Movie Ticket Reservation System

Team 7
Mitali Salvi
Pinho Wang

Pre requisites:

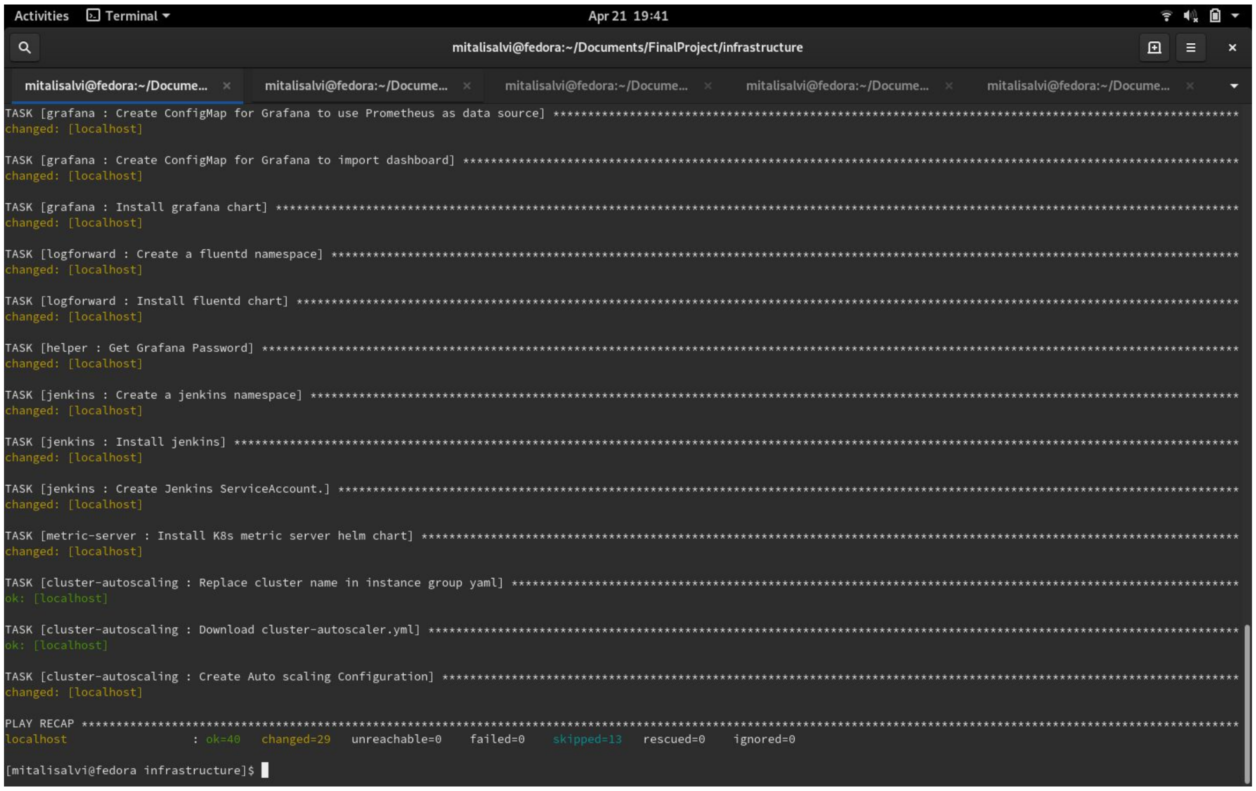
- Ansible is installed and configured to connect to AWS
- AWS credentials are configured in your system
- SSH key is created for passing to kops
- AWS credentials for account are available with you
- fluentD user is created in AWS and you have the credentials
- Domain is set in Route 53 with private hosted zone
- S3 bucket is created in AWS to store Kops configuration

1. Setup Infrastructure

- Take the latest pull from the repo - https://github.com/mitali-salvi/csye7200-infrastructure
- Go to the root folder of the repo
- Run the following command:

```
ansible-playbook main.yaml --extra-vars "command=start
kops_state_store=$S3_bucket_name cluster_name=$clusten_name dns_zone_id=
$DNS_zone_id ssh_path=$path_to_ssh_file profile=$aws_profile
fluentd_accessid=$fluent_user_access_id
fluentd_accesskey=$fluentd_access_key”
```

- On successful execution of the ansible playbook, check all the resources created in the AWS console
- Resources created:
    - EC2 instances (2 nodes, 1 master, 1 bastion)
    - RDS instance
    - Kubernetes VPC, IGW and subnets
    - Security groups
    - CloudWatch Log group - kubernetes


