Simulation of Infectious Disease using Cloud Services

Akash Malhotra
Anant Gupta
Emir Nurmatbekov
Haroon Rashid
Nithish Sankaranarayanan

Overview

Expected delivery

June 1, 2020

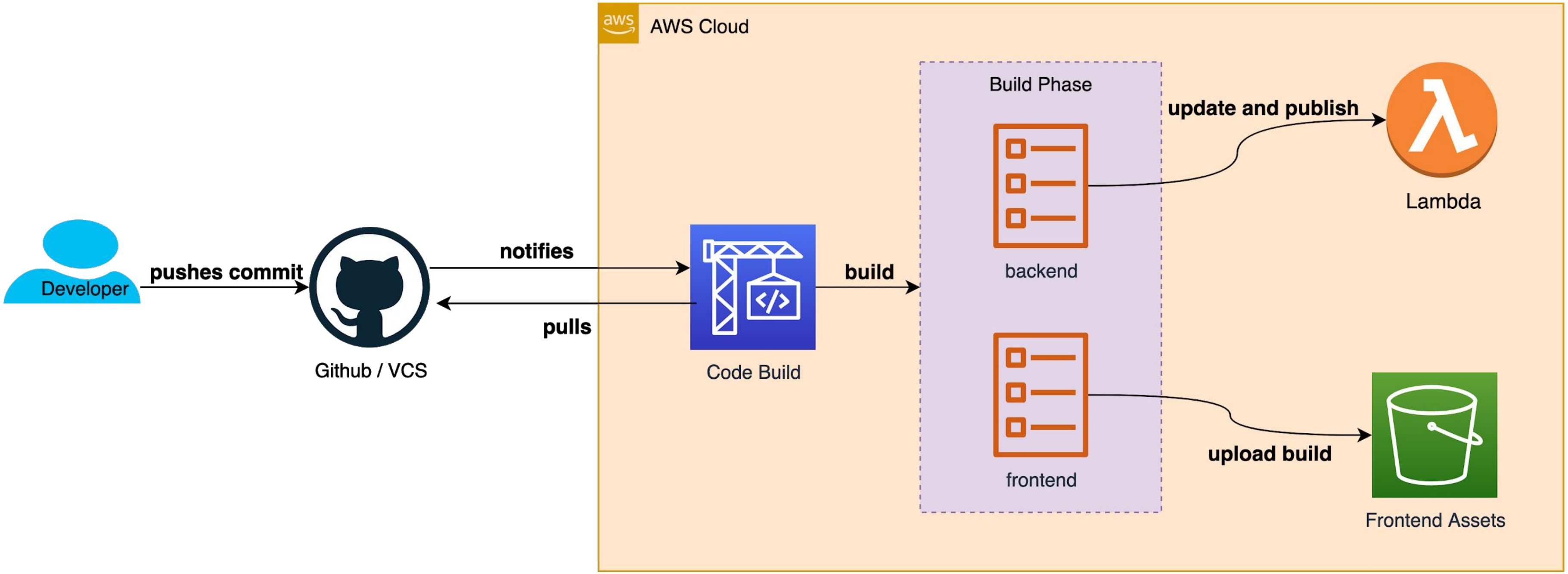
What we achieved

- Fully functional scalable simulation program using Spark.
- A sophisticated cloud architecture using AWS services at backend.
- Secure access to simulation data via front-end app.
- User-friendly visualization of simulation data.

