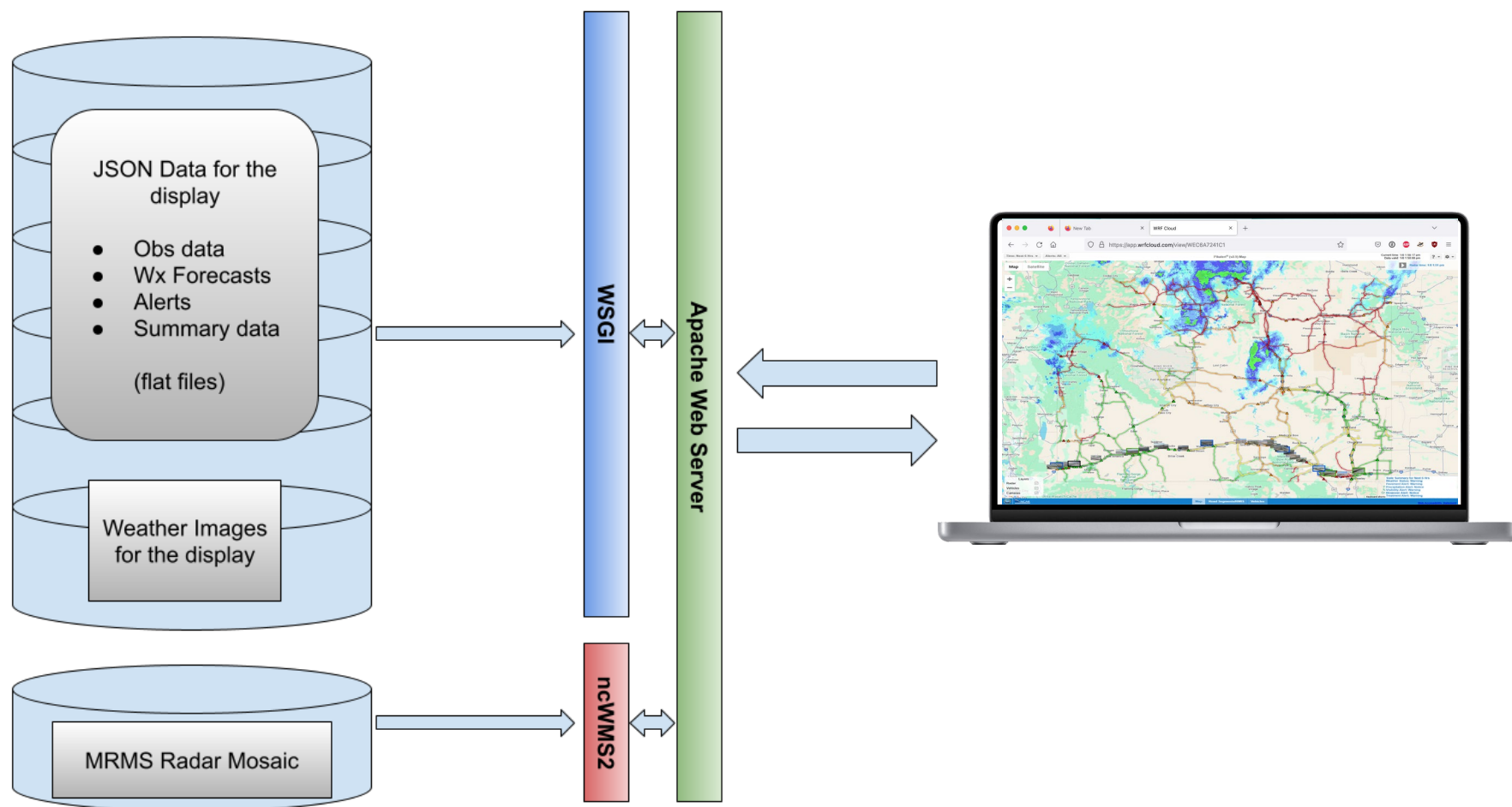
Software Components

- Browser-based web application
- Display coded on ExtJS by Sencha
 - JavaScript component framework
 - Commercially licensed
 - Cross-browser compatibility
 - Convenient and flexible data model architecture
 - Responsive layouts and themes
 - Dependency management
 - Code minification
- Services
 - GoogleMaps API
 - Backed by Pikalert data via WSGI (httpd with Python)
 - Radar from Web Mapping Service (WMS)
 - Via ncWMS2 hosted locally

Reminder: Pikalert High Level Diagram



Display Architecture



Real-Time Updates

Messaging Architecture

- Load Time
 - Static geographic data for road segments and RWIS sites (GeoJSON)
 - Pikalert output is applied to the geographic features
 - Contrast with use of OGC WFS
- When new data becomes available. . .
 - Display polls an “available times” service
 - Request to a specific service when new data is exposed
 - Site-specific forecast/observation requests
 - Camera images

Display Configuration

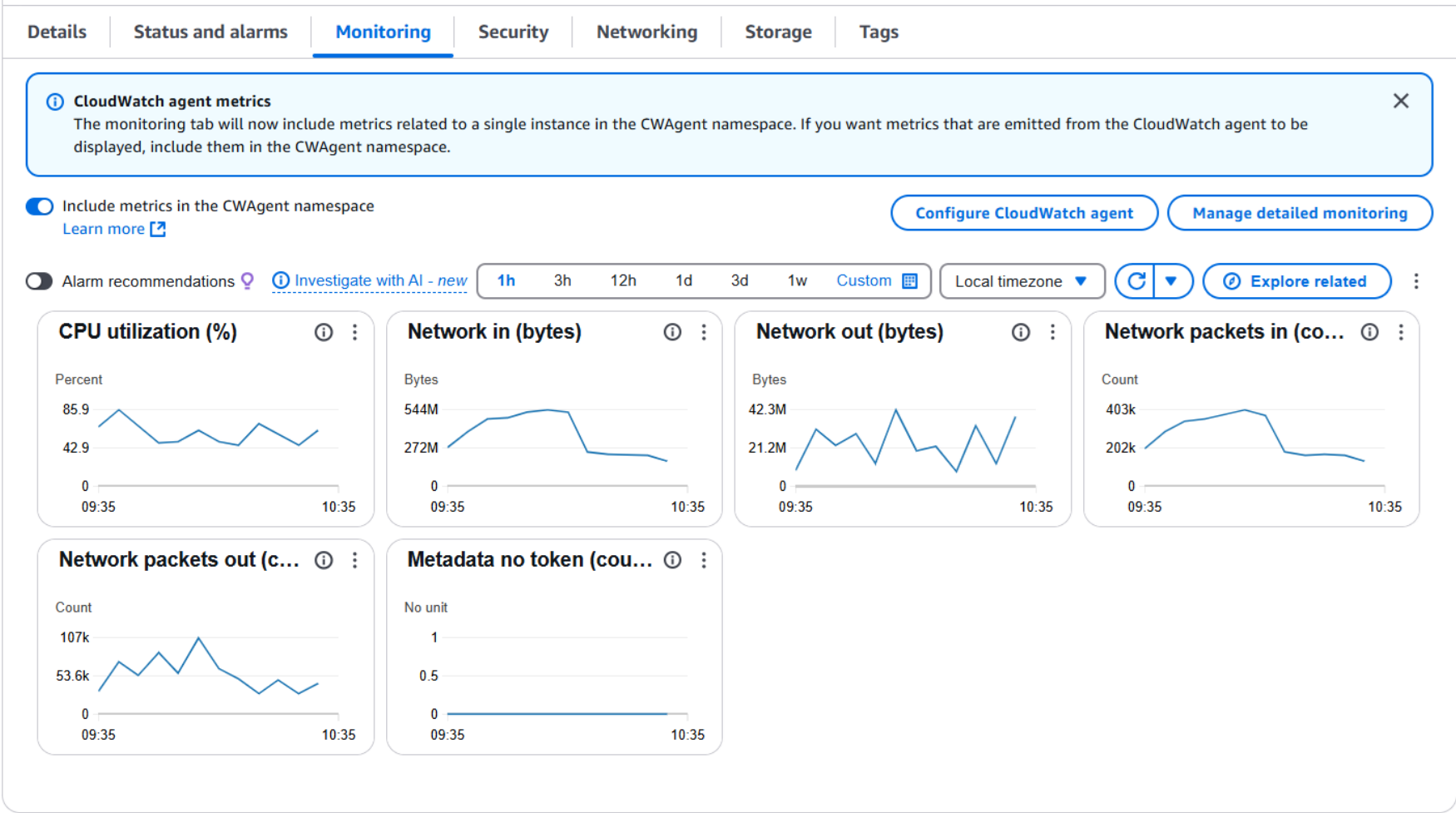
- System Configuration Files
 - Deployment-level settings (URLs, configured states)
 - Redeployment required (but ExtJS guarantees reloads)
- Display Configuration (per State)
 - GUI customizations
 - Visible tools
 - Visible layers
 - Options presented to users at that State
 - Default tab to realize in Site Detail view
 - No redeployment required, but caching of old configurations can be a problem

