Scaling Model Predictive Control with AWS: A Real-Time Simulation Platform

Ramin Esmzad

Mechanical Engineering, Michigan State University

April 14, 2025

Project Aim

- Build and deploy a real-time Model Predictive Control (MPC) simulation platform in the cloud.
- Enable users to define custom system models and run simulations online.
- Leverage AWS cloud resources for scalable, asynchronous, and secure backend processing.
- Help researchers conduct better and faster experimentation through scalable cloud infrastructure.

Demo: Real-Time MPC Simulation (Screenshot)

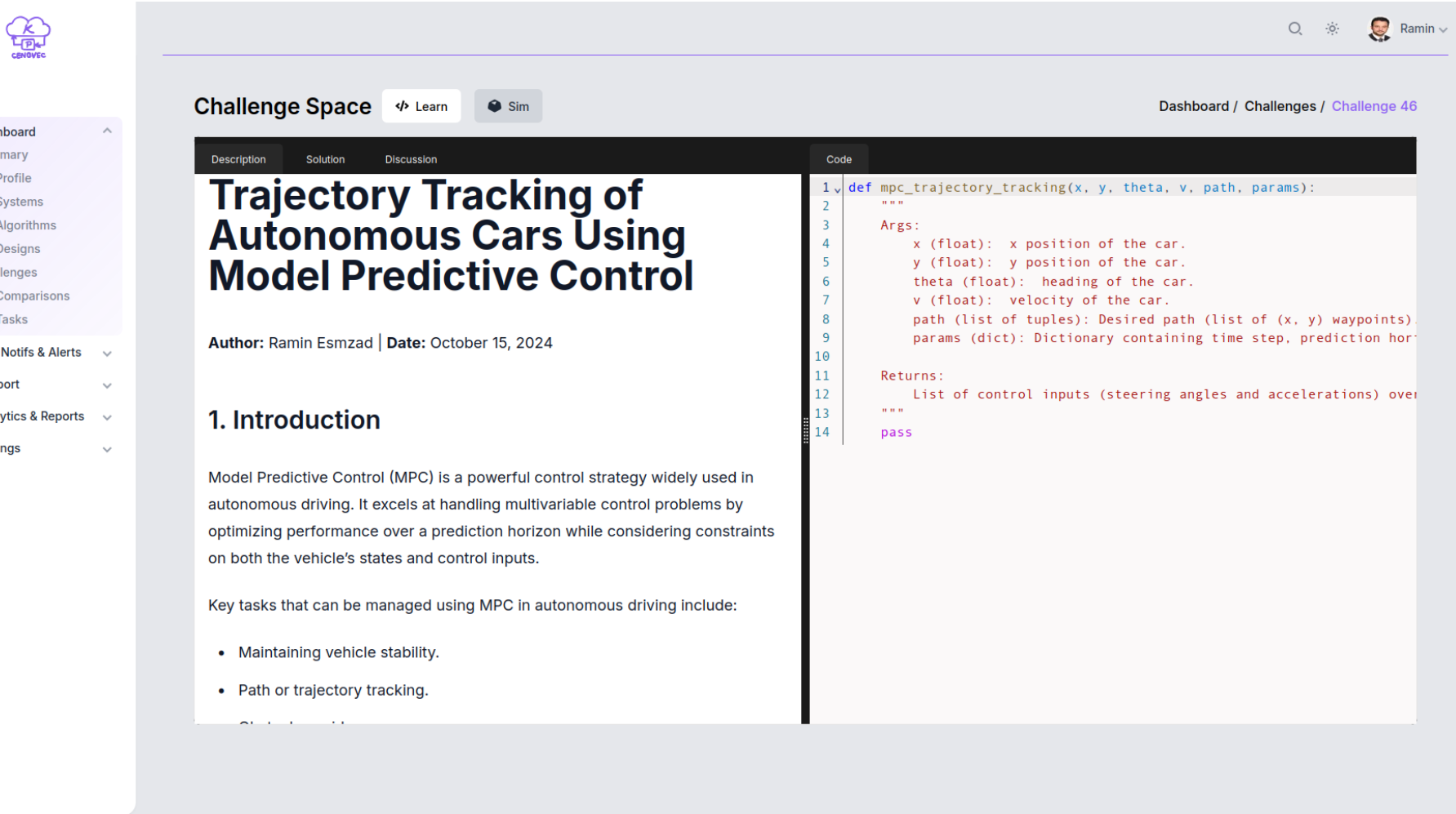


Figure: User running a challenge simulation from a web browser

Demo: Real-Time MPC Simulation (Screenshot)