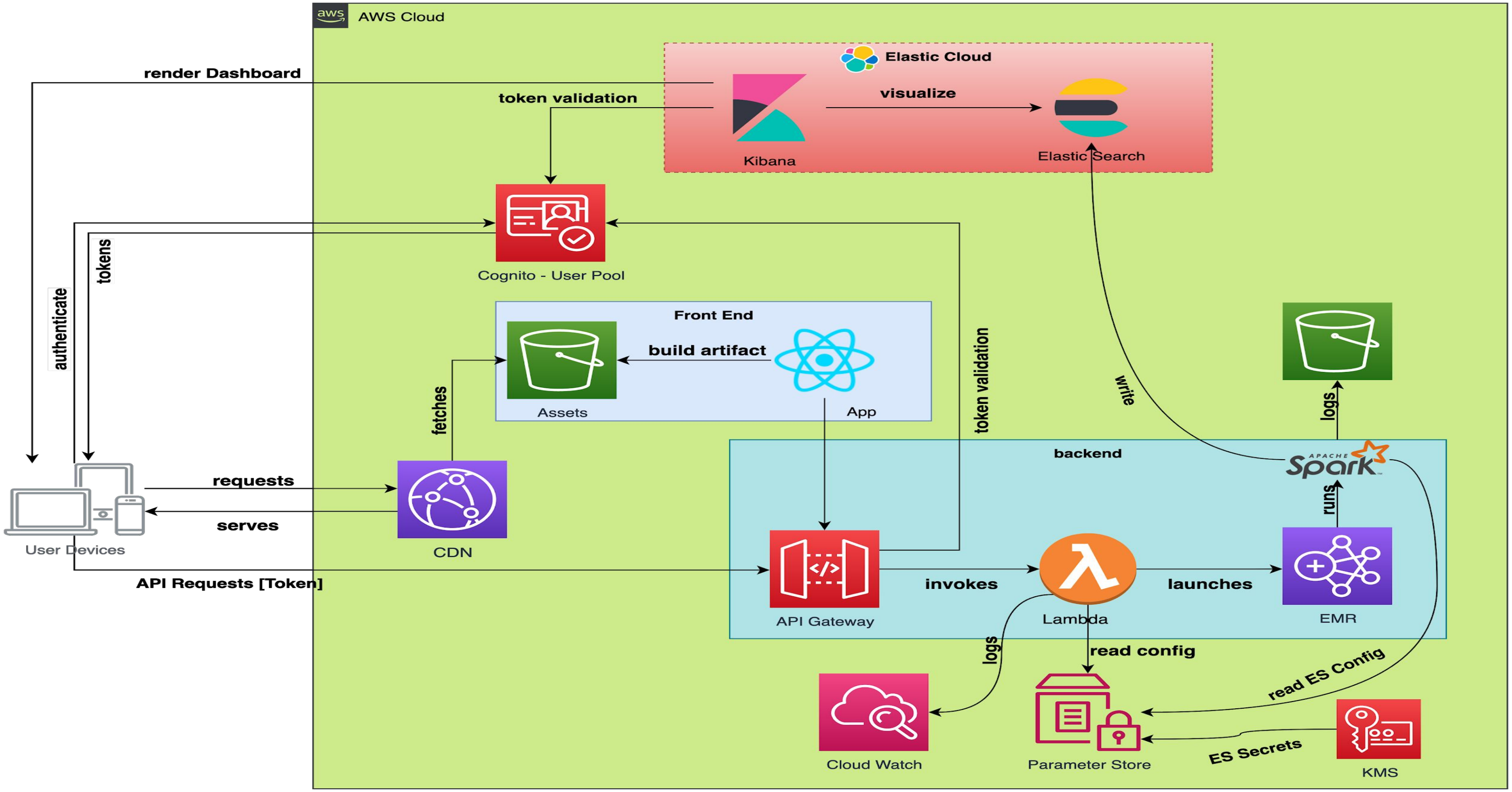
Problem statement and motivation

- To create a customizable simulation program which probabilistically simulates the spread and impact of any air-borne infectious disease.
- Use cloud services to implement scalability, fault tolerance and security.
- Select the most cost-effective solution to run the simulation.
- Display simulation data to the end user in a user friendly manner.

