MICHIGAN STATE
U N I V E R S I T Y

Project Plan
SPARTI

The Capstone Experience

Team Amazon

Noah Girard
Matthew Maple
Jordan Mulcahy
Tyler Rozwadowski
Rose Wang

Department of Computer Science and Engineering
Michigan State University

Fall 2019

From Students…
…to Professionals

Functional Specifications

- More than 50% of items sold on Amazon are listed by 3rd party sellers
  - They typically develop custom applications to do this
- **SPARTI**: Selling Partner Application Ready to Integrate
- Our application makes the process of building a third-party seller application super simple
  - Creates all necessary resources (ie. databases), handles authentication seamlessly, and provides monitoring tools

Design Specifications

- Provide turn-key integration with AWS in a single solution

- Establish connections to Seller Partner APIs

- Utilize AWS services with minimal developer configuration and setup

- Enable application telemetry with no manual setup

- Securely handle sensitive credentials and data

Screen Mockup: Welcome Page

Screen Mockup: Manage Orders

Screen Mockup: View Seller Inventory

Screen Mockup: Seller Performance
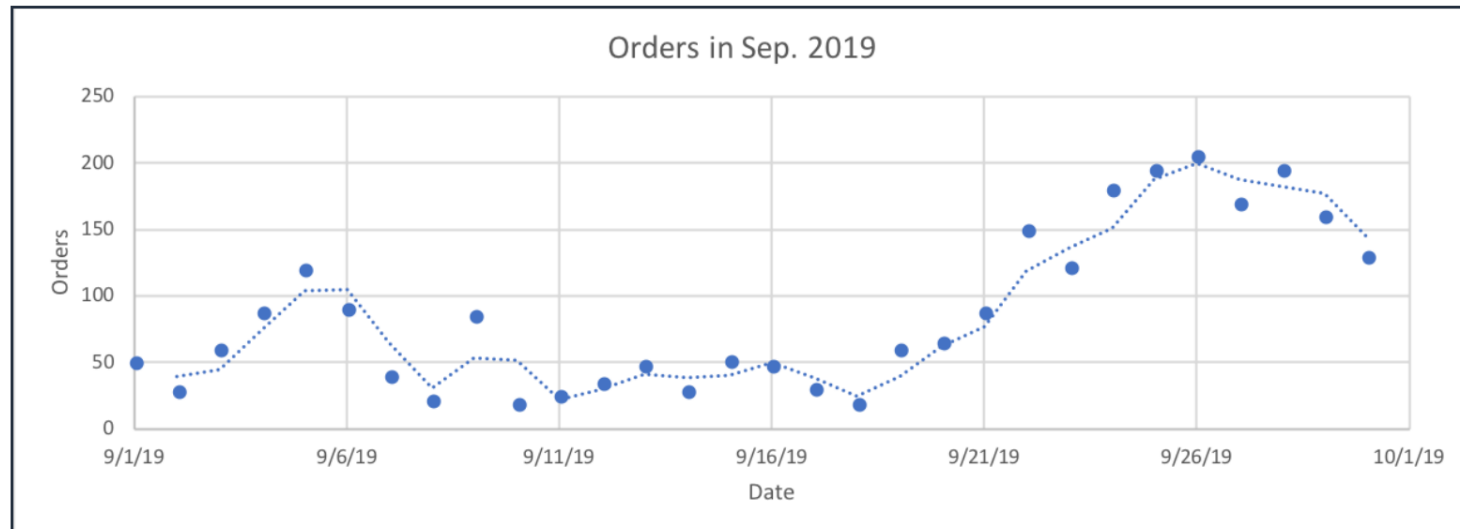
Technical Specifications

- Main application is written using C# ASP.NET Core
  - Visual web front end and backend in the same app
- SPARTI uses a variety of Amazon Web Services
  - ECR, Fargate, CloudFormation, CloudWatch, DynamoDB, Parameter Store
- Connects to the Amazon Selling Partner API (SPAPI)
  - This is how we will get 3rd Party Seller Partner data
- Entire package lives in a Docker Image
  - When deployed, this will configure all of the additional required resources (ie. DynamoDB for tokens)