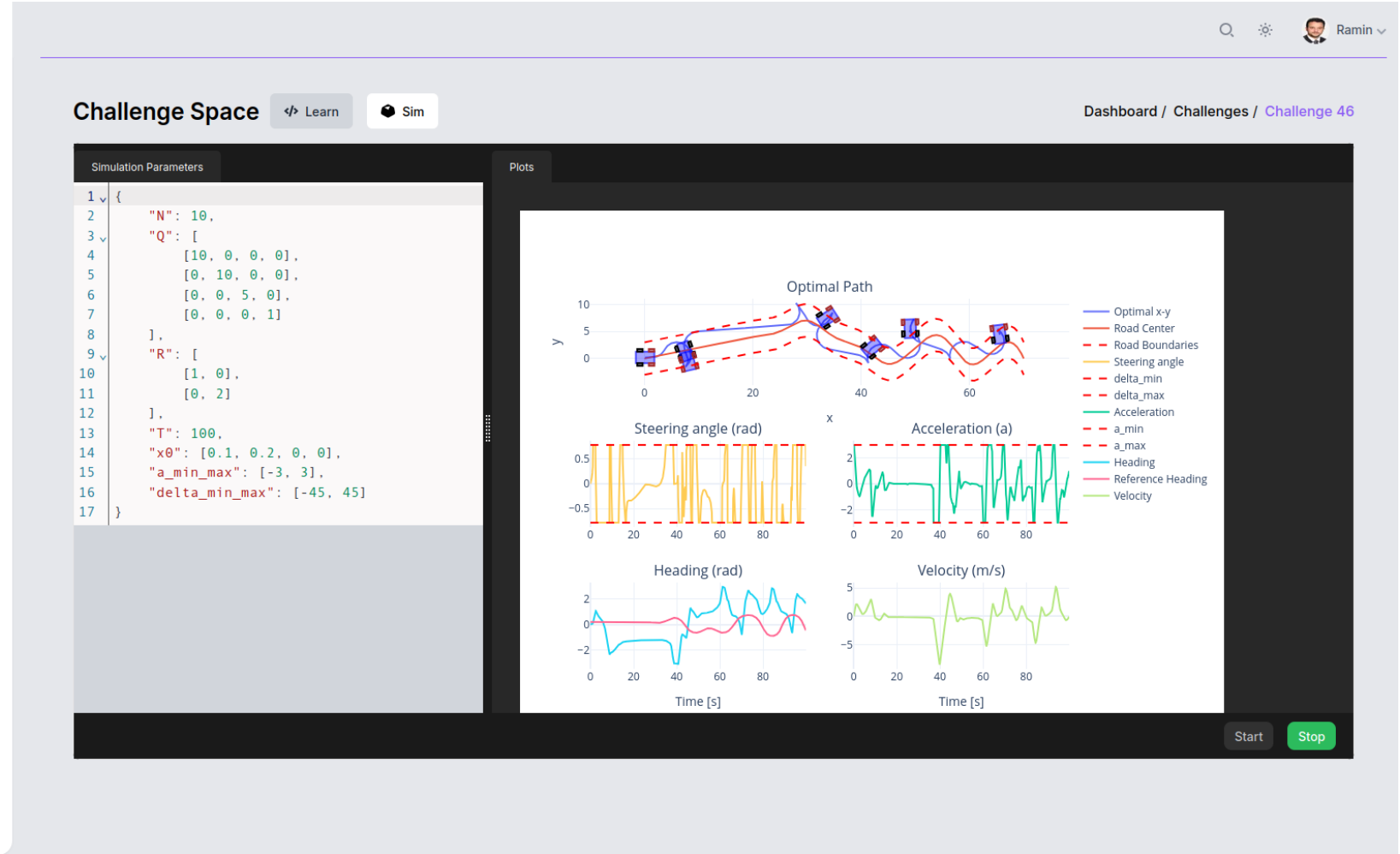


Figure: User running a challenge simulation from a web browser

MPC Problem Formulation and its Challenges

Model Predictive Control (MPC):

- At each timestep, solve an optimization problem to compute a control action:

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)$$

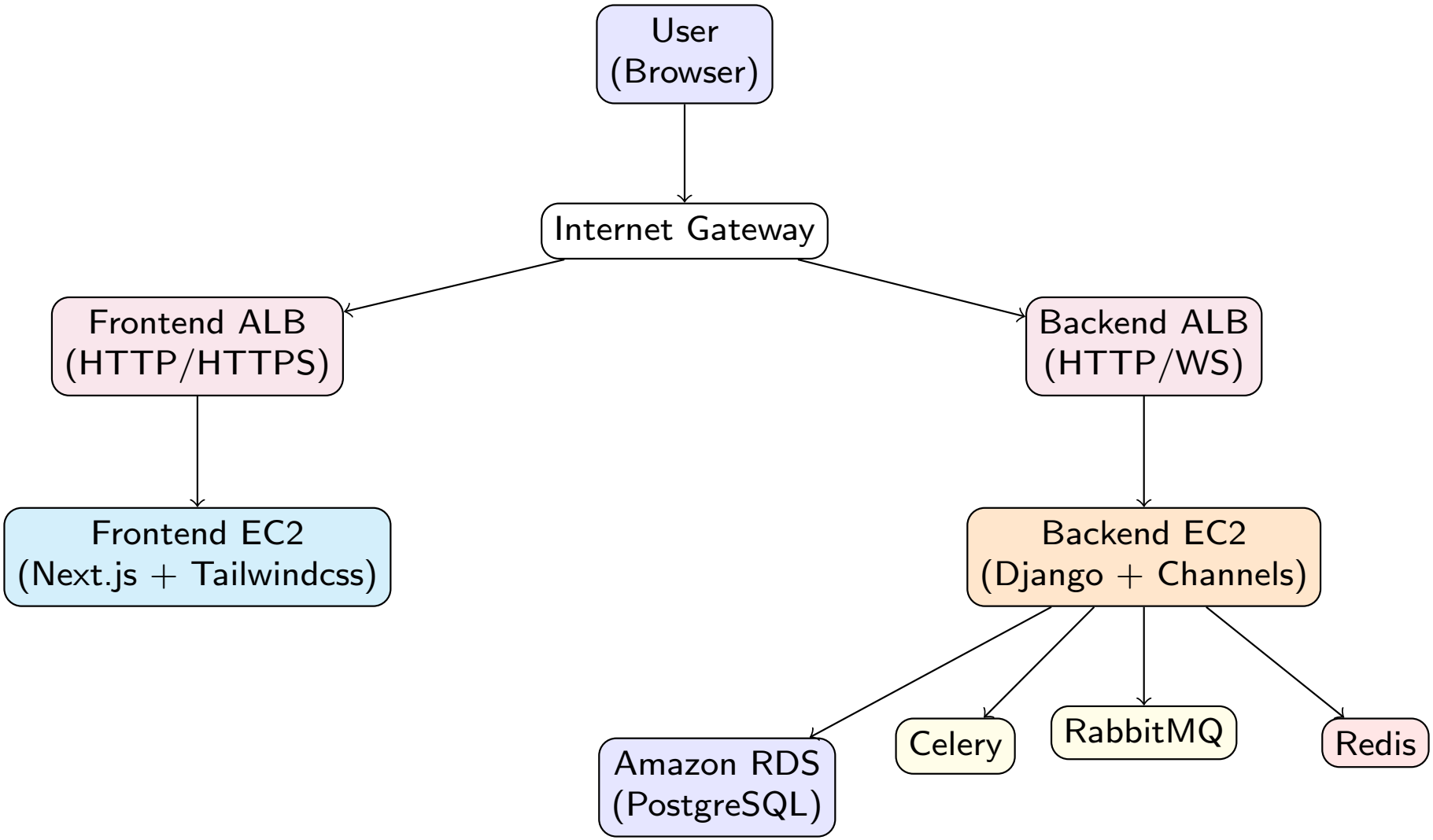
$$\text{s.t} \quad x_{k+1} = f(x_k, u_k), \quad x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}$$

