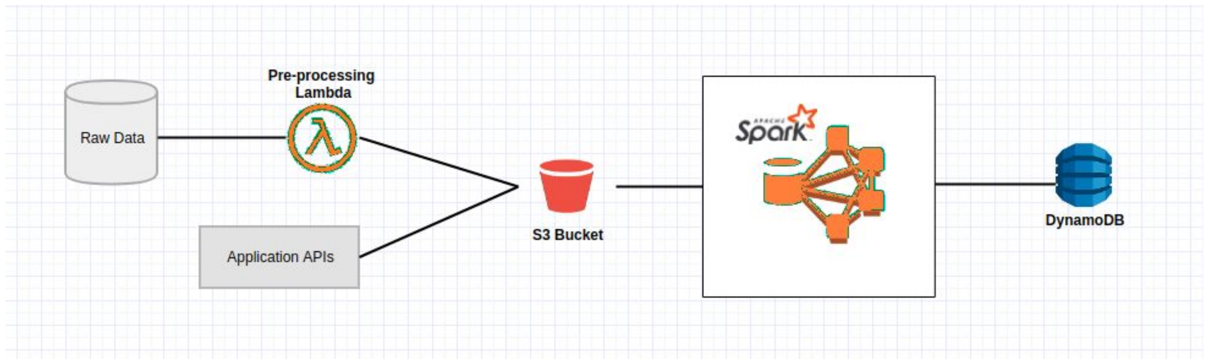
CHECKPOINT 1
DATA CENTER SCALE COMPUTING

PROBLEM DESCRIPTION

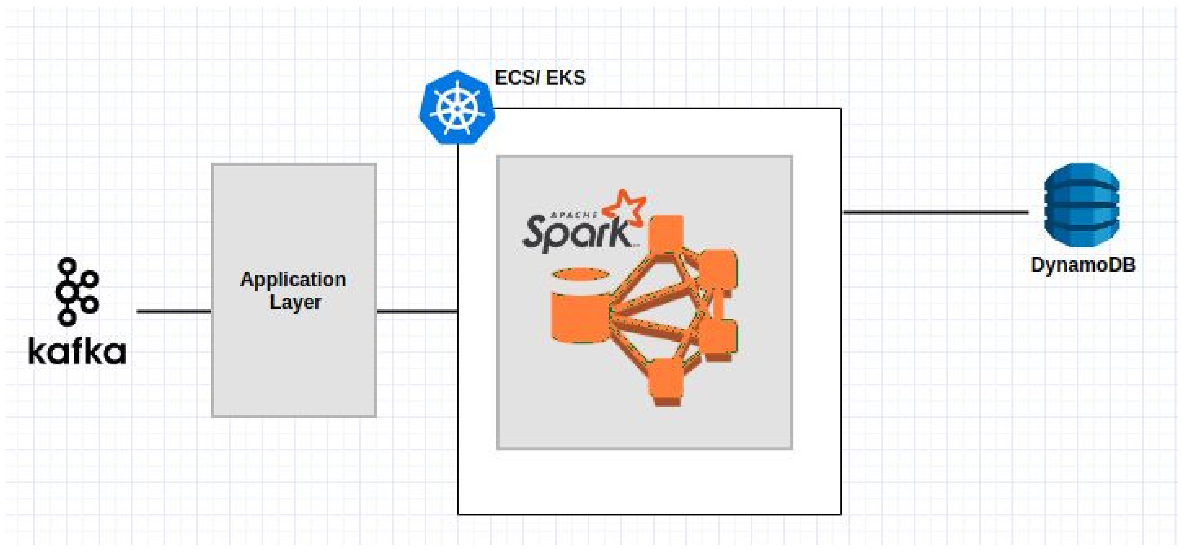
- Through this project, we aim to focus on scalability. In an application like Uber or any taxi service app, it is pertinent that the user requests of a cab are met in an acceptable amount of time. If the application delays a user's response a lot or fails to return a response, this could lead to a reduced popularity for the application.
- The application should be able to handle a large number of requests at the same time, to promote user satisfaction and also for monetary purposes.
- We plan to explore Kubernetes/Docker services (EKS/ECS) to handle multiple requests parallelly. We also plan to split independent requirements of a request and delegate them to different microservices so that they are computed parallelly and independently.

HIGH-LEVEL ARCHITECTURE



Pre-Processing data and storing the structured data in DynamoDB database

- Raw data would be put into some kind of database (Maybe S3)
- Pre-processing lambda function takes data from this database, preprocesses it and stores it in S3.
- Spark then takes care of creating tables and columns in DynamoDB.



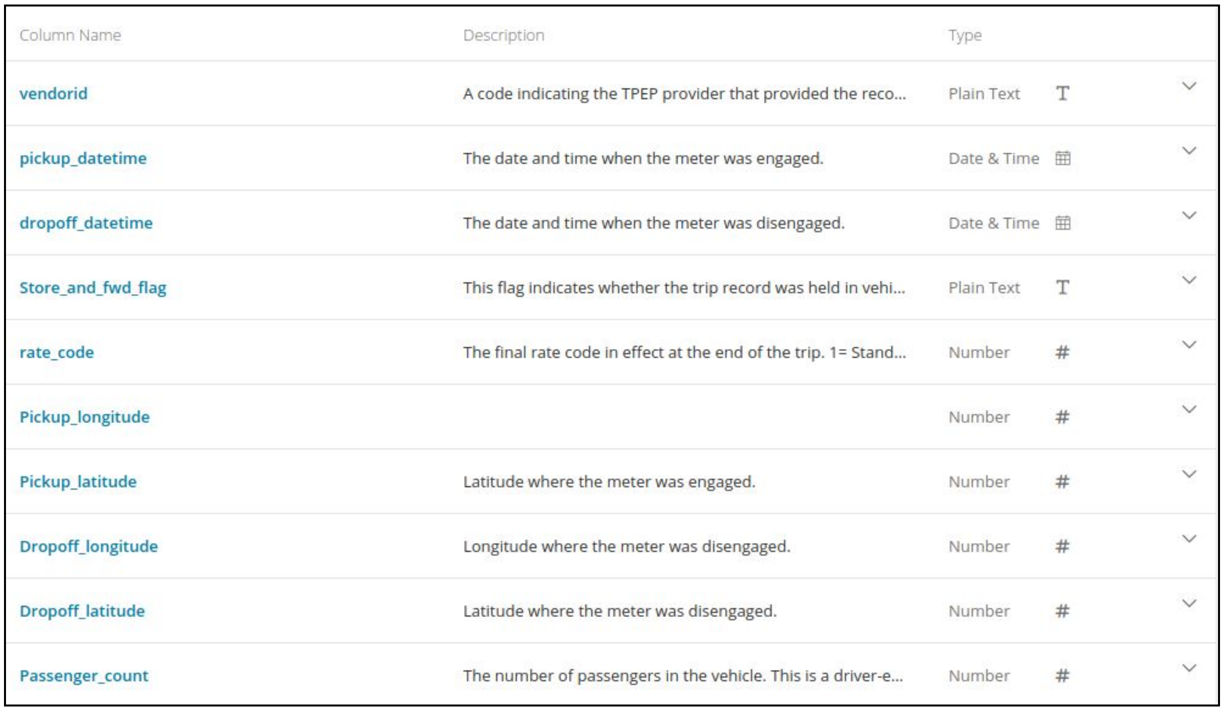
Kafka to process "real time" requests

- Kafka would contain user requests. For example, a request to search for a cab at a particular location.
- This request would then go to the application layer which would structure the user query, maybe add additional context and send it to a node with Spark capabilities in EKS/ECS (Elastic Kubernetes Service or Elastic Container Service).
- EKS is Elastic Kubernetes Service which provides an option to configure a Kubernetes cluster. ECS is Elastic Container Service which provides an option of creating multiple dockers.
- We then configure some nodes in EKS/ECS to have Spark functionalities.
- When a request reaches the EKS/ECS a new node might be created on the fly or use an existing node with spark functionalities.
- Spark takes care of interacting with the DynamoDb and querying for results which it sends back to the application layer.
- Extended Goal: Show some kind of visualization of the result.