Proposed CI / CD Architecture



Proposed Solution (Architecture)



Infrastructure as Code (Terraform)

Requirements

- Provisioning
- Declarative style
- No vendor lock
- Immutable infrastructure paradigm

Compute

Requirements

- Parallelizable via scale out
- Should be possible to automate scalability
- Pay per use
- Minimal cold start time

Tool selection: Lambda + EMR

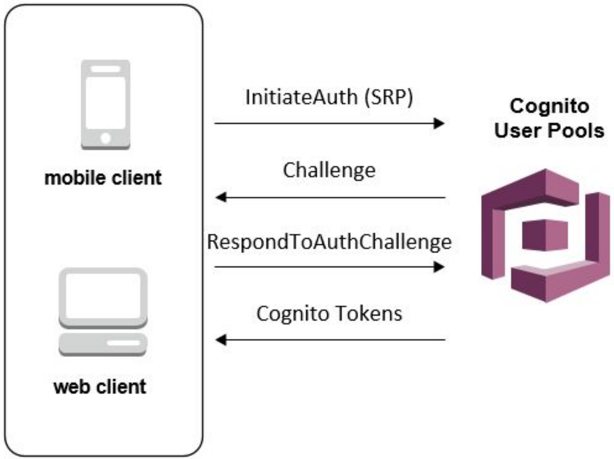
Storage

Requirements

We need storage to do the following:

1. Store the end-user data.
   a. Secure.
   b. Fault-tolerant.
2. Store simulation parameters.
   a. High speed access
   b. No need to persist
3. Store the configuration data.
   a. Persistent.

Tool selection: Cognito User pools + EMR + Parameter store

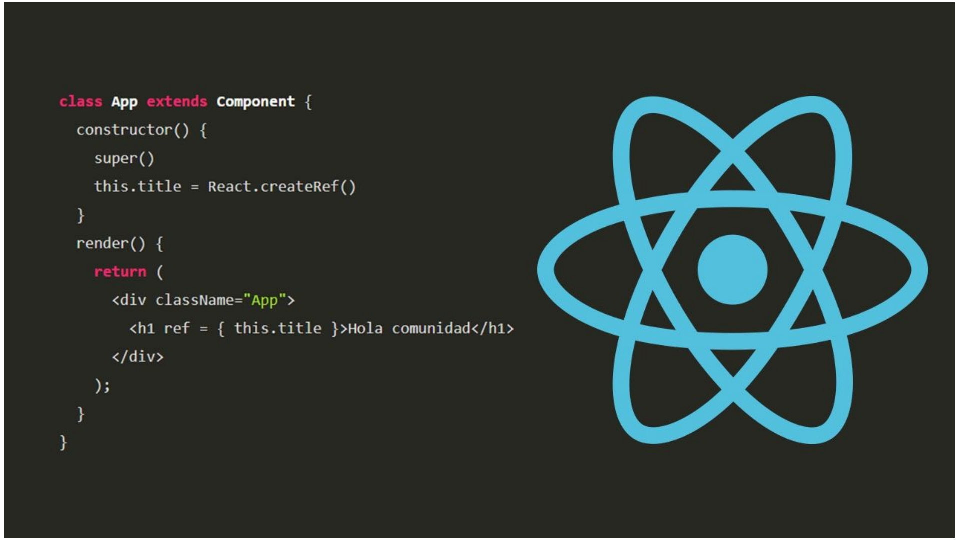


Front-end

Requirements

- Minimal coding required
- Should be integratable with security
- Should be integratable with visualization dashboard

Tool selection: React



```
class App extends Component {
  constructor() {
    super()
    this.title = React.createRef()
  }
  render() {
    return (
      <div className="App">
        <h1 ref = { this.title }>Hola comunidad</h1>
      </div>
    );
  }
}
```

Visualization

Requirements

- Should be real-time
- Customizable with a visualization language
- Automatic deployment should be possible
- Instantiation should be possible
- Mature