System Monitoring Software

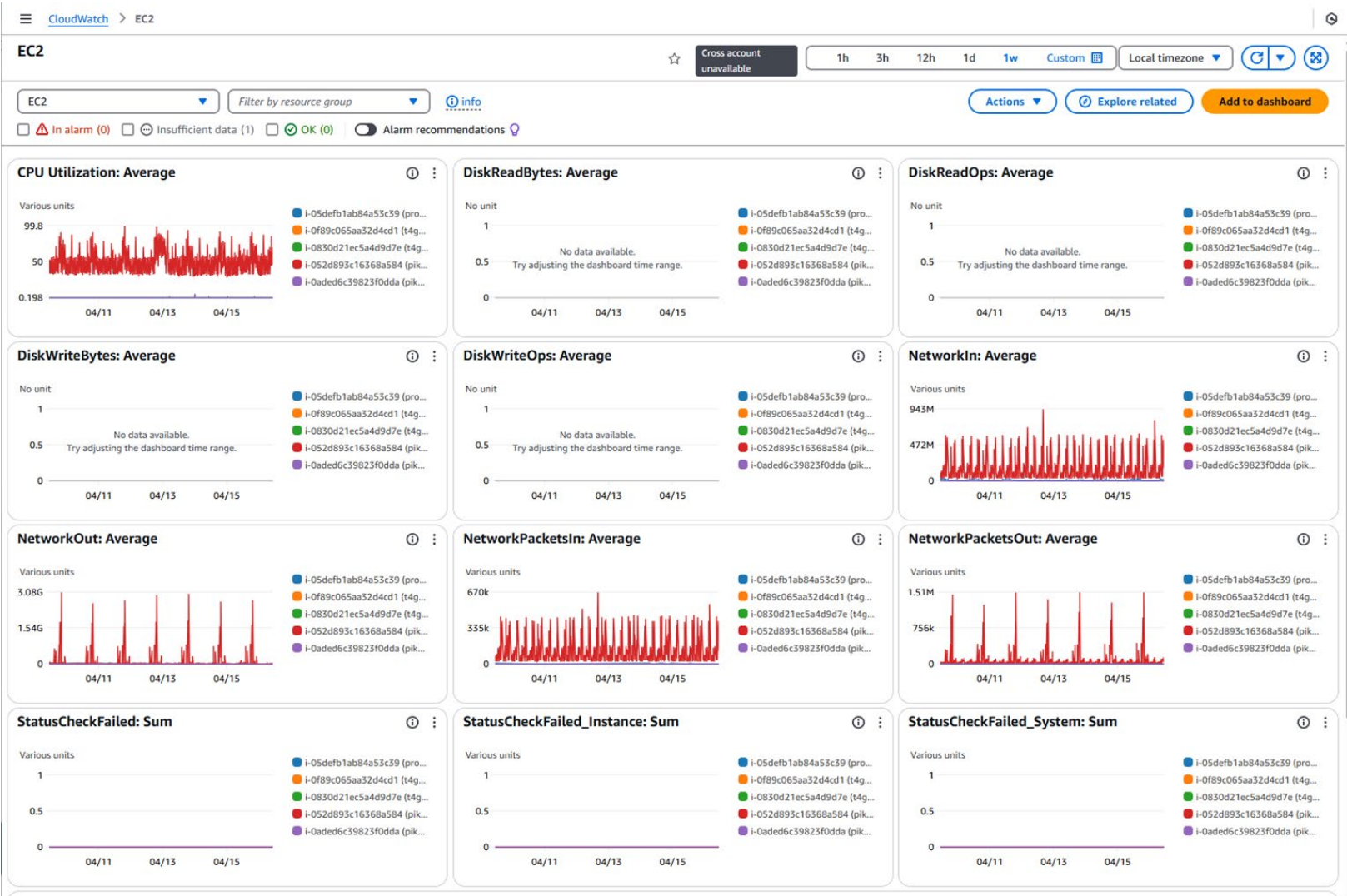
- Various tools are in place to monitor the system
 - File watcher
 - Does a file exist in a specific location at a specific time?
 - IE. new VDT output every 5 minutes
 - Emails are sent if an error is detected
 - AWS CloudWatch services
 - Is the EC2 online, performance/load in an acceptable range
 - Web based monitor
 - Allows for a quick view of the entire system

AWS

https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1#



AWS CloudWatch



Web Monitor

Pikalert2 CONUS VII System Monitor

Host: ip-172-31-28-17.ec2.internal, Tue Apr 15, 2025 21:54:16 up 575 days, 6:32, 7 users, load average: 5.15, 5.46, 5.99
(Next update: Tue, 15 Apr 2025 21:59:16 GMT)