DATASET

- We will be using NYC taxi trip data from Yellow Taxi, Green Taxi and Uber from 2014 - database
- Description:
  - Format: .csv
  - The data requires minimal preprocessing. (Removal of unnecessary metadata)
  - The dataset is static.

- The data can be exported as a CSV. There is no need of a developer account.
- The data will be stored in S3 buckets.
- Sample dataset - (Next Page)



Column Name	Description	Type
vendorid	A code indicating the TPEP provider that provided the reco...	Plain Text T
pickup_datetime	The date and time when the meter was engaged.	Date & Time
dropoff_datetime	The date and time when the meter was disengaged.	Date & Time
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehi...	Plain Text T
rate_code	The final rate code in effect at the end of the trip. 1= Stand...	Number #
Pickup_longitude		Number #
Pickup_latitude	Latitude where the meter was engaged.	Number #
Dropoff_longitude	Longitude where the meter was disengaged.	Number #
Dropoff_latitude	Latitude where the meter was disengaged.	Number #
Passenger_count	The number of passengers in the vehicle. This is a driver-e...	Number #

Some Columns that are present in the [database]

CHALLENGES

- One of the challenges is to process the incoming requests fast and gain a major boost in the processing time through our architecture.
- Our architecture consists of multiple technologies like Kafka, EKS/ECS, ElasticSearch alongwith Microservices. The biggest challenge would be to integrate all the components and run the queries.
- There are technologies like Kubernetes, ECS, EKS, Kafka which our group is not familiar with. It would be a challenge to gain holistic conceptual understanding and recognize appropriate implementation strategies.
- Our metrics of measuring performance of our project would be to calculate the average running times of 100 different requests which requires a considerable processing power and time and compare the running times with and without using ECS/EKS etc.

GENERAL TASKS AND TIMELINE

Given that we are to present two checkpoints for the given project, we have divided our timeline accordingly.

Weeks	Tasks planned
Oct 23 - Oct 30	Project Proposal ● Teaming up to discuss architecture and write proposal
Nov 1 - Nov 13	● Finalise project Design ● Start setting up EKS/ECS ● Preprocessing, data cleaning and structuring ● Setting up Kafka ● Designing the Application Layer
Nov 14- Nov 29	Project Implementation
	● Integrating Spark with EKS/ECS ● Structuring the queries from Spark on DynamoDB ● Kafka processing ● Kafka-Application Layer integration ● Application Layer ● Application Layer-EKS/ECS Integration ● Writing Unit test cases
Nov 30 - Dec 10	● Project Completion ● Documentation ● Integration testing ● Performance Measure

TASK DIVISION AND TIMELINE

S. No.	Name	Tasks	Timeline
1	Akriti Kapur	● Defining Architecture ● Overlooking preprocessing, data cleaning and structuring ● Kafka processing ● Structure the queries ● Creating test cases to produce “real-time” data	● Week 0 ● Week 1 ● Week 2 ● Week 2 ● Week 3
2	Amith Gopal	● Identifying Challenges ● Containerization using EKS/ECS ● Integration of Spark with EKS/ECS ● Integration of Application Layer with EKS/ECS ● Simple queries to test Spark-dynamoDB	● Week 0 ● Week 1 ● Week 2 ● Week 3 ● Week 3
3	Sowmya	● Identifying the AWS services ● Containerization using EKS/ECS ● Integration of Spark with EKS/ECS ● Integration of Application Layer with EKS/ECS ● Documentation	● Week 0 ● Week 1 ● Week 2 ● Week 3 ● Week 3
4	Tarunianand	● Finding datasets ● Working with Akriti on Kafka processing ● Developing application layer (Django etc.) ● Creating test cases to produce “real-time” data ● Documentation and Demonstration prep	● Week 0 ● Week 1 ● Week 2 ● Week 3 ● Week 3

CONCERN

- Since our project is heavily dependent on the AWS services, we are concerned about the potential cost incurred by the time we complete the project.

CHECKPOINT 1

Changes to the existing proposal

1. Instead of using Dynamodb as the database, we would be using MongoDB database.

New Timeline

Work done so far:

Summary of the work done

high level progress looks good

1. **Cleaning and Preprocessing of data**
2. **Storing the cleaned data in a database (MongoDB)**
3. **Setting up Kubernetes Cluster**
4. **Running a sample Spark job on the Kubernetes Cluster to verify the EKS setup**
5. **Setting up Minikube**
6. **Setting up a basic Django App**

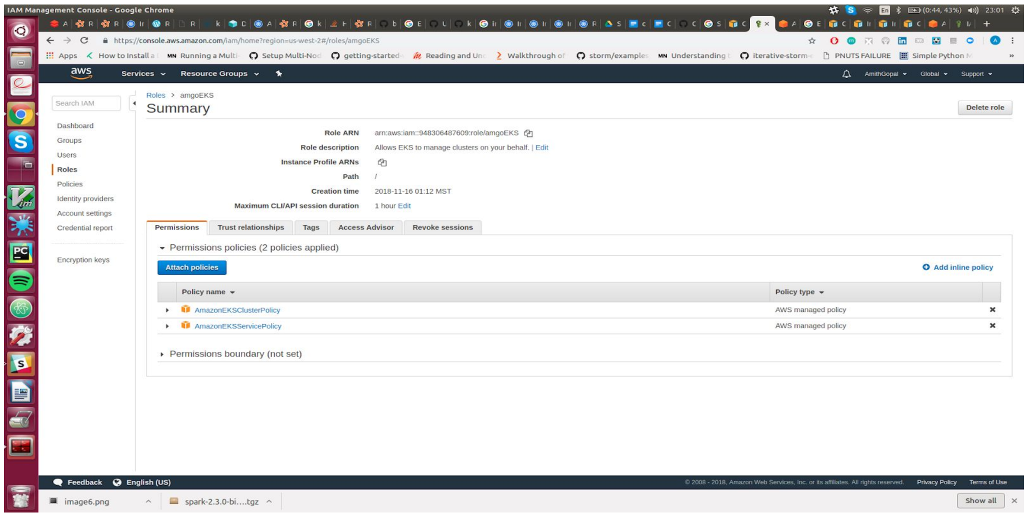
Akriti Kapur:

- Preprocessing architecture implemented
  - Lambda function to get rid of certain rows
  - Spark to operate on the sanitized data, create dataframe from the csv
  - Store the spark dataframe to a mongodb database
- Project Structured
  - Project settings, dependencies, environment variables added.

