GeoCache App

That-Team:

Jordan Severance **Shawyn Kane**

Johnny Sheerin **Christopher Hill**

Daniel Wallace

Geocaching: a primer

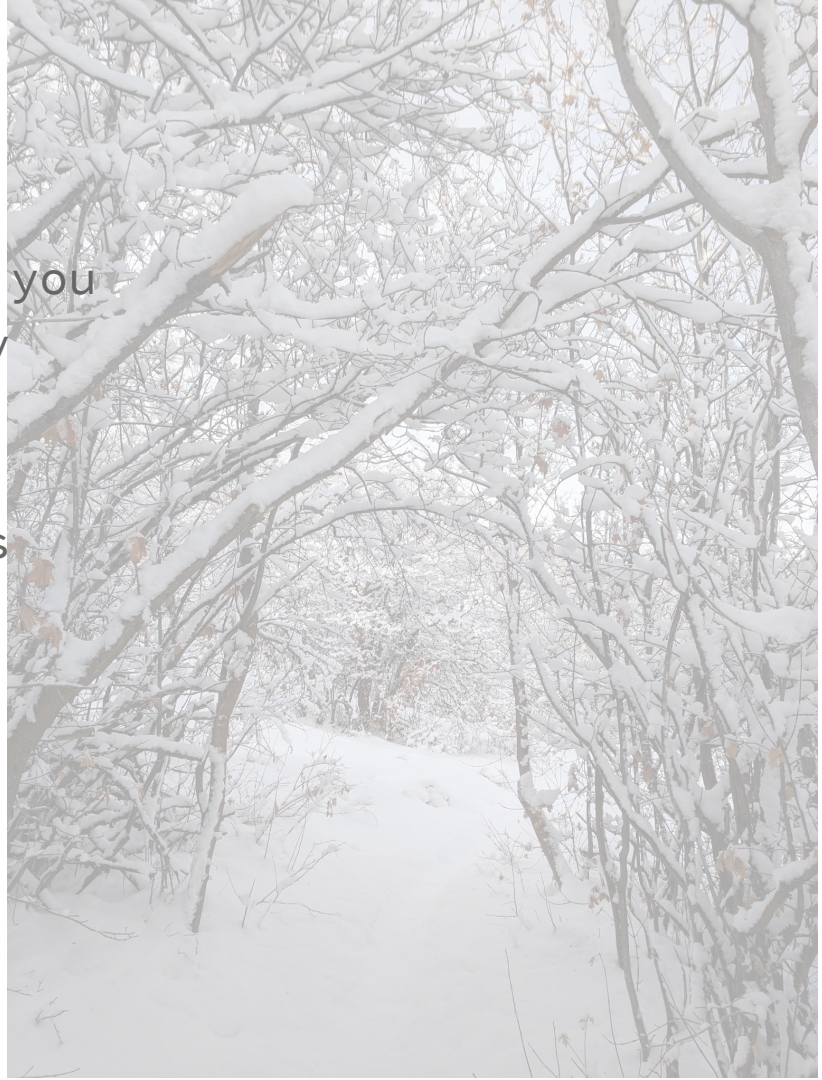
- What is geocaching?
- Who does this?
- When did this craziness begin?
- Why geocache?
- How do I get involved?