- Only the first input u_0 is applied before re-solving.

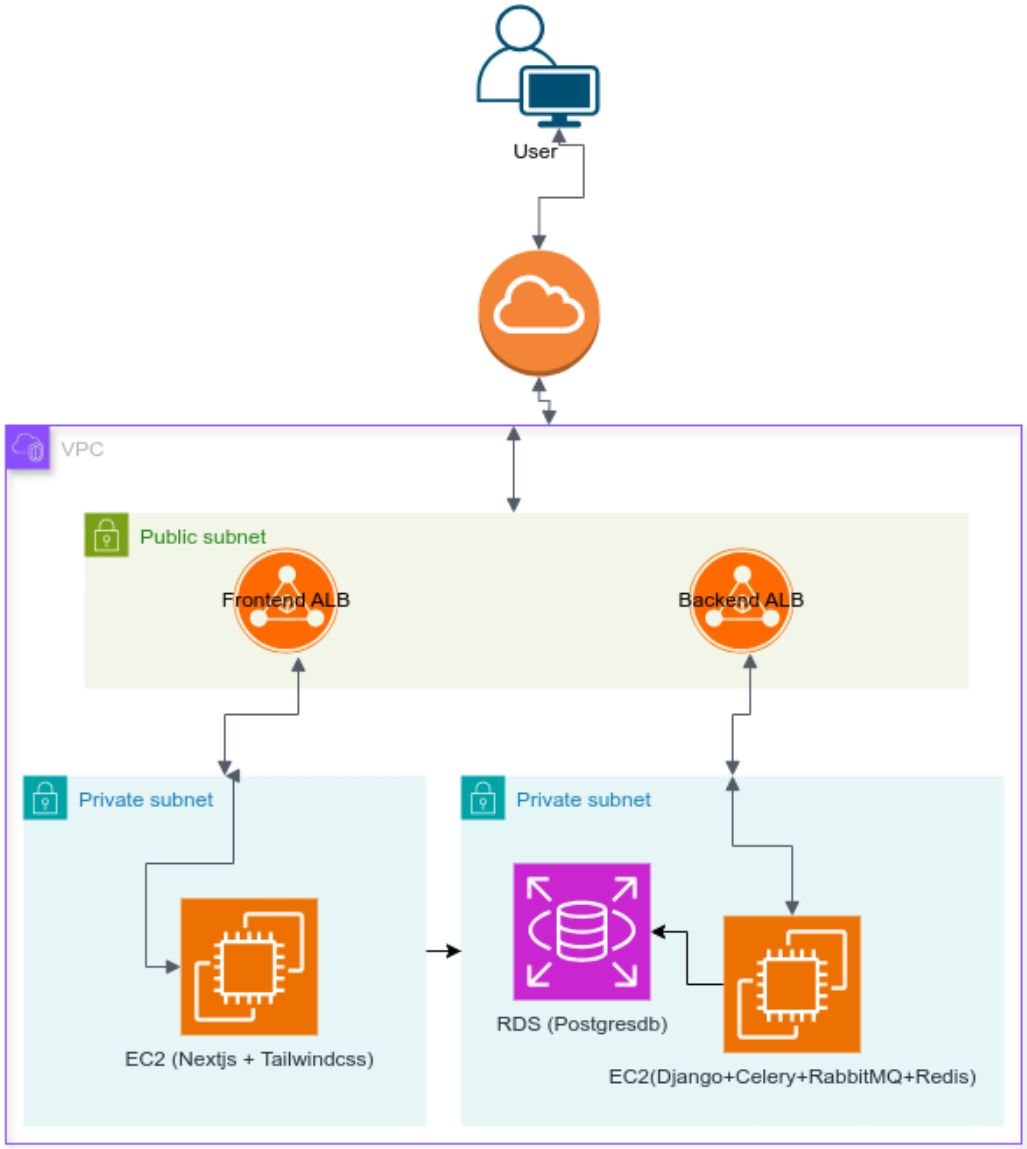
Challenges in Research and Practice:

- **Real-time Solvability:** Must solve optimization quickly (especially for fast systems).
- **Scalability:** High-dimensional or nonlinear systems increase computational burden.
- **Robustness:** MPC must handle uncertainty, noise, and model mismatch.
- **Deployment:** Running MPC simulations on the cloud requires managing computation, latency, and fault tolerance.

Request Flow in the Cloud Architecture



AWS Diagram


User
VPC
Public subnet
Frontend ALB
Backend ALB
Private subnet
Private subnet
EC2 (Nextjs + Tailwindcss)
RDS (Postgresdb)
EC2(Django+Celery+RabbitMQ+Redis)

System Architecture Overview

- **Frontend:** Next.js + TailwindCSS for interactive UI
- **Backend:** Django + Django Channels (WebSocket)
- **Asynchronous Tasks:** Celery with RabbitMQ and Redis
- **Simulation Core:** Python, CVXPy, NumPy, SciPy, PyTorch (for future RL support)
- All services deployed in a logically separated VPC with strict public/private subnet boundaries
- Backend stack supports modular addition of new solvers (e.g., Pyomo, CasADi)
- WebSocket and REST API layers abstract simulation scheduling and visualization

AWS Cloud Services Used

- **Amazon EC2** – Run Django backend, Celery workers, and simulation engine. Also used to host the frontend Next.js app.
- **Amazon RDS (PostgreSQL)** – Relational DB for user data and problem storage
- **Elastic Load Balancer (ALB)** – Route frontend and backend requests securely to respective EC2 instances
- **AWS VPC, Subnets, NAT** – Isolated private/public subnet setup
- **AWS IAM** – Secure user and role-based permissions
- **AWS CloudWatch** – Application-level monitoring, logs, and metrics