Amith Gopal

good

- I went through the tutorials, did research on prerequisites required before setting up EKS. After this, I installed kubectl, awscli and aws-iam-authenticator before proceeding to create EKS.
- I created the IAM roles, VPCs required for the creation of the EKS cluster.

- 

IAM Management Console - Google Chrome

https://console.aws.amazon.com/iam/home?region=us-west-2#/roles/amgoEKS

Roles > amgoEKS

Summary

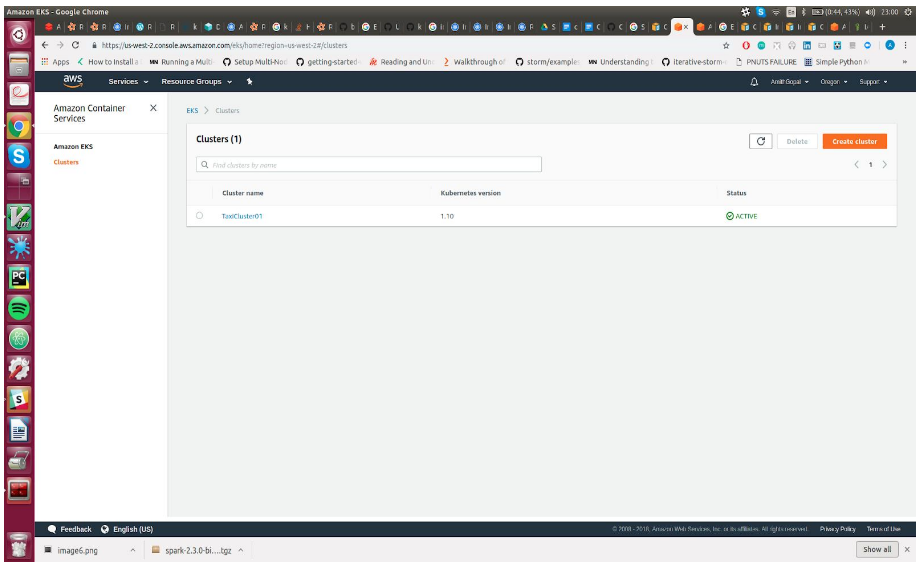
Role ARN	arn:aws:iam::948306487609:role/amgoEKS
Role description	Allows EKS to manage clusters on your behalf. | Edit
Instance Profile ARNs	
Path	/
Creation time	2018-11-16 01:12 MST
Maximum CLI/API session duration	1 hour Edit

Permissions policies (2 policies applied)

Policy name	Policy type
AmazonEKSClusterPolicy	AWS managed policy
AmazonEKSServicePolicy	AWS managed policy

Permissions boundary (not set)

- I set up the EKS cluster which is the EKS control plane after the prequisites.



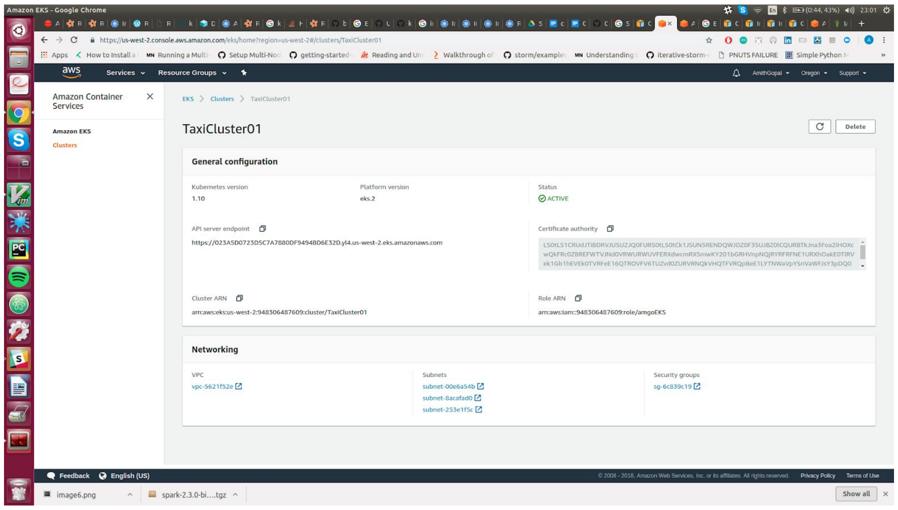
Amazon EKS - Google Chrome

https://us-west-2.console.aws.amazon.com/eks/home?region=us-west-2#/clusters

EKS > Clusters

Clusters (1)

Cluster name	Kubernetes version	Status
TaxiCluster01	1.10	ACTIVE



EKS > Clusters > TaxiCluster01

TaxiCluster01

General configuration

Kubernetes version: 1.10

Platform version: eks.2

Status: ACTIVE

API server endpoint: https://023A5D0723D5C7A7880DF9494BD6E32D.yl4.us-west-2.eks.amazonaws.com

Certificate authority:

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXc
wQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRFNE1URXhOakE0TlRV
ek1Gb1hEVEk0TVRFeE16QTROVFV6TUZvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0
```

Cluster ARN: arn:aws:eks:us-west-2:948306487609:cluster/TaxiCluster01

Role ARN: arn:aws:iam::948306487609:role/amgoEKS

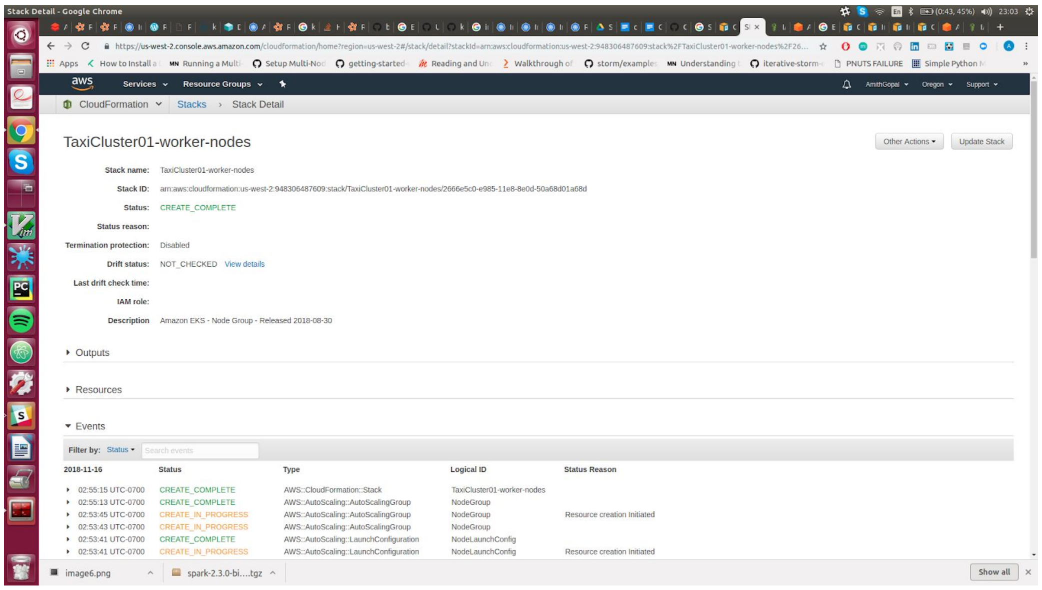
Networking

VPC: vpc-5621f52e

Subnets: subnet-00e6a54b, subnet-8acafad0, subnet-253e1f5c

Security groups: sg-6c839c19

- Next, I set up the worker nodes after the cluster was created and later configured kubectl with a .yml file to include the worker nodes.



CloudFormation > Stacks > Stack Detail

TaxiCluster01-worker-nodes

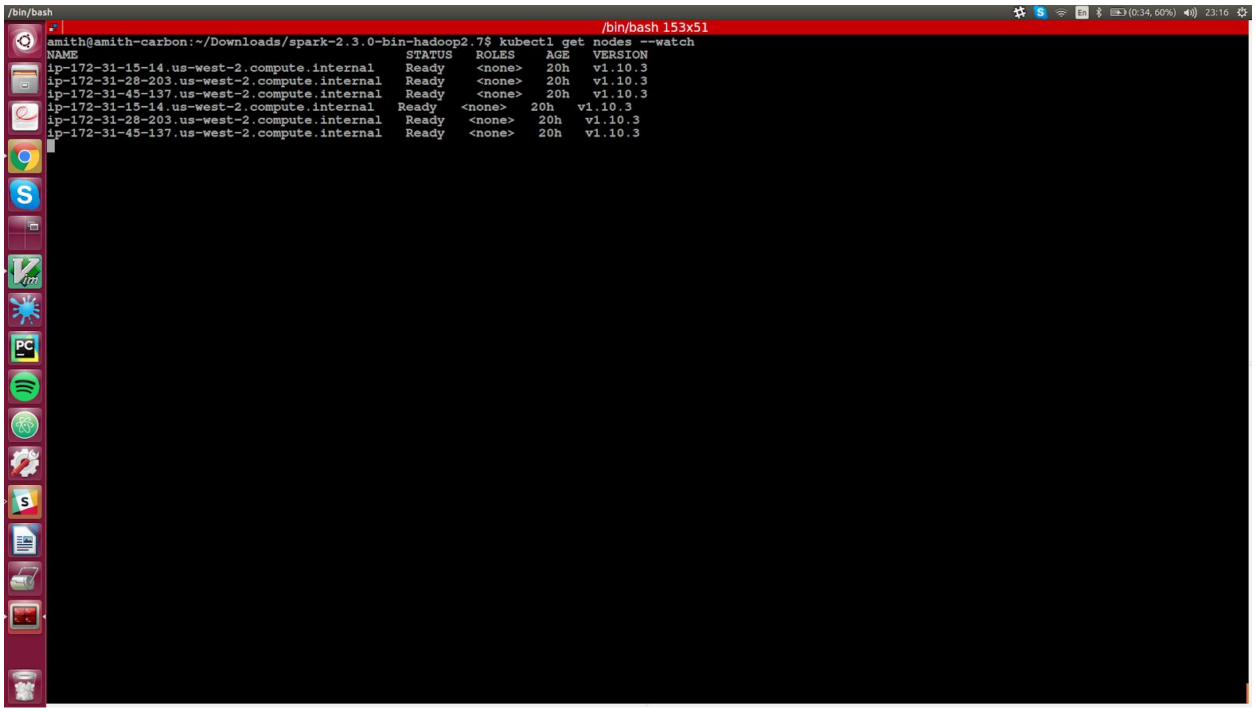
Stack name: TaxiCluster01-worker-nodes
Stack ID: arn:aws:cloudformation:us-west-2:948306487609:stack/TaxiCluster01-worker-nodes/2666e5c0-e985-11e8-8e0d-50a68d01a68d
Status: CREATE_COMPLETE
Status reason:
Termination protection: Disabled
Drift status: NOT_CHECKED View details
Last drift check time:
IAM role:
Description Amazon EKS - Node Group - Released 2018-08-30

Outputs

Resources

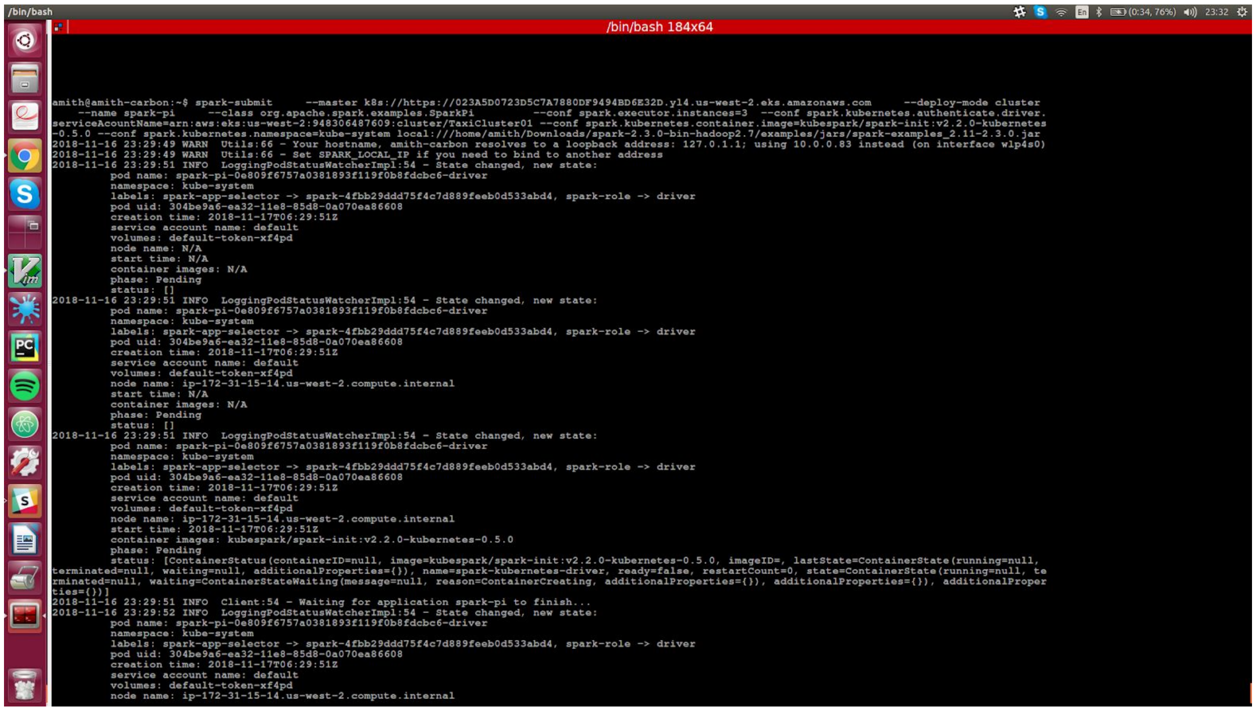
Events

2018-11-16	Status	Type	Logical ID	Status Reason
02:55:15 UTC-0700	CREATE_COMPLETE	AWS::CloudFormation::Stack	TaxiCluster01-worker-nodes	
02:55:13 UTC-0700	CREATE_COMPLETE	AWS::AutoScaling::AutoScalingGroup	NodeGroup	
02:53:45 UTC-0700	CREATE_IN_PROGRESS	AWS::AutoScaling::AutoScalingGroup	NodeGroup	Resource creation Initiated
02:53:43 UTC-0700	CREATE_IN_PROGRESS	AWS::AutoScaling::AutoScalingGroup	NodeGroup	
02:53:41 UTC-0700	CREATE_COMPLETE	AWS::AutoScaling::LaunchConfiguration	NodeLaunchConfig	
02:53:41 UTC-0700	CREATE_IN_PROGRESS	AWS::AutoScaling::LaunchConfiguration	NodeLaunchConfig	Resource creation Initiated

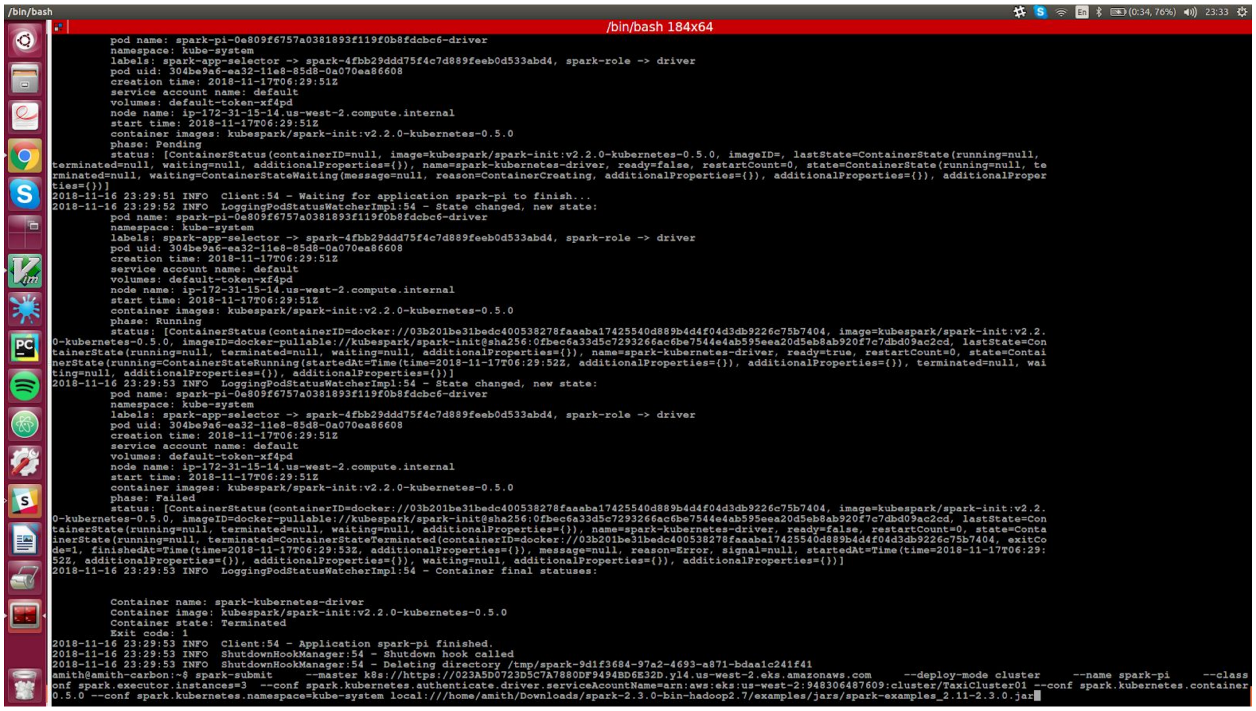