		Tue Apr 15, 2025	Mon Apr 14, 2025	Sun Apr 13, 2025	Sat Apr 12, 2025	Fri Apr 11, 2025	Thu Apr 10, 2025	Wed Apr 09, 2025
Useful Links	System	Cron	Cron	Cron	Cron	Cron	Cron	Cron
		Disk Space	Disk Space	Disk Space	Disk Space	Disk Space	Disk Space	Disk Space
		System Load	System Load	System Load	System Load	System Load	System Load	System Load
Alaska Display	Input Data	Obs Data Files	Obs Data Files	Obs Data Files	Obs Data Files	Obs Data Files	Obs Data Files	Obs Data Files
Colorado Display	System Processes	Metar2nc	Metar2nc	Metar2nc	Metar2nc	Metar2nc	Metar2nc	Metar2nc
Iowa Display		RTMA	RTMA	RTMA	RTMA	RTMA	RTMA	RTMA
Oregon Display		Cleanup	Cleanup	Cleanup	Cleanup	Cleanup	Cleanup	Cleanup
Wyoming Display	Alaska Processes	AK Data Ingest	AK Data Ingest	AK Data Ingest	AK Data Ingest	AK Data Ingest	AK Data Ingest	AK Data Ingest
		AK RWIS to Madis	AK RWIS to Madis	AK RWIS to Madis	AK RWIS to Madis	AK RWIS to Madis	AK RWIS to Madis	AK RWIS to Madis
		AK Dual Pol Conversion	AK Dual Pol Conversion	AK Dual Pol Conversion	AK Dual Pol Conversion	AK Dual Pol Conversion	AK Dual Pol Conversion	AK Dual Pol Conversion
	Iowa Processes	AK Processes	AK Processes	AK Processes	AK Processes	AK Processes	AK Processes	AK Processes
		IA Data Ingest	IA Data Ingest	IA Data Ingest	IA Data Ingest	IA Data Ingest	IA Data Ingest	IA Data Ingest
		IA RWIS to Madis	IA RWIS to Madis	IA RWIS to Madis	IA RWIS to Madis	IA RWIS to Madis	IA RWIS to Madis	IA RWIS to Madis
	Wyoming Processes	IA Dual Pol Conversion	IA Dual Pol Conversion	IA Dual Pol Conversion	IA Dual Pol Conversion	IA Dual Pol Conversion	IA Dual Pol Conversion	IA Dual Pol Conversion
		IA Processes	IA Processes	IA Processes	IA Processes	IA Processes	IA Processes	IA Processes
		IA S3 Processes	IA S3 Processes	IA S3 Processes	IA S3 Processes	IA S3 Processes	IA S3 Processes	IA S3 Processes
	Oregon Processes	WY Data Ingest	WY Data Ingest	WY Data Ingest	WY Data Ingest	WY Data Ingest	WY Data Ingest	WY Data Ingest
		WY Image Processing/Rekognition	WY Image Processing/Rekognition	WY Image Processing/Rekognition	WY Image Processing/Rekognition	WY Image Processing/Rekognition	WY Image Processing/Rekognition	WY Image Processing/Rekognition
		WY RWIS to Madis	WY RWIS to Madis	WY RWIS to Madis	WY RWIS to Madis	WY RWIS to Madis	WY RWIS to Madis	WY RWIS to Madis
	Colorado Processes	WY Dual Pol Conversion	WY Dual Pol Conversion	WY Dual Pol Conversion	WY Dual Pol Conversion	WY Dual Pol Conversion	WY Dual Pol Conversion	WY Dual Pol Conversion
		WY Processes	WY Processes	WY Processes	WY Processes	WY Processes	WY Processes	WY Processes
		WY S3 Processes	WY S3 Processes	WY S3 Processes	WY S3 Processes	WY S3 Processes	WY S3 Processes	WY S3 Processes
	Colorado Processes	OR Data Ingest	OR Data Ingest	OR Data Ingest	OR Data Ingest	OR Data Ingest	OR Data Ingest	OR Data Ingest
		OR Image Ingest/Rekognition	OR Image Ingest/Rekognition	OR Image Ingest/Rekognition	OR Image Ingest/Rekognition	OR Image Ingest/Rekognition	OR Image Ingest/Rekognition	OR Image Ingest/Rekognition
		OR RWIS to Madis	OR RWIS to Madis	OR RWIS to Madis	OR RWIS to Madis	OR RWIS to Madis	OR RWIS to Madis	OR RWIS to Madis
	Colorado Processes	OR Dual Pol Conversion	OR Dual Pol Conversion	OR Dual Pol Conversion	OR Dual Pol Conversion	OR Dual Pol Conversion	OR Dual Pol Conversion	OR Dual Pol Conversion
		OR Processes	OR Processes	OR Processes	OR Processes	OR Processes	OR Processes	OR Processes
		OR S3 Processes	OR S3 Processes	OR S3 Processes	OR S3 Processes	OR S3 Processes	OR S3 Processes	OR S3 Processes
	Colorado Processes	CO Data Ingest	CO Data Ingest	CO Data Ingest	CO Data Ingest	CO Data Ingest	CO Data Ingest	CO Data Ingest
		CO Image Ingest	CO Image Ingest	CO Image Ingest	CO Image Ingest	CO Image Ingest	CO Image Ingest	CO Image Ingest
		CO RWIS to Madis	CO RWIS to Madis	CO RWIS to Madis	CO RWIS to Madis	CO RWIS to Madis	CO RWIS to Madis	CO RWIS to Madis
	Colorado Processes	CO Dual Pol Conversion	CO Dual Pol Conversion	CO Dual Pol Conversion	CO Dual Pol Conversion	CO Dual Pol Conversion	CO Dual Pol Conversion	CO Dual Pol Conversion
		CO Processes	CO Processes	CO Processes	CO Processes	CO Processes	CO Processes	CO Processes
		CO S3 Processes	CO S3 Processes	CO S3 Processes	CO S3 Processes	CO S3 Processes	CO S3 Processes	CO S3 Processes

Web Monitor

RWIS & Vehicle

2025-04-15

Process starts: 263

Process ends: 263

Minimum expected: 288

Warnings found: 1558

Errors found: 0

Process status: **WARNING**

Process log files:

[run_co_data_ingest.20250415.pyl](#)

Avalanche

2025-04-15

Process starts: 44

Process ends: 44

Minimum expected: 48

Warnings found: 0

Errors found: 0

Process status: **OK**

Process log files:

[get_co_avalanche_data.20250415.asc](#)

```
21:55:04 Starting: run_co_data_ingest.py
21:55:04 Info: downloading raw_data
21:55:17 Writing: /d1/vii/colorado/data/raw/rwis_raw/20250415/rwis_raw.20250415.2155.json
21:55:17 Writing: /d1/vii/colorado/data/raw/incidents/20250415/incidents.20250415.2155.json
21:55:17 Writing: /d1/vii/colorado/data/raw/vehicle/20250415/avl_raw.20250415.2155.json
21:55:20 Writing: /d1/vii/colorado/data/raw/road_conditions/20250415/road_conditions.20250415.2155.json
21:55:20 Writing: /d1/vii/colorado/data/raw/planned_events/20250415/planned_events.20250415.2155.json
21:55:21 Writing: /d1/vii/colorado/data/raw/destinations/20250415/destinations.20250415.2155.json
21:55:23 Writing: /d1/vii/colorado/data/raw/signs/20250415/signs.20250415.2155.json
21:55:23 Ret: 0
21:55:23 calling process_co_rwis_data.process_rwis()
21:55:23 Info: in process_rwis()
21:55:23 Info: after make_dir
21:55:23 Info: calling process_rwis_files: /d1/vii/colorado/data/raw/rwis_raw/20250415/rwis_raw.202504
21:55:23 Writing: /d1/vii/colorado/data/processed/rwis_processed/20250415/rwis_processed.20250415.2155
21:55:23 Calling RWIS_QC with config file: /d1/vii/colorado/data/static/config/CO_rwis_qc_config.json
21:55:23 Reading: /d1/vii/colorado/data/processed/rwis_processed/20250415/rwis_processed.20250415.2155
21:55:23 Reading: /d1/vii/colorado/data/static/config/CO_rwis_qc_config.json
21:55:23 Reading previous data up to 25 hours
21:55:27 Producing Neighbor Map
21:55:27 Reading: /d1/dicast/rctm/static_data/site_list/road_cond_pp_nbr.nc
21:55:27 Warning: Input site: 15718438506 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Warning: Input site: 18751288381 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Warning: Input site: 21474703680 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Warning: Input site: 23788319846 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Warning: Input site: 23805756192 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Warning: Input site: 7328496415 doesn't exist in the neighbor 'Dicast_site_list'
21:55:27 Reading previous data for neighbor check up to 1 hours
21:55:27 Performing QC checks
21:55:31 Info: For Variable 'windGust' QC'd out 1 value(s) (low: 0 high: 0 stuck: 1)
21:55:32 Info: For Variable 'relHumidity' QC'd out 1 value(s) (low: 0 high: 0 stuck: 1)
21:55:33 Info: For Variable 'roadState1' QC'd out 4 value(s) (low: 0 high: 0 stuck: 4)
21:55:34 Info: For Variable 'roadSurfaceFriction1' QC'd out 8 value(s) (low: 0 high: 8 stuck: 7)
21:55:35 Info: For Variable 'windDir' QC'd out 2 value(s) (low: 0 high: 0 stuck: 2)
21:55:36 Info: For Variable 'windSpeed' QC'd out 1 value(s) (low: 0 high: 0 stuck: 1)
21:55:37 Info: For Variable 'visibility' QC'd out 3 value(s) (low: 0 high: 0 stuck: 3)
21:55:40 Info: For Variable 'windGustDir' QC'd out 2 value(s) (low: 0 high: 0 stuck: 2)
21:55:40 Writing flagged qc data to: /d1/vii/colorado/data/processed/rwis_processed/20250415/rwis_proc
21:55:40 Writing output: /d1/vii/colorado/data/processed/rwis_processed/20250415/rwis_processed_qc.202
21:55:40 Info: concat_rwis
21:55:40 Info: wrote processed file in /d1/vii/colorado/data/processed/rwis_processed
21:55:40 Info: Ret: 0
21:55:40 Processing vehicle data
21:55:40 Writing: /d1/vii/colorado/data/processed/vehicle_generic/20250415/vehicle_generic.20250415.21
21:55:40 Info: calling generic_to_probe() on /d1/vii/colorado/data/processed/vehicle_generic/20250415/
21:55:40 Info: generic2probe_message(): writing output file /d1/vii/colorado/data/processed/probe_mess
21:55:40 Info: running run_probe_to_json
21:55:40 Info: file time: 20250415 2155
21:55:40 Info: Attempting: probe_to_json -l /d1/vii/colorado/data/log/probe_to_json_co 202504152155 /d
21:55:40 Info: probe_to_json -l /d1/vii/colorado/data/log/probe_to_json_co 202504152155 /d1/vii/colora
21:55:40 Info: Successfully processed: /d1/vii/colorado/data/processed/probe_message/20250415/probe_me
21:55:40 Info: Ret: 0
21:55:40 Ending: run_colorado_data_ingest.py, exit status = 0
```

Image Recognition

- Currently we leverage AWS Rekognition ‘Custom Labels’

Amazon Rekognition Custom Labels

Get started

Projects Updated

Pricing

Documentation

Custom Labels > Projects

Projects now manage datasets

Your previous datasets need to be associated with a project. We have associated the previously created datasets that trained the latest version of your models with their corresponding projects. Access your previously created datasets from the [Prior datasets](#) node. [Learn more](#)

Projects (41) Info

Delete

Download validation results

Train new model

Create project

Search projects by project name

< 1 2 3 4 ... >

	Name	Versions	Date created	Model performance	Model status	Status message
<input type="radio"/>	<div>wydot_expanded_set_20250408</div>	1	2025-04-15			
<input type="radio"/>	<div>wydot_expanded_set_20250408.2025-04-15T16.24.38</div>		2025-04-15	0.976	TRAINING_COMPLETED	The model is ready to run.
<input type="radio"/>	<div>wydot_expanded_set_20250319</div>	1	2025-03-20			
<input type="radio"/>	<div>wydot_expanded_set_20250319.2025-03-19T19.47.38</div>		2025-03-20	0.949	TRAINING_COMPLETED	The model is ready to run.
<input type="radio"/>	<div>oregon_stratified_L5_masking</div>	1	2025-01-28			
<input type="radio"/>	<div>oregon_stratified_L5_masking.2025-01-28T13.28.10</div>		2025-01-28	0.966	TRAINING_COMPLETED	The model is ready to run.
<input type="radio"/>	<div>oregon_stratified_uncropped</div>	1	2025-01-09			
<input type="radio"/>	<div>oregon_stratified_uncropped.2025-01-09T15.13.11</div>		2025-01-09	0.968	STOPPED	The model is ready for hosting.
<input type="radio"/>	<div>oregon_stratified_cropped</div>	1	2025-01-09			
<input type="radio"/>	<div>oregon_stratified_cropped.2025-01-09T15.09.39</div>		2025-01-09	0.963	STOPPED	The model is ready for hosting.
<input type="radio"/>	<div>strat_model_a-20230918_expanded_train_set_L5_masking</div>	1	2025-01-08			
<input type="radio"/>	<div>strat_model_a-20230918_expanded_train_set_L5_masking.2025-01-08T16.16.46</div>		2025-01-08	0.950	TRAINING_COMPLETED	The model is ready to run.
<input type="radio"/>	<div>strat_model_a-20230918_expanded_train_set_SAM_masking</div>	1	2025-01-08			
<input type="radio"/>	<div>strat_model_a-20230918_expanded_train_set_SAM_masking.2025-01-08T16.13.33</div>		2025-01-08	0.948	TRAINING_COMPLETED	The model is ready to run.
<input type="radio"/>	<div>roman_test_2</div>	1	2025-01-07			

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Image Recognition

Custom Labels > Projects > wydot_model_a_20230918 > Models > wydot_model_a_20230918.2023-09-18T15.37.52

wydot_model_a_20230918.2023-09-18T15.37.52Info

Delete model

Evaluation

Model details

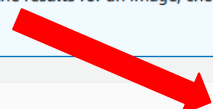
Use model

Tags



Evaluation

The Evaluation tab shows the testing results for your trained model. This helps you understand the overall performance of your model. To view the results for an image, choose the View test results button.