GeoCache App

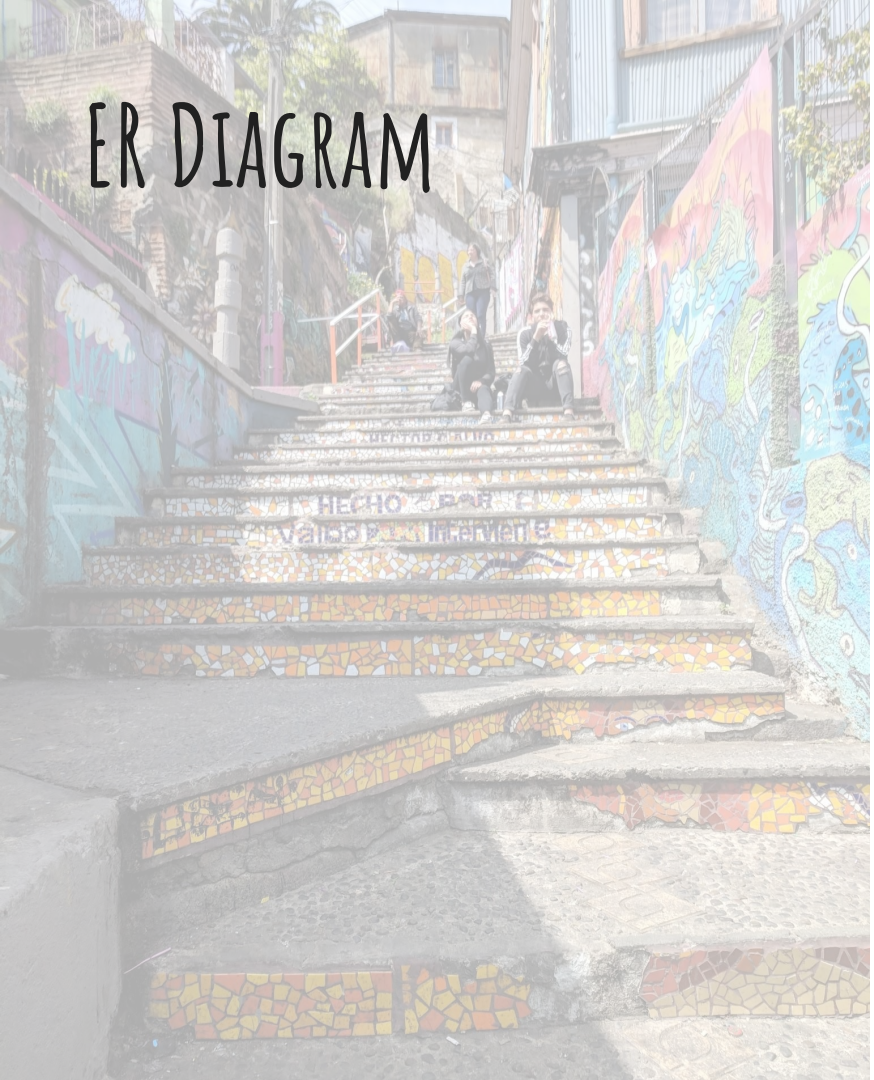
Brought to you by That-Team

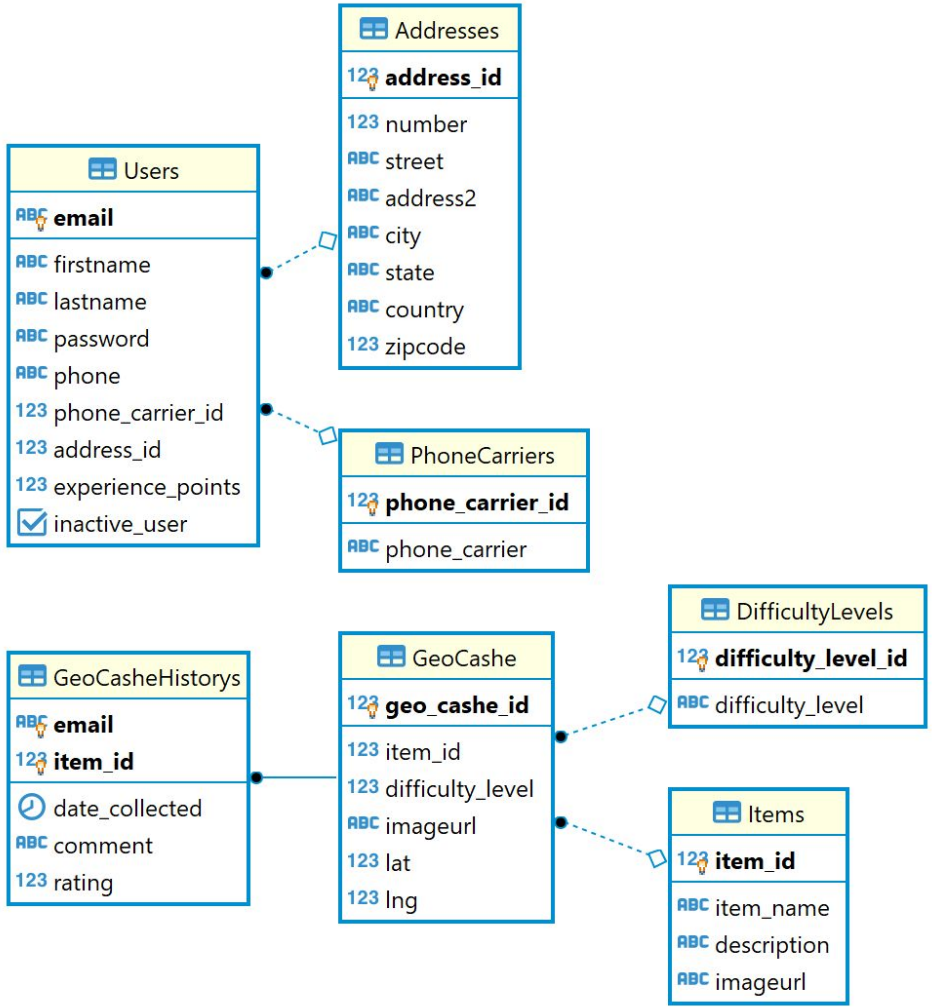
Why do I need GeoCache App?