```
amith@amith-carbon:~/Downloads/spark-2.3.0-bin-hadoop2.7$ kubectl get nodes --watch
NAME                                         STATUS    ROLES     AGE    VERSION
ip-172-31-15-14.us-west-2.compute.internal   Ready     <none>    20h    v1.10.3
ip-172-31-28-203.us-west-2.compute.internal  Ready     <none>    20h    v1.10.3
ip-172-31-45-137.us-west-2.compute.internal  Ready     <none>    20h    v1.10.3
ip-172-31-15-14.us-west-2.compute.internal  Ready    <none>   20h   v1.10.3
ip-172-31-28-203.us-west-2.compute.internal  Ready    <none>    20h   v1.10.3
ip-172-31-45-137.us-west-2.compute.internal  Ready    <none>    20h   v1.10.3
```

- Lastly, I downloaded spark2.3 and and ran a spark job on the kubernetes cluster using the command shown in the “Steps-followed-for-setting-EKS.txt” and verified the working of the kubernetes cluster.



```
amith@amith-carbon:~$ spark-submit     --master k8s://https://023A5D0723D5C7A7880DF9494BD6E32D.yl4.us-west-2.eks.amazonaws.com     --deploy-mode cluster
    --name spark-pi     --class org.apache.spark.examples.SparkPi          --conf spark.executor.instances=3  --conf spark.kubernetes.authenticate.driver.
serviceAcountName=arn:aws:eks:us-west-2:948306487609:cluster/TaxiCluster01 --conf spark.kubernetes.container.image=kubespark/spark-init:v2.2.0-kubernetes
-0.5.0 --conf spark.kubernetes.namespace=kube-system local:///home/amith/Downloads/spark-2.3.0-bin-hadoop2.7/examples/jars/spark-examples_2.11-2.3.0.jar
2018-11-16 23:29:49 WARN  Utils:66 - Your hostname, amith-carbon resolves to a loopback address: 127.0.1.1; using 10.0.0.83 instead (on interface wlp4s0)
2018-11-16 23:29:49 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to another address
2018-11-16 23:29:51 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
         pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
         namespace: kube-system
         labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
         pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
         creation time: 2018-11-17T06:29:51Z
         service account name: default
         volumes: default-token-xf4pd
         node name: N/A
         start time: N/A
         container images: N/A
         phase: Pending
         status: []
2018-11-16 23:29:51 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
         pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
         namespace: kube-system
         labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
         pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
         creation time: 2018-11-17T06:29:51Z
         service account name: default
         volumes: default-token-xf4pd
         node name: ip-172-31-15-14.us-west-2.compute.internal
         start time: N/A
         container images: N/A
         phase: Pending
         status: []
2018-11-16 23:29:51 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
         pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
         namespace: kube-system
         labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
         pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
         creation time: 2018-11-17T06:29:51Z
         service account name: default
         volumes: default-token-xf4pd
         node name: ip-172-31-15-14.us-west-2.compute.internal
         start time: 2018-11-17T06:29:51Z
         container images: kubespark/spark-init:v2.2.0-kubernetes-0.5.0
         phase: Pending
         status: [ContainerStatus(containerID=null, image=kubespark/spark-init:v2.2.0-kubernetes-0.5.0, imageID=, lastState=ContainerState(running=null,
terminated=null, waiting=null, additionalProperties={}), name=spark-kubernetes-driver, ready=false, restartCount=0, state=ContainerState(running=null, te
rminated=null, waiting=ContainerStateWaiting(message=null, reason=ContainerCreating, additionalProperties={}), additionalProperties={}), additionalProper
ties={})]
2018-11-16 23:29:51 INFO  Client:54 - Waiting for application spark-pi to finish...
2018-11-16 23:29:52 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
         pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
         namespace: kube-system
         labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
         pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
         creation time: 2018-11-17T06:29:51Z
         service account name: default
         volumes: default-token-xf4pd
         node name: ip-172-31-15-14.us-west-2.compute.internal
```