Evaluation results

[View test results](#)

F1 scoreInfo

0.962

Date completed

September 18, 2023

Trained in 3,804 hours

Average precisionInfo

0.961

Training dataset

5 labels, 3,210 images

Overall recallInfo

0.963

Testing dataset

5 labels, 806 images

Per label performance (5)


 Find labels

< 1 >

Label name	F1 score	Test images	Precision	Recall	Assumed threshold
Dirty_obsured	0.985	67	0.985	0.985	0.449
dry	0.993	277	0.986	1.000	0.218
slick	0.942	169	0.969	0.917	0.716
slick_in_spots	0.917	141	0.893	0.943	0.329
wet	0.970	152	0.974	0.967	0.604

Image Recognition

[Custom Labels](#) > [Projects](#) > [wydot_model_a_20230918](#) > [Models](#) > [wydot_model_a_20230918.2023-09-18T15.37.52](#) > Performance

 Evaluate image

Review the test results of your trained model for individual images. Below each image is information about the model's predicted label compared with the label assigned to the image in the test dataset, noted by result type. You can also filter by label and result types.

Filter by label


Choose labels
Choose labels to filter images

☒ True positive
☐ False positive
☐ False negative

Images (778)[Info](#)

< 1 ... 9 10 11 12 13 14 ... >

Vid-000178000-00-01-2023-05-23-14-58.jpg



Labels


Confidence

dry

True positive

99.9%

Vid-000178000-00-01-2023-05-23-16-07.jpg



Labels


Confidence

dry


True positive



100.0%

Vid-000178000-00-01-2023-05-23-18-59.jpg



Vid-000178000-00-01-2023-05-23-20-42.jpg





NCAR
National Center for
Atmospheric Research

Image Recognition

Filter by label

Choose labels
Choose labels to filter images

Q Select a label


☐ True positive
☒ False positive
☐ False negative

Images (30) [info](#)

Q Search images by file name


< 1 2 3 4 ... >

Vid-020100017-00-01-2022-12-13-14-24.jpg




Labels	Confidence
<div>slick_in_spots</div> <div>False positive</div>	58.3%
<div>Dirty_obsured</div> <div>False negative</div>	2.1%

Vid-000178000-00-01-2022-12-10-21-11.jpg




Labels	Confidence
<div>dry</div> <div>True positive</div>	47.7%
<div>Dirty_obsured</div> <div>False positive</div>	52.1%

Vid-020100016-00-01-2023-05-24-16-05.jpg



Labels	Confidence
<div>dry</div> <div>True positive</div>	21.8%
<div>wet</div> <div>False positive</div>	77.2%

Vid-000178000-00-01-2023-03-05-20-21.jpg



Labels	Confidence
<div>slick_in_spots</div> <div>False positive</div>	83.4%
<div>slick</div> <div>False negative</div>	16.6%

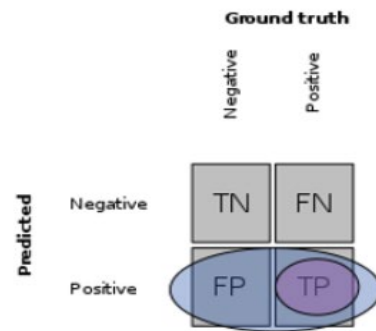
Image Recognition Real -Time Evaluation (2023-2024 Winter Season)

- Precision: percentage of model predicted true values that are correctly classified

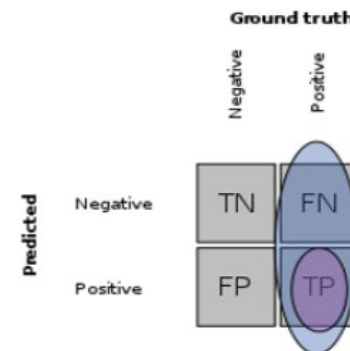
$$\frac{TP}{TP+FP}$$

- Recall (sensitivity): percentage of actual true values that are correctly classified

$$\frac{TP}{TP+FN}$$



Precision



Recall

- F(1)-Score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Image Recognition Real -Time Evaluation (2023-2024 Winter Season)

- Primary Conclusions
 - Model correctly labeled 99.7% of the nondry images
 - Model called over 1,770 dry images ‘wet’ (27% of all dry images)
 - Lots of ‘False Alarms’, most often incorrectly labeled as ‘wet’
 - Model struggles to identify wet roads
 - Largely due to different pavement shading (freshly paved roads are generally darker and can cause the model to label them wet)
 - Model struggled between different levels of snow coverage
 - Hard to identify if there is more or less than 50% coverage
 - Model does much better when just tasked with determining if road is bare or impacted.

True category			
Labeled category	bare	impacted	other
bare	1885	17	25
impacted	64	2757	104
other	141	21	1103

Recent Project Successes & Enhancements

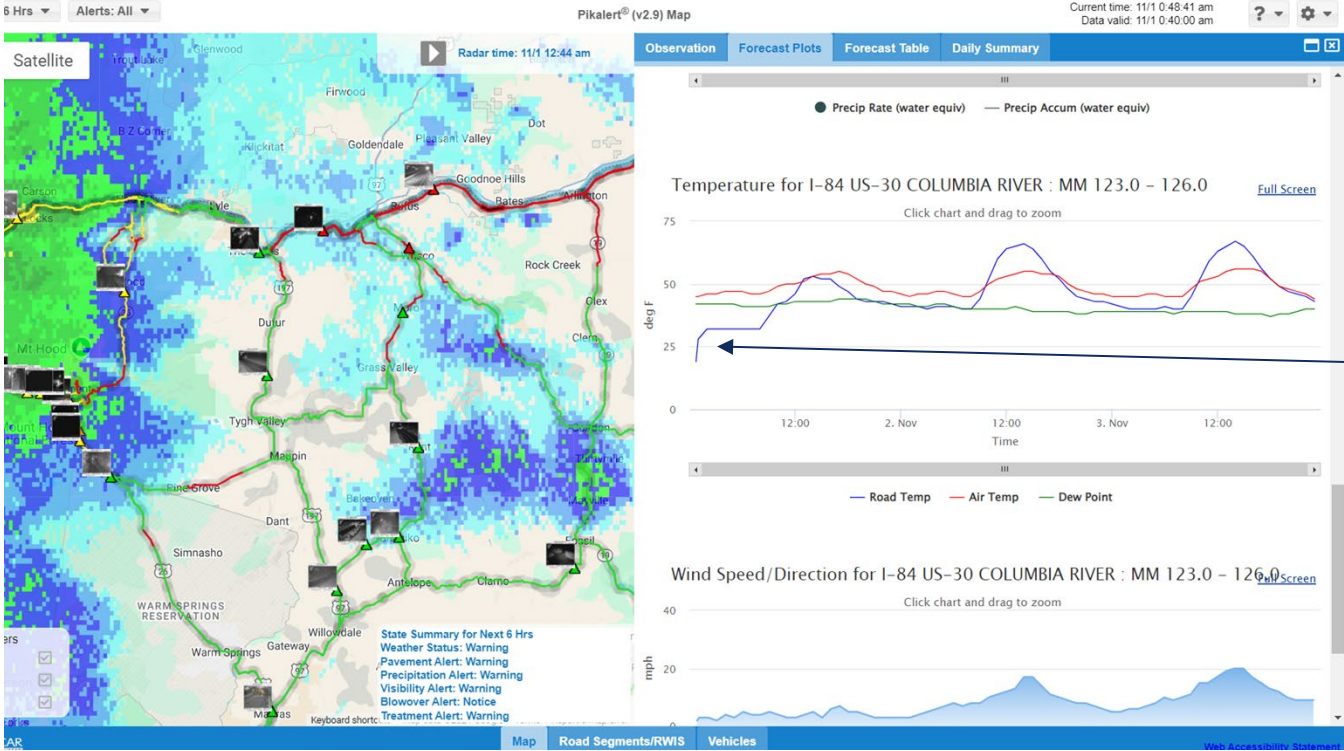
- Updates to the Road Temperature forecasting module (METRo)
 - Older versions of METRo has known warm biases in the afternoon, especially during the warmer shoulder seasons.
 - Upgraded METRo from version 3.2.6 to 4.0.1
 - New version uses forecasted solar radiation from DICAST
 - Prior version used only cloud cover
 - GOES 18/19 Satellite data is downloaded to create pseudo solar radiation observations that are used to tune the DICAST radiation forecasts.
 - Led to large reduction of the warm bias, as well as overall reduction in mean absolute error (MAE)
 - Accurate road temperature forecasts are critical to identifying the road state.