- Quick access to geocaches near you
- Create new geocaches on the fly
- Worldwide, digital logbook
- Community driven ratings
- Badges and fake internet points



ER Diagram

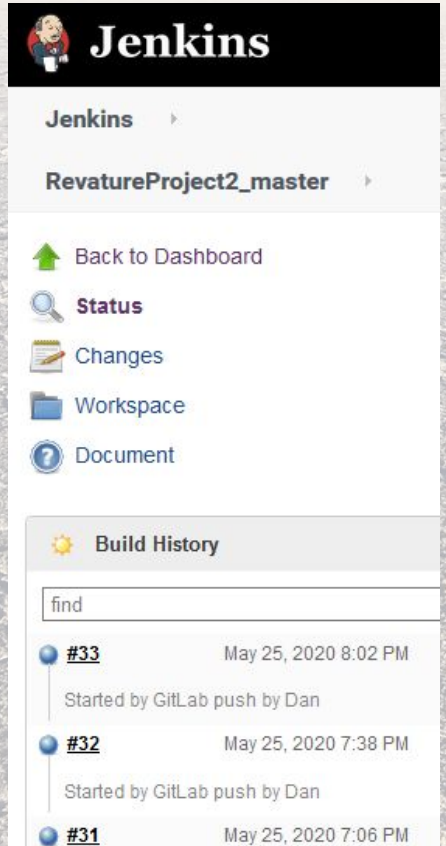



Addresses
address_id
number
street
address2
city
state
country
zipcode
Users
email
firstname
lastname
password
phone
phone_carrier_id
address_id
experience_points
inactive_user
PhoneCarriers
phone_carrier_id
phone_carrier
DifficultyLevels
difficulty_level_id
difficulty_level
GeoCashe
geo_cashe_id
item_id
difficulty_level
imageurl
lat
lng
GeoCasheHistorys
email
item_id
date_collected
comment
rating
Items
item_id
item_name
description
imageurl

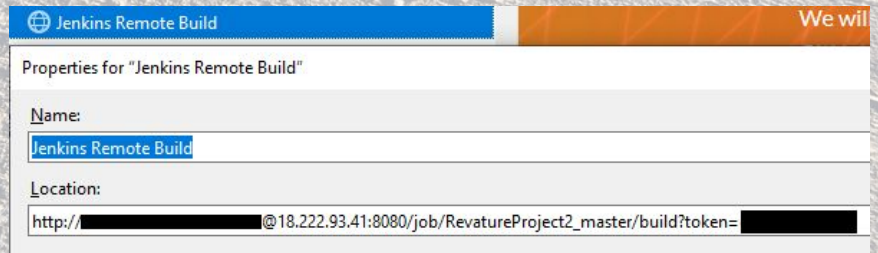
Hibernate

- Difficult to configure.
- Hibernate maps the model classes to the database tables.
- Setting up the environment variables