System Architecture
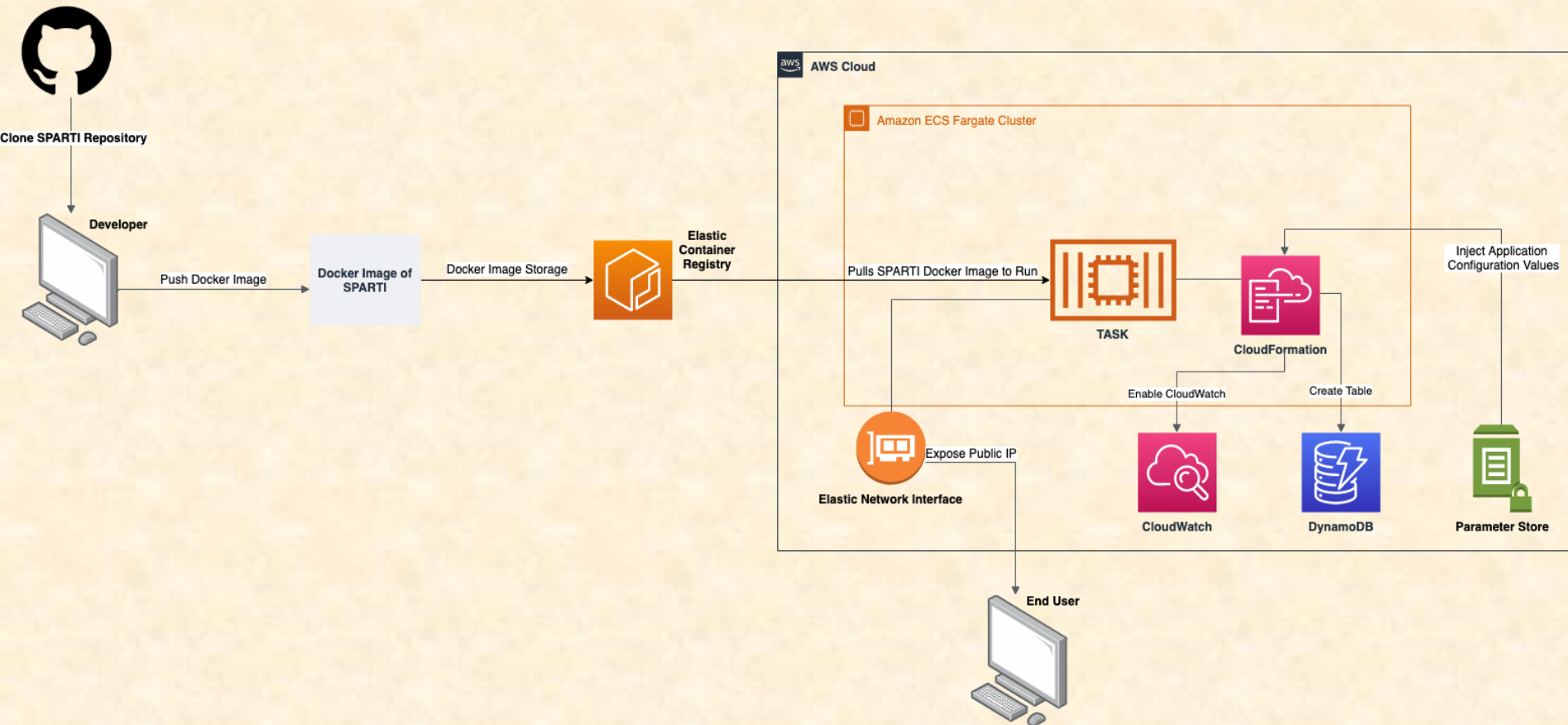
System Components

- Software Platforms / Technologies
  - ASP.NET Core
  - Docker
  - GitHub
  - Amazon Web Services
    - CloudFormation
    - DynamoDB
    - Parameter Store
    - CloudWatch
    - Fargate
    - Elastic Container Registry (ECR)
    - Elastic Container Service (ECS)

Risks

- Turn Key Integration with AWS
  - Project must have as few configuration steps as possible
  - Docker images and CloudFormation templates will be used to create all necessary components
- Securely Handling AWS Credentials and Secrets
  - Crucial that credentials used for retrieving data are stored securely
  - Utilize Parameter Store to keep credentials out of source code
- Signing Requests with AWS Signature Version 4
  - Signatures are created using user-specific parameters
  - Create a unit-testable class for generating signatures
- Automatic Integration of Amazon CloudWatch for Telemetry
  - CloudWatch integration process is typically done manually in the AWS console
  - Create a script using the AWS CLI to enable CloudWatch on cluster

Questions?

? ? ? ?

? ? ? ?

?