```
        pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
        namespace: kube-system
        labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
        pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
        creation time: 2018-11-17T06:29:51Z
        service account name: default
        volumes: default-token-xf4pd
        node name: ip-172-31-15-14.us-west-2.compute.internal
        start time: 2018-11-17T06:29:51Z
        container images: kubespark/spark-init:v2.2.0-kubernetes-0.5.0
        phase: Pending
        status: [ContainerStatus(containerID=null, image=kubespark/spark-init:v2.2.0-kubernetes-0.5.0, imageID=, lastState=ContainerState(running=null,
terminated=null, waiting=null, additionalProperties={}), name=spark-kubernetes-driver, ready=false, restartCount=0, state=ContainerState(running=null, te
rminated=null, waiting=ContainerStateWaiting(message=null, reason=ContainerCreating, additionalProperties={}), additionalProperties={}), additionalProper
ties={})]
2018-11-16 23:29:51 INFO  Client:54 - Waiting for application spark-pi to finish...
2018-11-16 23:29:52 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
        pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
        namespace: kube-system
        labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
        pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
        creation time: 2018-11-17T06:29:51Z
        service account name: default
        volumes: default-token-xf4pd
        node name: ip-172-31-15-14.us-west-2.compute.internal
        start time: 2018-11-17T06:29:51Z
        container images: kubespark/spark-init:v2.2.0-kubernetes-0.5.0
        phase: Running
        status: [ContainerStatus(containerID=docker://03b201be31bedc400538278faaaba17425540d889b4d4f04d3db9226c75b7404, image=kubespark/spark-init:v2.2.
0-kubernetes-0.5.0, imageID=docker-pullable://kubespark/spark-init@sha256:0fbec6a33d5c7293266ac6be7544e4ab595eea20d5eb8ab920f7c7dbd09ac2cd, lastState=Con
tainerState(running=null, terminated=null, waiting=null, additionalProperties={}), name=spark-kubernetes-driver, ready=true, restartCount=0, state=Contai
nerState(running=ContainerStateRunning(startedAt=Time(time=2018-11-17T06:29:52Z, additionalProperties={}), additionalProperties={}), terminated=null, wai
ting=null, additionalProperties={}), additionalProperties={})]
2018-11-16 23:29:53 INFO  LoggingPodStatusWatcherImpl:54 - State changed, new state:
        pod name: spark-pi-0e809f6757a0381893f119f0b8fdcbc6-driver
        namespace: kube-system
        labels: spark-app-selector -> spark-4fbb29ddd75f4c7d889feeb0d533abd4, spark-role -> driver
        pod uid: 304be9a6-ea32-11e8-85d8-0a070ea86608
        creation time: 2018-11-17T06:29:51Z
        service account name: default
        volumes: default-token-xf4pd
        node name: ip-172-31-15-14.us-west-2.compute.internal
        start time: 2018-11-17T06:29:51Z
        container images: kubespark/spark-init:v2.2.0-kubernetes-0.5.0
        phase: Failed
        status: [ContainerStatus(containerID=docker://03b201be31bedc400538278faaaba17425540d889b4d4f04d3db9226c75b7404, image=kubespark/spark-init:v2.2.
0-kubernetes-0.5.0, imageID=docker-pullable://kubespark/spark-init@sha256:0fbec6a33d5c7293266ac6be7544e4ab595eea20d5eb8ab920f7c7dbd09ac2cd, lastState=Con
tainerState(running=null, terminated=null, waiting=null, additionalProperties={}), name=spark-kubernetes-driver, ready=false, restartCount=0, state=Conta
inerState(running=null, terminated=ContainerStateTerminated(containerID=docker://03b201be31bedc400538278faaaba17425540d889b4d4f04d3db9226c75b7404, exitCo
de=1, finishedAt=Time(time=2018-11-17T06:29:53Z, additionalProperties={}), message=null, reason=Error, signal=null, startedAt=Time(time=2018-11-17T06:29:
52Z, additionalProperties={}), additionalProperties={}), waiting=null, additionalProperties={}), additionalProperties={})]
2018-11-16 23:29:53 INFO  LoggingPodStatusWatcherImpl:54 - Container final statuses:


        Container name: spark-kubernetes-driver
        Container image: kubespark/spark-init:v2.2.0-kubernetes-0.5.0
        Container state: Terminated
        Exit code: 1
2018-11-16 23:29:53 INFO  Client:54 - Application spark-pi finished.
2018-11-16 23:29:53 INFO  ShutdownHookManager:54 - Shutdown hook called
2018-11-16 23:29:53 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-9d1f3684-97a2-4693-a871-bdaa1c241f41
amith@amith-carbon:~$ spark-submit     --master k8s://https://023A5D0723D5C7A7880DF9494BD6E32D.yl4.us-west-2.eks.amazonaws.com     --deploy-mode cluster     --name spark-pi     --class
onf spark.executor.instances=3  --conf spark.kubernetes.authenticate.driver.serviceAcountName=arn:aws:eks:us-west-2:948306487609:cluster/TaxiCluster01 --conf spark.kubernetes.container
0.5.0 --conf spark.kubernetes.namespace=kube-system local:///home/amith/Downloads/spark-2.3.0-bin-hadoop2.7/examples/jars/spark-examples_2.11-2.3.0.jar
```

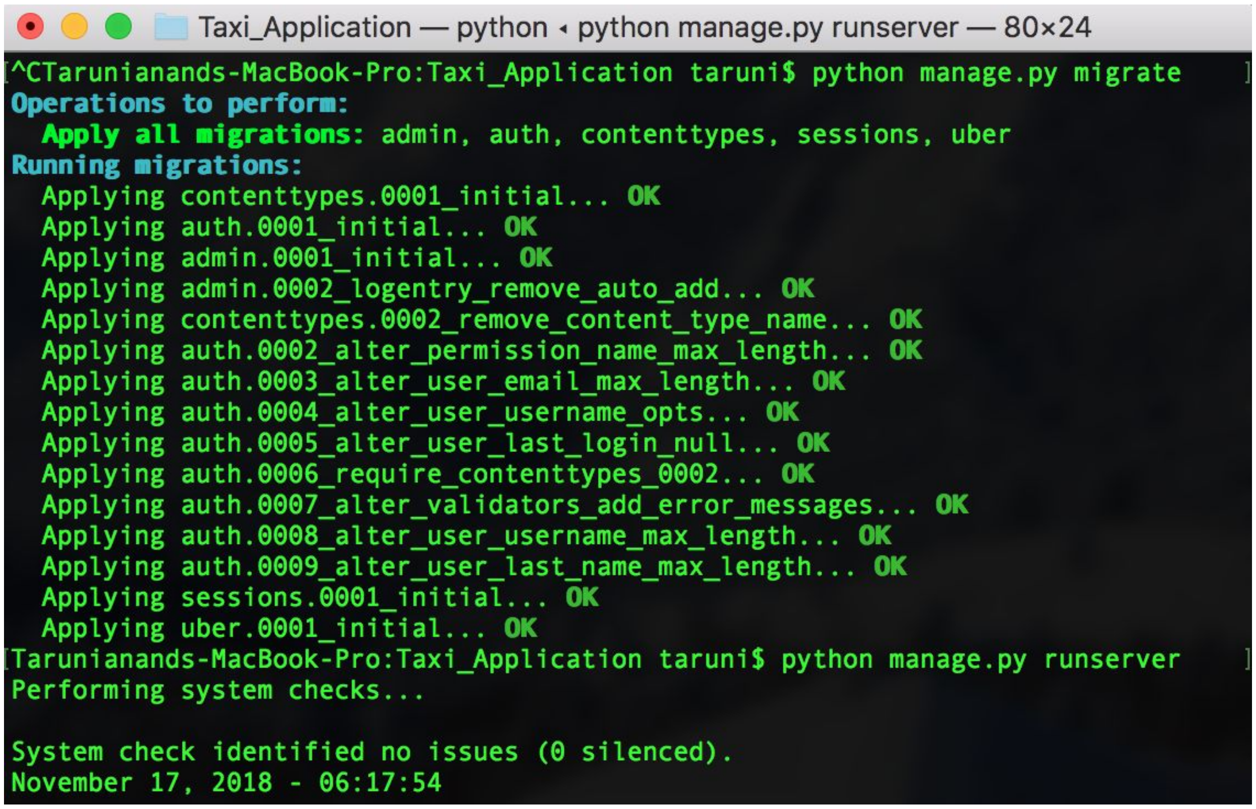
- I learnt more about Spark's integration with EKS and other functionalities and properties of EKS which can be used

Sowmya Ramakrishnan :

- Learnt Kubernetes basics, concepts, commands, creating pods and clusters with labels and selectors, scaling up/ down, zero downtime deployment.(Documentation and tutorials)
- Installed Minikube on local machine.
- Set up a cluster on AWS EKS and went through tutorials to ensure its working.

Tarunianand Muruganandan:

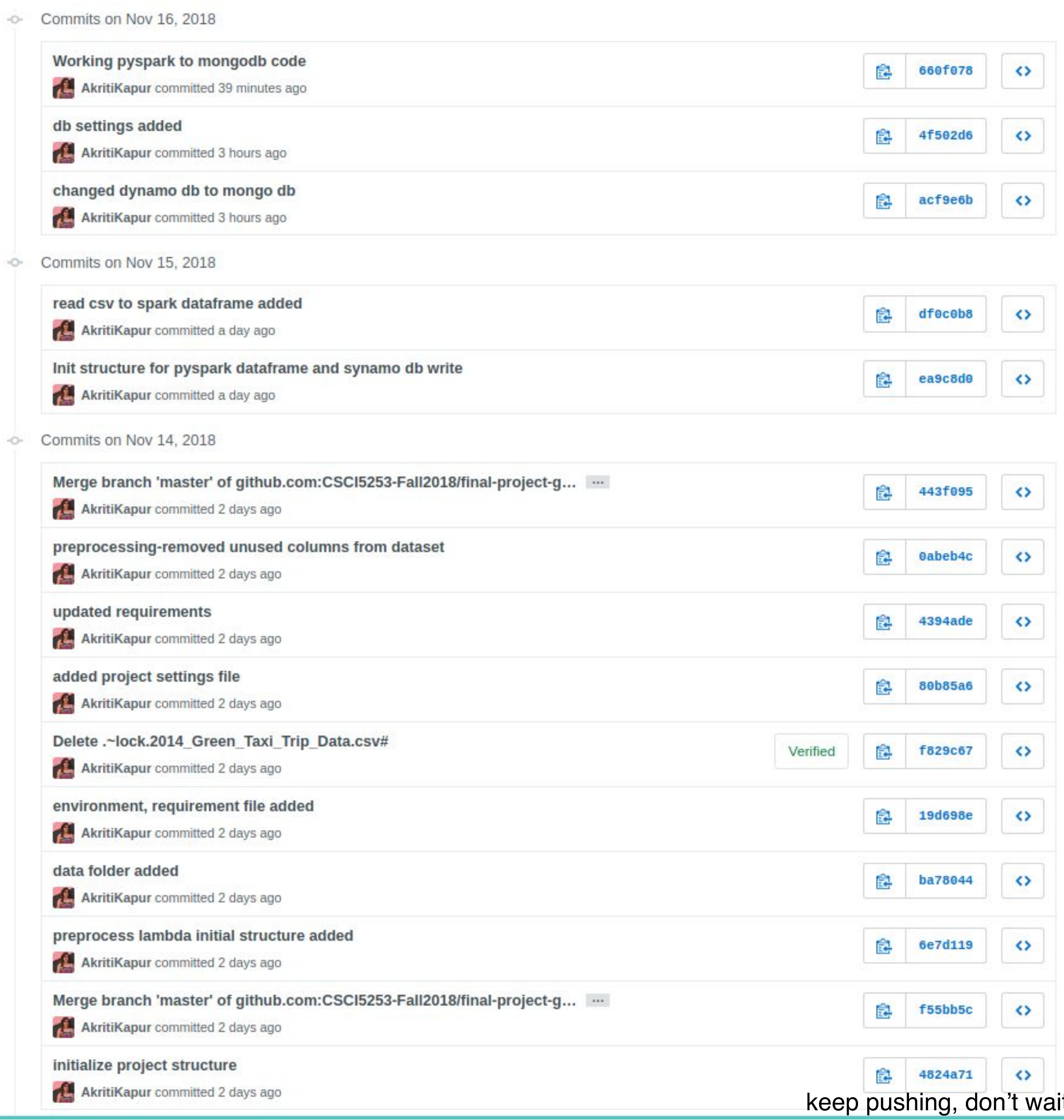
- Began with working on the basic Django framework as a starting point to build an application.
- Have been working locally to understand the backend portion, at a bare bones level.
- Reading up on Spring and other RESTful web APIs that can be used for the project.
- I have been trying to make a presentable application, but still working on setting up the right environment. I have currently managed to run all migration and start the server, with some rendering issues left to be fixed.
- Learnt how submodules work on GitHub if we push a pre-existing repo into another repo.


Taxi_Application — python ◂ python manage.py runserver — 80×24

```
[^CTarunianands-MacBook-Pro:Taxi_Application taruni$ python manage.py migrate    ]
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, uber
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying uber.0001_initial... OK
[Tarunianands-MacBook-Pro:Taxi_Application taruni$ python manage.py runserver    ]
Performing system checks...