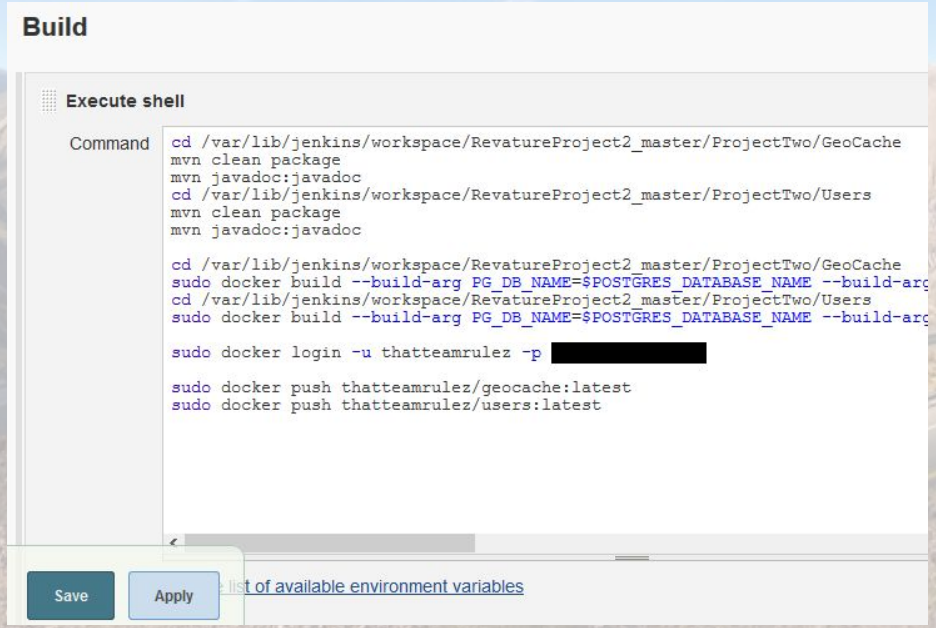
Jenkins Automation Server



- Configured to provide a simple CI/CD pipeline.
- Hosted on an AWS EC2 web server running Ubuntu Linux.
- Automated builds via:
  - Remote URL call (HTTP Post request) with user credentials and access token. Can be activated in scripts or from any web browser.
  - WebHooks - GitLab source code repo, updates to that-team's master branch.



Jenkins Automation Server - Job



- Runs the Maven package phase on our two microservices.
- Generates their Javadocs.
- Builds a Docker image for each and includes the required environment variables.
- Pushes the new Docker images to that-team's Docker Hub repository.

Test Coverage

Coverage: All in ThatTeam_GeoCache (3)

63% classes, 53% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
antlr			
com	63% (7/11)	60% (42/69)	53% (147/275)
java			
javafx			
javassist			
javax			
jdk			
META-INF			
net			
netscape			
oracle			
org			
sun			

Coverage: All in ThatTeam_Users

57% classes, 45% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
antlr			
com	57% (4/7)	52% (23/44)	45% (116/256)
java			
javafx			
javassist			
javax			
jdk			
META-INF			
net			
netscape			
oracle			
org			

Kubernetes Clusters

- An open-source system for automating deployment, scaling, and management of containerized applications.

- Groups together containers that make up an application into units to easily manage and discover.

NGINX Ingress Controller

- Ingress controller exposes your application to external users.
- NGINX Ingress controller offers
  - SSL Termination
  - Layer 7 Routing
  - Load Balancing

That Team's Kubernetes Cluster

- **GeoCache application = 2 Pods.**
- Both serving a unique and specific purpose.
    1. **ThatTeam_Users**
        a. Deployment.yaml
        b. Service.yaml
    2. **ThatTeam_GeoCache**
        a. Deployment.yaml
        b. Service.yaml

That Team Service & Deployment

ThatTeam_Users

```yaml
---
kind: Service
apiVersion: v1
metadata:
  name: users-service
spec:
  type: ClusterIP
  selector:
    app: ThatTeam_Users
  ports:
    - port: 3050 # used by other pods
      targetPort: 8080 # port exposed on the con
      protocol: TCP
```

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: users-deployment
  labels:
    app: users-dep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ThatTeam_Users
  template:
    metadata:
      labels:
        app: ThatTeam_Users
    spec:
      containers:
        - name: users
          image: thatteamrulez/users
          ports:
            - containerPort: 8080 #due to containers be
```

- ClusterIP: expose the service on a cluster-internal IP.
- To reach the clusterIP from an external source:
  - Open a Kubernetes proxy between the external computer and the cluster

That Team Ingress Configuration

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: thatteam-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  #namespace: default
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /ThatTeam_Users(/|$)(.*)
            backend:
              serviceName: users-service
              servicePort: 3050 #This port needs to match the PORT of hte service mapped
          - path: /ThatTeam_GeoCache(/|$)(.*)
            backend:
              serviceName: geocache-service
              servicePort: 3051 #This port needs to match the PORT of hte service mapped
```

- Acts as a reverse proxy and single entry-point to your cluster that routes the request to different services.
- Ingress is exposed to the outside of the cluster via ClusterIP and routes incoming traffic accordingly

Next Steps

- Push notifications
- Deploy cluster on GCP or other cloud provider
- JenkinsX
- Add SonarCloud and email notifications to Jenkins.
- Integrate geocaching.com API
- Integrating front-end into cluster environment

Takeaways

- Kubernetes is hard
- Kubernetes NGINX rewriting takes and strips away the root level of each module thus hindering the paths of CSS/Javascript

Questions?