Recent Project Successes & Enhancements

- Implementation of a new RWIS QC module
 - Previously, all QC took place within the DICAST system, which primarily relied on a min/max bound check, a step check and a climatology check.
 - Climatology checks aren't performed on RWIS data.
 - A new QC module was created for the RWIS data and applied to all states. The new QC module includes stuck value checks as well as neighbor checks. The module is able to find and remove more questionable data so that it doesn't impact the system.
 - Stuck value check has the biggest impact in terms of flagging bad data
 - QC data is tracked and summarized on a weekly basis, and reports are sent to the sponsors so that they can perform maintenance on faulty devices

Recent Project Successes & Enhancements

- RWIS QC Module + Weekly QC Reports
 - The system relies heavily on accurate observations => bad observations can cause big problems



Nearby RWIS was reporting unrealistically low road T values, which lowered the road T of nearby segments, turning water into ice and triggering hazards.

Example QC Report

```

Site: FR3027      Site Name: Five trees
  roadTemperature1_QCFLAG: 922      failed_data_min,max,mean: 244.71, 247.08, 246.20
                                      NEIGHBOR_QC: 0, RANGE_QC: 922, STUCK_QC: 0

Site: FR3070      Site Name: Reynolds at Bancroft
  temperature_QCFLAG: 779      failed_data_min,max,mean: -99.99, -99.99, -99.99
                                NEIGHBOR_QC: 0, RANGE_QC: 779, STUCK_QC: 744
  dewpoint_QCFLAG: 779      failed_data_min,max,mean: -139.99, -139.99, -139.99
                                NEIGHBOR_QC: 0, RANGE_QC: 779, STUCK_QC: 744
  roadTemperature1_QCFLAG: 779      failed_data_min,max,mean: 1037.37, 1037.37, 1037.37
                                NEIGHBOR_QC: 0, RANGE_QC: 779, STUCK_QC: 744

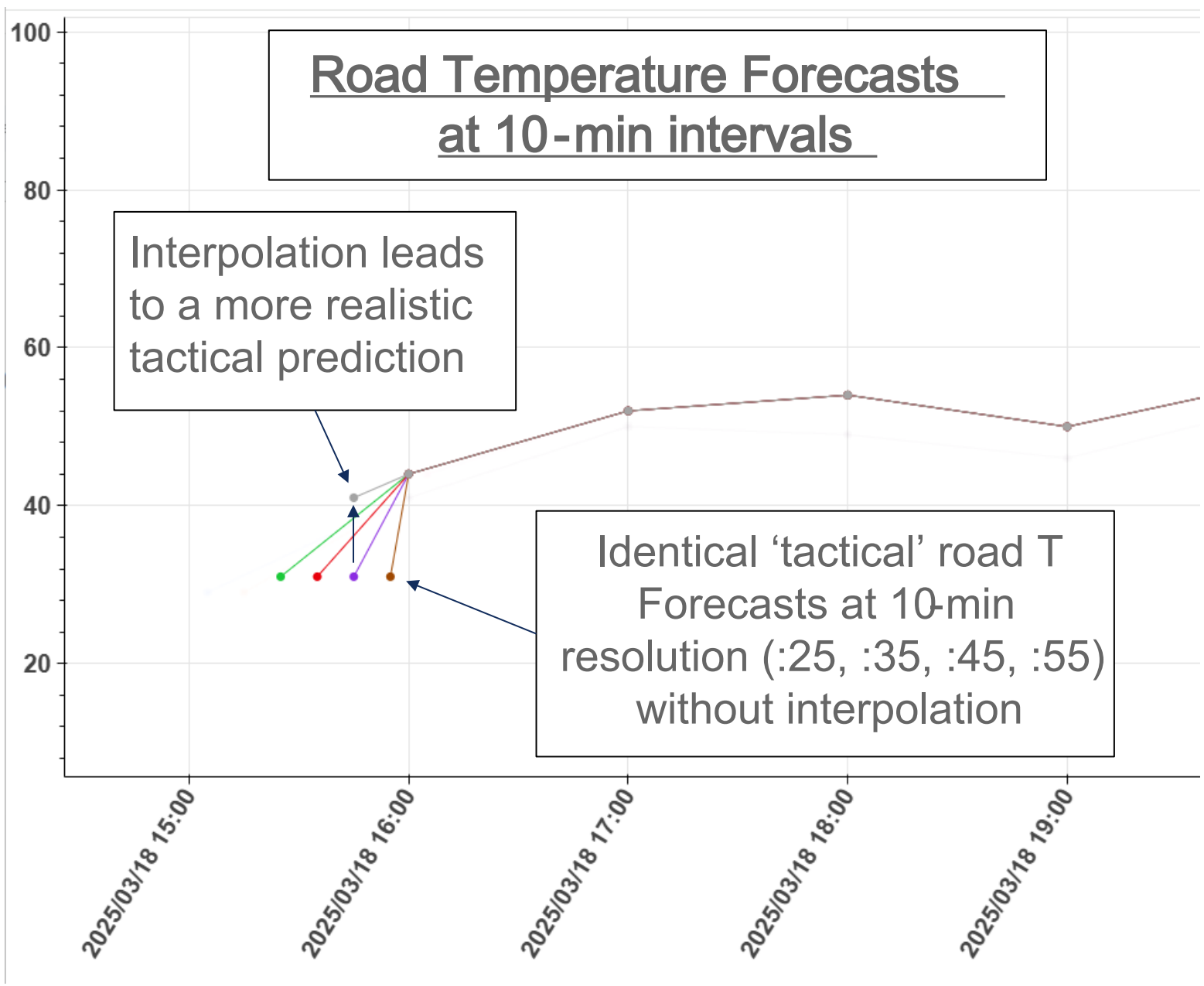
Site: FR3215      Site Name: South Tewksbury
  temperature_QCFLAG: 837      failed_data_min,max,mean: -99.99, -99.99, -99.99
                                NEIGHBOR_QC: 0, RANGE_QC: 837, STUCK_QC: 801
  dewpoint_QCFLAG: 837      failed_data_min,max,mean: -139.99, -139.99, -139.99
                                NEIGHBOR_QC: 0, RANGE_QC: 837, STUCK_QC: 801

Site: FR3909      Site Name: Cambridge
  temperature_QCFLAG: 1006      failed_data_min,max,mean: -99.99, -99.99, -99.99
                                NEIGHBOR_QC: 0, RANGE_QC: 1006, STUCK_QC: 1006
  dewpoint_QCFLAG: 1006      failed_data_min,max,mean: -139.99, -139.99, -139.99
                                NEIGHBOR_QC: 0, RANGE_QC: 1006, STUCK_QC: 1006
  roadTemperature1_QCFLAG: 1006      failed_data_min,max,mean: 1037.37, 1037.37, 1037.37
                                NEIGHBOR_QC: 0, RANGE_QC: 1006, STUCK_QC: 1006