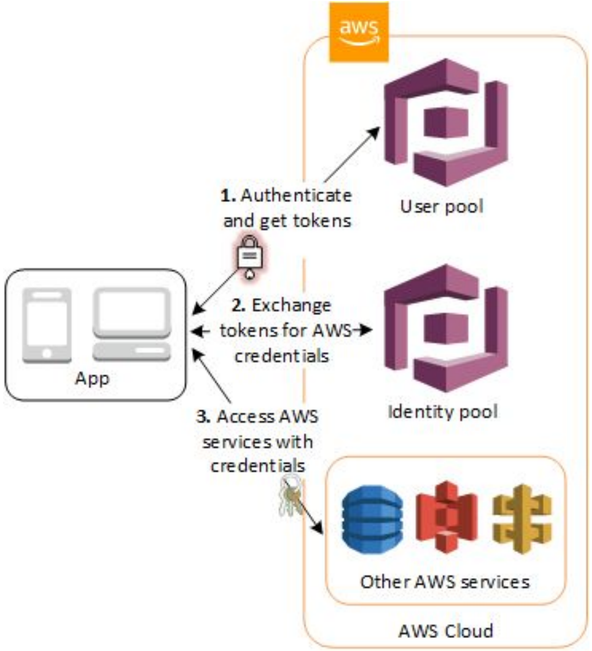
Tool selection: ElasticSearch + Kibana

Security

Requirements

- State of the art.
- Support federated sign in
- Scalable.
- Cost effective

Tool selection


aws
1. Authenticate and get tokens
User pool
App
2. Exchange tokens for AWS credentials
Identity pool
3. Access AWS services with credentials
Other AWS services
AWS Cloud

Logging

Requirements

- Centralized
  - Infrastructure
  - Application
- Interactive Querying
- Dashboards
- Custom metrics

Tool selection

Amazon CloudWatch

Configuration

Requirements

- Scalable
- Versioned configuration
- Encrypted values

Tool selection (Parameter Store + KMS)

Simulation

Requirements

- Simulation of a pandemic that can run on the cloud infrastructure
- Should be elastically scalable
- Should take different input parameters under which a simulation instance can be run
- Data should be visualized using a suitable cloud visualization service
- Features include:
  - motion adjustment of population
  - infect, kill, hospitalize and cure a person.
  - Determine the economic impact