Resource Map

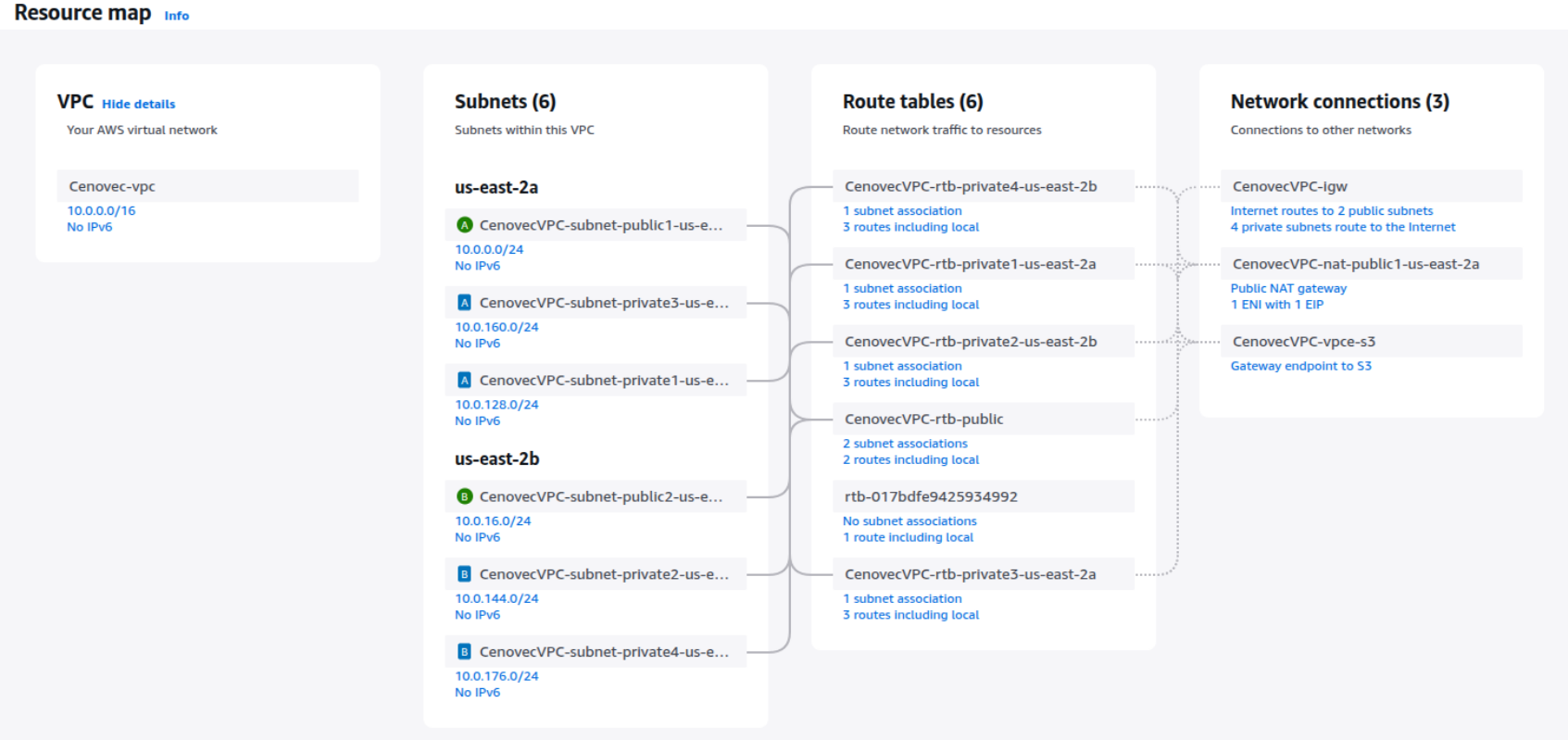


Figure:Resource Map of the Current Platform Setup

Successes

- Successfully deployed MVP version with multiple integrated AWS services
- Real-time simulation flow works reliably through Celery queues and Django Channels
- Frontend/backend communicate securely through ALB and private VPC routing
- Environment variables and deployment scripts streamlined for future scalability

Challenges

- Configuring multi-tier VPC, security groups, and internal routing across EC2 and RDS
- Debugging ALB and WebSocket issues behind the load balancer
- Handling cookies, SameSite security, and CORS policies in production
- Keeping build/dev environments consistent across local and cloud servers

Outcomes and Future Work

- Cenovec MVP successfully deployed and publicly accessible
- Enables simulation of MPC-based challenge problems by external users
- Next steps:
  - Move RabbitMQ, Redis, and Celery to separate EC2 instances for better resource isolation
  - Enable distributed simulation workers with autoscaling
  - Improve WebSocket reliability via ALB + Nginx proxy
  - Integrate RL and data-driven control solvers (e.g., policy learning, Koopman-based MPC)
  - Add benchmarking leaderboards and reproducibility support
  - Automate horizontal scaling with ECS or Kubernetes

Acknowledgments

I would like to thank the following individuals and programs for their support throughout the AWS Cloud Computing Fellowship:

- Dr. Mahmoud Parvizi — for guidance and feedback on AWS services
- Patrick Bills — for guidance and feedback on AWS services
- The AWS Cloud Computing Fellows Program — for providing resources and learning tracks.

Thank You!

Questions?

Email: esmzadra@msu.edu