```

Recent Project Successes & Enhancements

- Various application / configuration updates
 - Improvements to snow accumulation predictions
 - Improvements to D1Cast to produce more accurate precip types, leading to better snowfall accumulation predictions
 - Updates to the RWH module to improve thresholds which determine precip intensity (light, moderate, heavy)
 - Improvements to spatial and temporal inconsistencies
 - Due to spatial and temporal averaging, inconsistencies can occur between segments that use different neighbors. Configuration updates were implemented to improve neighbors and reduce inconsistencies
 - Linear interpolation was implemented for the tactical weather forecasts



Pikalert in Rural Areas

- Pikalert skill correlates to data / observation availability. In a rural area, surface stations and camera images are going to be sparse, so the system will have to rely on vehicle data (if available) and/or satellite data (satellite data is available over all of CONUS).
 - If no station obs aren't available, system will rely on RTMA
- If no observations are available, forecasts will depend strictly on model data processed through the DICAST system, with no ability to tune to station observations.

Project Success Criteria

- No real 'success criteria' for the system.
- Statistics are calculated in real time for atmospheric weather variables
 - (IE. Temperature, wind speed, dewpoint, etc. . .)
- What matters for most clients is storm prediction and snowfall forecasting (IE. How much impact will this storm have on the traveling public).
 - This is difficult to quantify, as snowfall observations are often difficult to obtain and can be unreliable.
 - Difficult to quantify impact of a weather event on the public.
 - Rush hour, construction, accidents can complicate this data.
- Our team relies mostly on manual checks and feedback from our sponsors in terms of whether or not the system is providing accurate guidance.

Recent Project Success & Enhancements

- Feedback from sponsors that our alerts are more reliable and accurate than other tools.
 - Alaska DOT is increasing its reliance on Pikalert.
 - Winter storm event in February 2024 where a sponsor was concerned about a low snowfall forecast coming out of Pikalert for some areas north of Chicago (along the western edge of Lake Michigan). Pikalert was showing 2-3" while other weather outlets were showing 10" or more. The region ended up getting only 23" of snow. Our sponsor received highly positive feedback from their clientele and this case further strengthened their confidence in our system.
- Recently signed contract with WYDOT to expand image recognition to all sites and improve performance.
- CDOT chose Pikalert as their preferred road weather decision support platform.

Project Challenges and Opportunities

- Spatial and temporal inconsistencies still exist, and will likely always exist in a data driven system, particularly around the boundaries between differing observing stations.
- Data Quality Control (QC) is never going to be able to flag everything, improvements in the QC area are being researched.
- Lack of useful vehicle information (how can we get more?)
- Image Recognition:
 - Computer vision performance can suffer on unseen stations
 - if a new station/site is added, images from that station must be manually labeled and added to the training dataset (\$\$\$)
 - AWS Rekognition models are essentially a 'black box'. We have no ability to get into the nuts and bolts of the model to determine why/how it's making the classifications.
 - We are pursuing our own inhouse CNN (Convolutional Neural Network) to replace Rekognition.