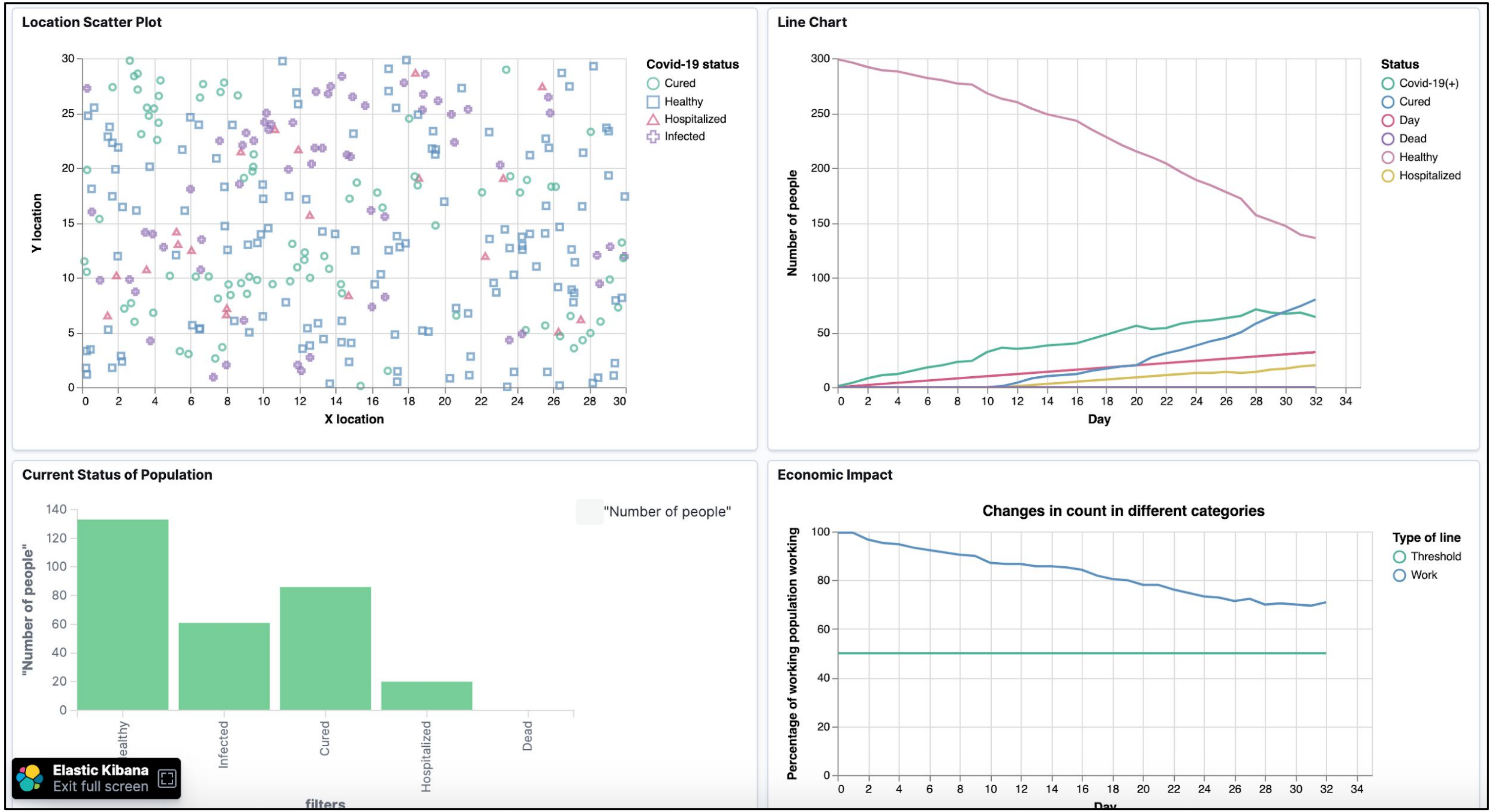
Tool selection

- Programming Language: Python
  - Easy and fast scripting with excellent online support
- Distributed computations using Spark
  - Widely used distributed processing engine with excellent support
- Spark Dataframes API
  - Easy code portability from pandas dataframes to spark
- Kibana for Data Visualization
  - Used over CloudWatch for

Simulation - Logic

- Data frame (**covid_df**) for whole population is maintained as a snapshot of the current status of the pandemic.
    - **X (float), Y (float), Covid-19 (String), Day (Int), status (String), work_hours (Float)**
- Another dataframe (**stats_df**) is maintained to keep track of the evolution of the disease
    - **simID(string), Day(int), Healthy(int), Covid-19(+) (int), Hospitalized (int), Cured (int), Dead (int), Work (int)**
- Simulation starts by infecting a random person, motion of the people is modelled by random walks
- Distance of all the infected people is calculated with rest of population, and any person within the radius is infected.
- Simulation is progressed by killing or hospitalizing an infected person by a given probability. A person is cured after 14 days if he has not died yet.
- Finally, the economic impact is calculated by adding the working hours of all the healthy people.

Simulation Results



Challenges

- Continuous Integration: Resolving merge conflicts.
- Integration various different services to form a robust architecture.
- Authenticating various resources to be used in the web-app.

Learnings

Various tools to automate or semi-automate the big data applications.

CI/CD and agile methodology of software engineering.