System check identified no issues (0 silenced).
November 17, 2018 - 06:17:54
```

Github checkins



Commits on Nov 16, 2018

Working pyspark to mongodb code
AkritiKapur committed 39 minutes ago — 660f078

db settings added
AkritiKapur committed 3 hours ago — 4f502d6

changed dynamo db to mongo db
AkritiKapur committed 3 hours ago — acf9e6b

Commits on Nov 15, 2018

read csv to spark dataframe added
AkritiKapur committed a day ago — df0c0b8

Init structure for pyspark dataframe and synamo db write
AkritiKapur committed a day ago — ea9c8d0

Commits on Nov 14, 2018

Merge branch 'master' of github.com:CSCI5253-Fall2018/final-project-g…
AkritiKapur committed 2 days ago — 443f095

preprocessing-removed unused columns from dataset
AkritiKapur committed 2 days ago — 0abeb4c

updated requirements
AkritiKapur committed 2 days ago — 4394ade

added project settings file
AkritiKapur committed 2 days ago — 80b85a6

Delete .~lock.2014_Green_Taxi_Trip_Data.csv#
AkritiKapur committed 2 days ago — Verified — f829c67

environment, requirement file added
AkritiKapur committed 2 days ago — 19d698e

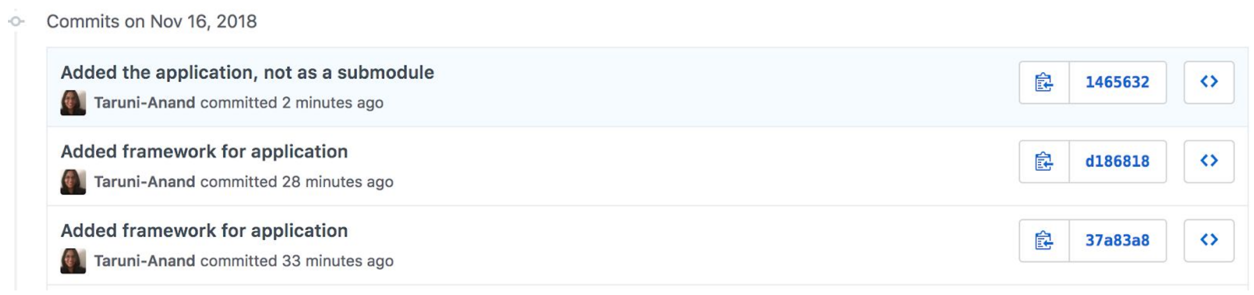
data folder added
AkritiKapur committed 2 days ago — ba78044

preprocess lambda initial structure added
AkritiKapur committed 2 days ago — 6e7d119

Merge branch 'master' of github.com:CSCI5253-Fall2018/final-project-g…
AkritiKapur committed 2 days ago — f55bb5c

initialize project structure
AkritiKapur committed 2 days ago — 4824a71

keep pushing, don't wait until last minute!

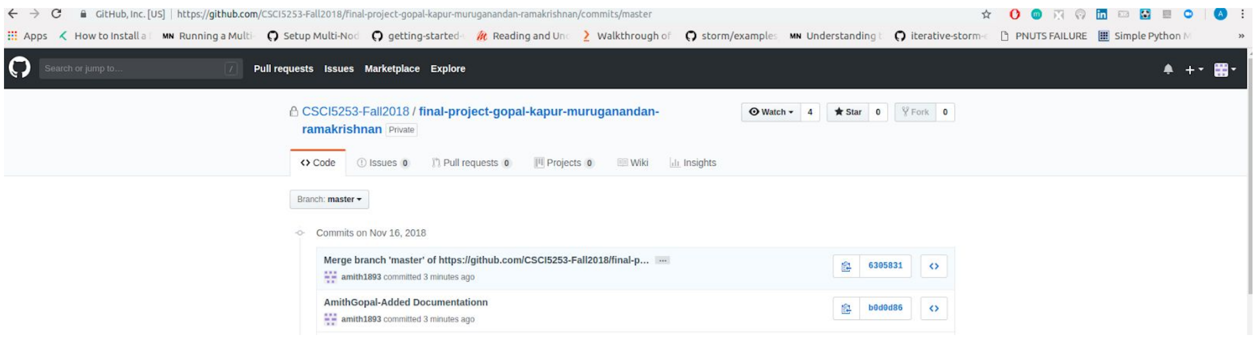


Commits on Nov 16, 2018

Added the application, not as a submodule
Taruni-Anand committed 2 minutes ago — 1465632

Added framework for application
Taruni-Anand committed 28 minutes ago — d186818

Added framework for application
Taruni-Anand committed 33 minutes ago — 37a83a8



https://github.com/CSCI5253-Fall2018/final-project-gopal-kapur-muruganandan-ramakrishnan/commits/master

CSCI5253-Fall2018 / final-project-gopal-kapur-muruganandan-ramakrishnan Private

Commits on Nov 16, 2018

Merge branch 'master' of https://github.com/CSCI5253-Fall2018/final-p... — amith1893 committed 3 minutes ago — 6305831

AmithGopal-Added Documentationn — amith1893 committed 3 minutes ago — b0d0d86

	Name	Tasks	Timeline
1	Akriti Kapur	● Kafka processing ● Structure the queries ● Creating test cases to produce “real-time” data ● Simple queries to test Spark-dynamoDB	● 19th Nov week ● 26th Nov week ● 26th Nov week
2	Amith Gopal	● Running custom application through Spark on EKS ● Running different tests to test Spark on EKS further ● Trying minikube as a fallback if we encounter issues in EKS ● Integration of Application Layer with EKS/ECS	● 19th Nov week ● 19th Nov week ● 26th Nov week ● 26th Nov week
3	Sowmya	● Integration of Spark with Kubernetes ● Integration of Application Layer with EKS/ECS ● Documentation	● 19th Nov week ● 26th Nov week ● 26th Nov week
4	Tarunianand	● Developing application layer, check how the backend would change to accomodate the rest of the project. ● Trying different frameworks for the application (Spring, Flask etc.) ● Integrating application layer with EKS/ECS ● Creating test cases to produce “real-time” data ● Documentation and Demonstration prep	● 19th Nov week ● 19th Nov week ● 26th Nov week ● 26th Nov week ● 26th Nov week

Costs

Costs incurred

Estimated Costs for future use

Database Management

- We did not have to do a lot of cleaning on the database. There were few columns we did not need and hence removed it from the database.
- As our focus in the project is on scalability, we need to perform certain tasks on the kubernetes cluster. We may need additional columns depending on the tasks we need spark to perform on this database. These additional columns if needed, will be added to the database in the pre-processing step while using spark to write the data frame to MongoDB.

which tasks?

Challenges faced:

- Setting up EKS cluster was a challenge since there were a lot of steps to be followed and lot of prerequisites to be satisfied.
- Running spark on EKS. Had to figure out the permissions to send spark jobs from a local machine to a remote EKS cluster.
- Storing data from Spark into DynamoDB was difficult since we couldn't find any documentation online for saving pyspark dataframe to DynamoDB which is why we switched to MongoDB where the whole data storage pipeline is successfully constructed.
- We will be testing minikube as a fallback if there are any issues with EKS.

good work, please don't wait until last second to make progress