Moving Forward

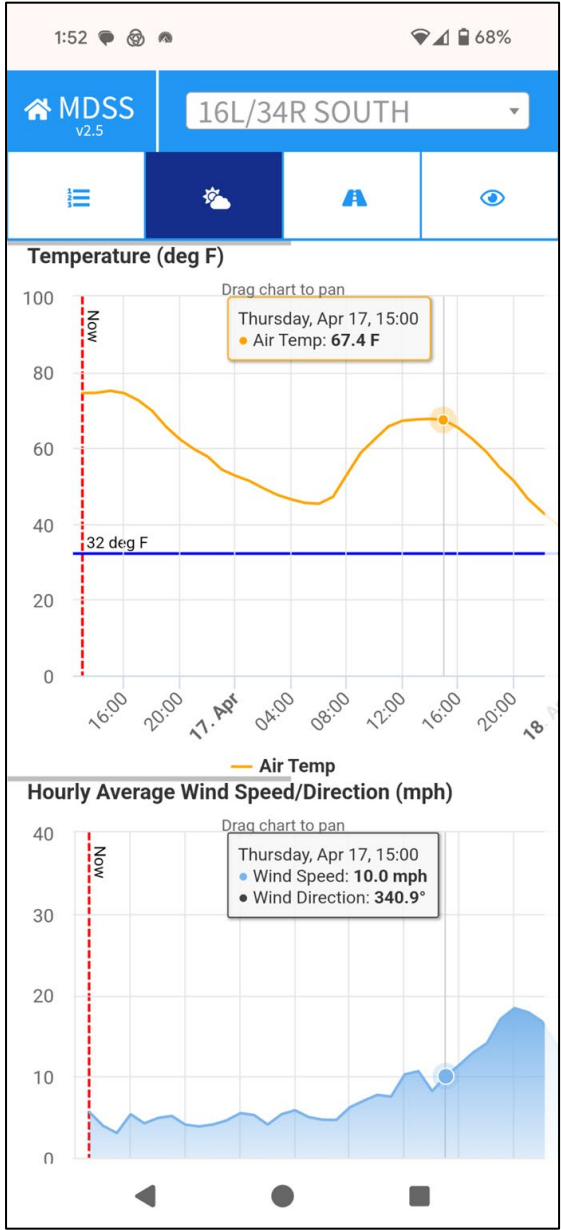
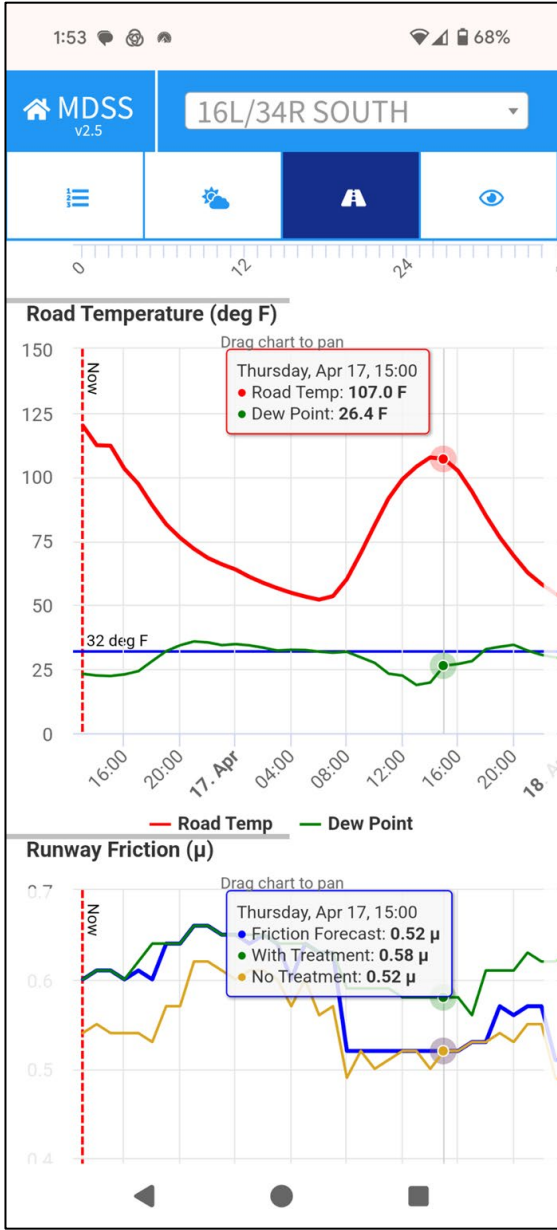
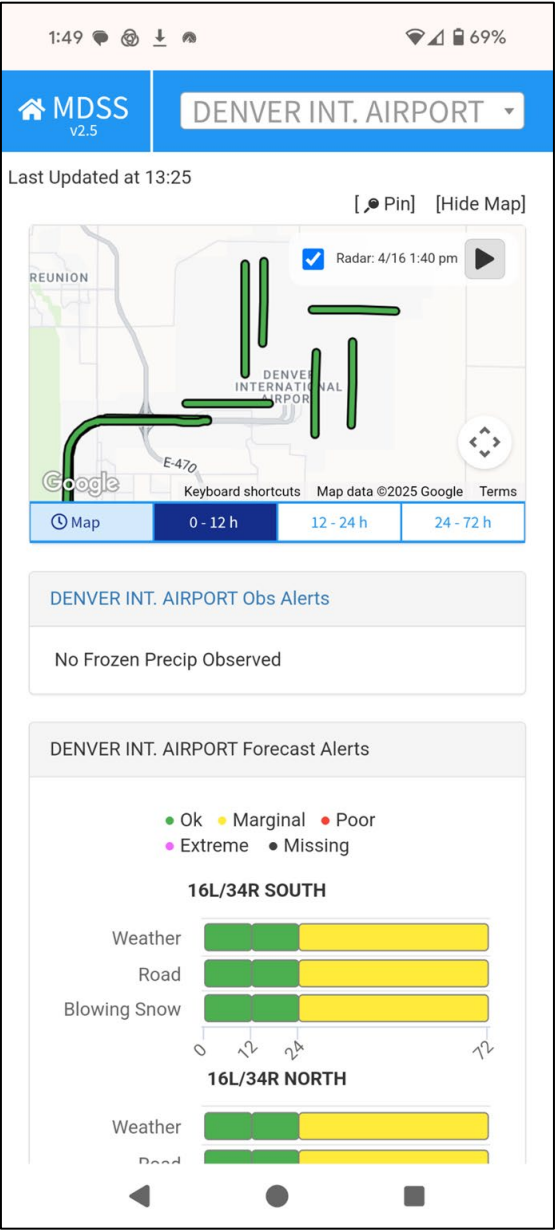
- Constant case studies to drill down into specific system successes and failures
 - Case studies often uncover system faults or broken assumptions
- Image Recognition Updates
 - Develop in-house Convolutional Neural Network (CNN)
 - Improve model performance
 - Work to generalize the model so that it can be expanded to new stations without requiring manual image labeling and retraining
 - Identify new weather events, specifically blowing snow and fog.
- Incorporate a ML-based roadway friction prediction into Pikalert

Related Research Areas (Road Surface Friction)

- NCAR has engaged in multiple research studies related to roadway friction modeling. The most recent won an AASHTO (American Association of State Highway and Transportation Officials) award for high value research.
 - <https://www.intrans.iastate.edu/news/aurora-project-on-friction-modeling-earns-aashto-award/>
 - <https://www.aurora-program.org/research/completed/roadway-friction-modeling-improving-the-use-of-friction-measurements-in-state-dots/>
- The work revolved around analyzing roadway friction measurements under various conditions and recorded by various non-invasive friction sensors.
- AI/ML models were evaluated to determine their predictive capabilities across varying domains.

Related Research Areas (Runway Surface Friction)

- Runway Friction models were also developed and deployed for the Minneapolis StPaul International Airport (MSP) and for the Denver International Airport (DEN).
 - MSP: 2017 - 2020
 - DEN: 2016 - 2029
- Going forward, we'd like to leverage our experience in the roadway friction domain to build this into our Pikalert system.



Related Research Areas (Variable Speed Limits)

- Variable Speed Limits (VSL) are used to promote safety by lowering speeds on certain roadways under dangerous conditions, such as winter storm events and limited visibility.
 - VSLs are often reactive to storm events (IE. they don't recommend slower speeds until the storm has hit or until observed traffic speed has slowed down).
- NCAR is nearing the final stages of a research project aimed at predicting VSL slowdowns using available weather data.
 - Our research focuses on utilizing RWIS weather and road condition observations to inform VSL speeds
 - Our goal is to develop AI/ML models that can reliably and rapidly provide safe recommended speeds as weather and road conditions begin to change on the ground

Conclusion

- Thanks to the Western States Forum for hosting.
- Special thanks to our Sponsors
 - Colorado DOT
 - Iowa DOT
 - Alaska DOT
 - Wyoming DOT
 - Oregon DOT
 - Frost Solutions
- NCAR Pikalert team--
 - Seth Linden, Amanda Anderson, Bill Petzke, Paddy McCarthy, Roman Lynch, and Curtis Walker
 - No longer with us (retired)
 - Gerry Wiener and Jim Cowie.
- Questions?
 - brummet@ucar.edu