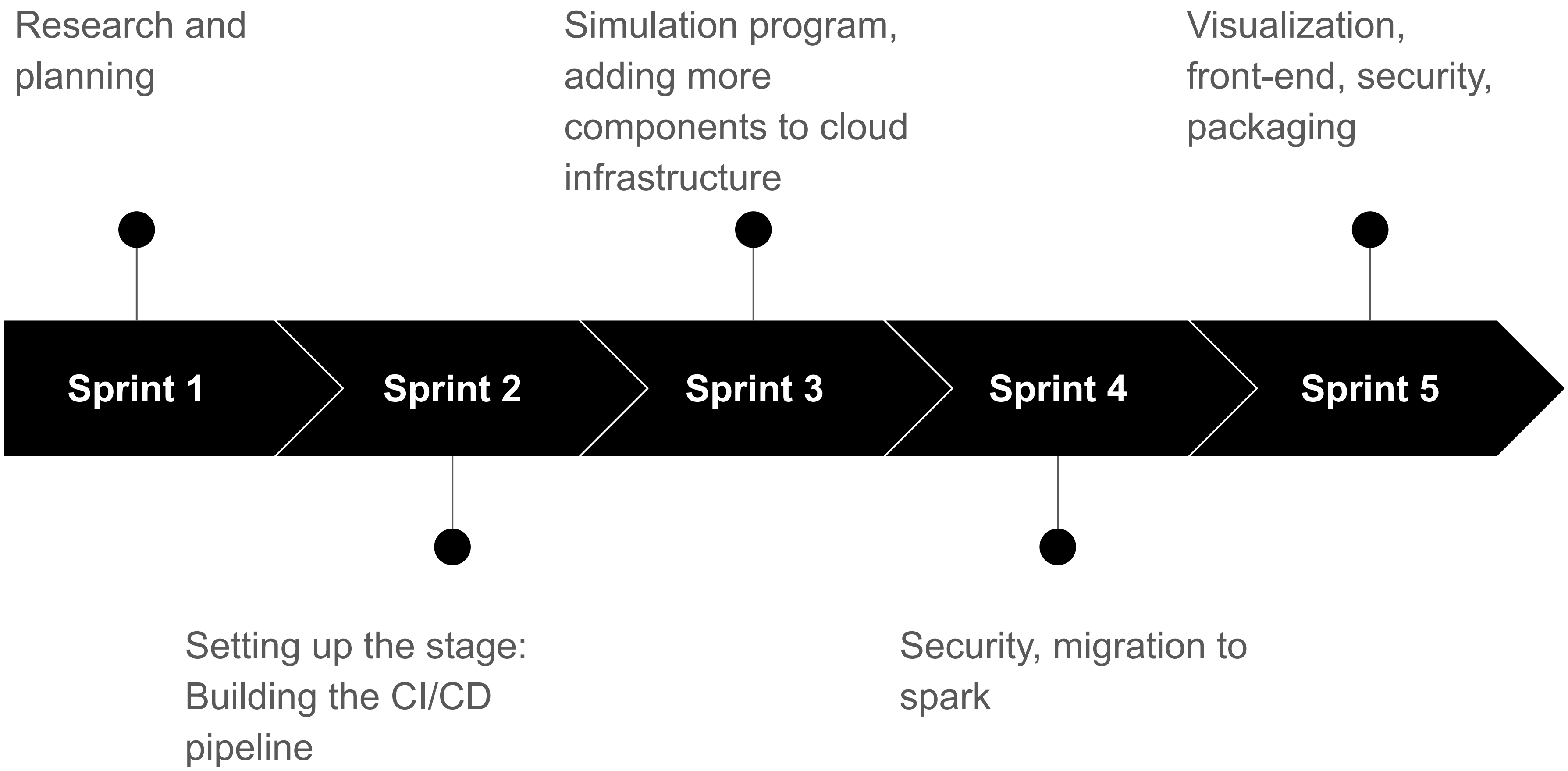
The basics of state of the art security flows like OAuth2 and OIDC.

The basics of IaC.

Understanding of building products using cloud services

Soft skills like project management and communication with team-mates.

Timeline of our project



Sprint 1

Research and planning

Sprint 2

Setting up the stage: Building the CI/CD pipeline

Sprint 3

Simulation program, adding more components to cloud infrastructure

Sprint 4

Security, migration to spark

Sprint 5

Visualization, front-end, security, packaging

Our vision for future

1. User state tracking
2. Queue for fault tolerance