Activities
Terminal
Apr 21 19:41
mitalisalvi@fedora:~/Documents/FinalProject/infrastructure
mitalisalvi@fedora:~/Docume...
mitalisalvi@fedora:~/Docume...
mitalisalvi@fedora:~/Docume...
mitalisalvi@fedora:~/Docume...
mitalisalvi@fedora:~/Docume...

```
TASK [grafana : Create ConfigMap for Grafana to use Prometheus as data source] **********************************************
changed: [localhost]

TASK [grafana : Create ConfigMap for Grafana to import dashboard] ***********************************************************
changed: [localhost]

TASK [grafana : Install grafana chart] **************************************************************************************
changed: [localhost]

TASK [logforward : Create a fluentd namespace] ******************************************************************************
changed: [localhost]

TASK [logforward : Install fluentd chart] ***********************************************************************************
changed: [localhost]

TASK [helper : Get Grafana Password] ****************************************************************************************
changed: [localhost]

TASK [jenkins : Create a jenkins namespace] *********************************************************************************
changed: [localhost]

TASK [jenkins : Install jenkins] ********************************************************************************************
changed: [localhost]

TASK [jenkins : Create Jenkins ServiceAccount.] *****************************************************************************
changed: [localhost]

TASK [metric-server : Install K8s metric server helm chart] *****************************************************************
changed: [localhost]

TASK [cluster-autoscaling : Replace cluster name in instance group yaml] ****************************************************
ok: [localhost]

TASK [cluster-autoscaling : Download cluster-autoscaler.yml] ****************************************************************
ok: [localhost]

TASK [cluster-autoscaling : Create Auto scaling Configuration] **************************************************************
changed: [localhost]

PLAY RECAP ******************************************************************************************************************
localhost                  : ok=40   changed=29   unreachable=0    failed=0    skipped=13   rescued=0    ignored=0

[mitalisalvi@fedora infrastructure]$ 
```

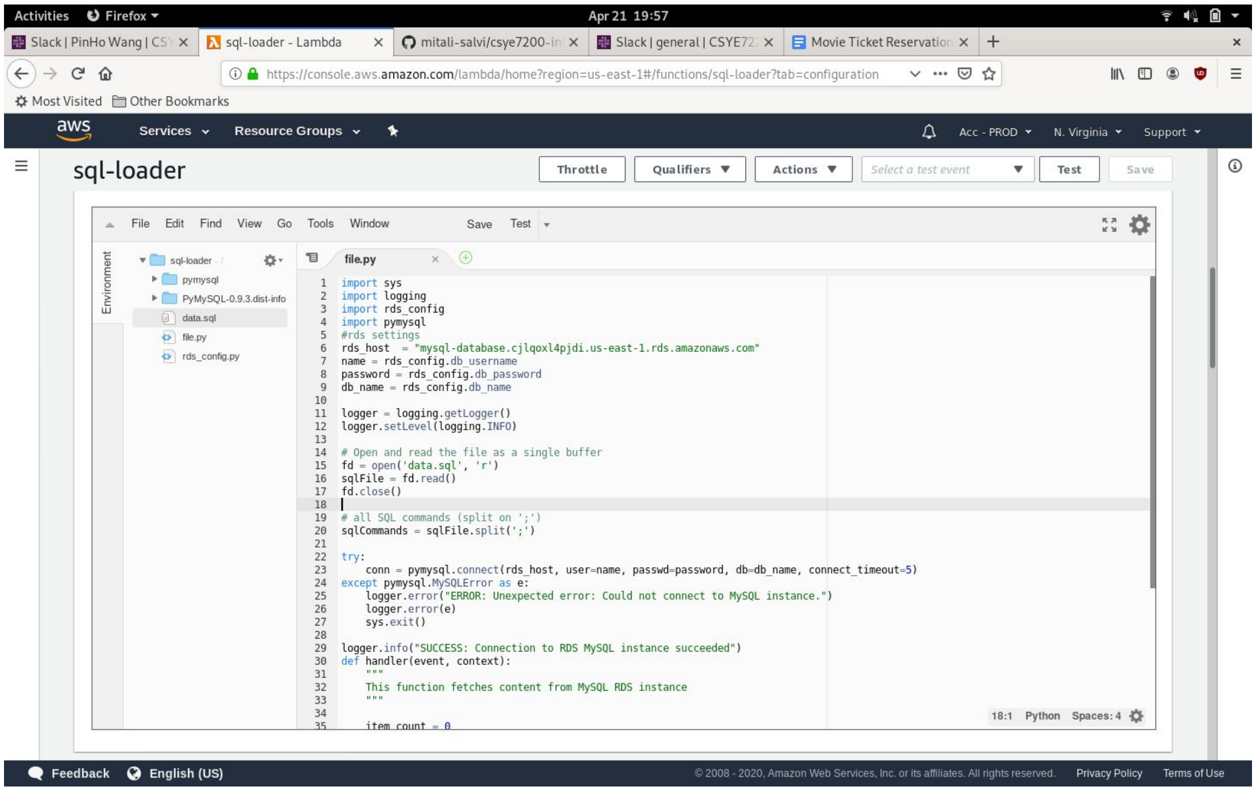
2. Create Lambda Function in AWS

- Zip the file and its dependent libraries in a folder
- Create a role in AWS with full access to AWS Lambda
- Create a security group for which allows traffic from the default VPC
- Execute the following command:

```
AWS_PROFILE=prod aws lambda create-function --function-name  sql-loader --runtime
python3.8 \
--zip-file fileb://function.zip --handler file.handler \
--role $role_ARN  \
--vpc-config
SubnetIds=$subnets_in_default_vpc,SecurityGroupIds=$security_group_id
```

- Update the created lambda function by uploading the zip folder previously created

```
AWS_PROFILE=prod aws lambda update-function-code --function-name sql-loader
--zip-file fileb://function.zip
```



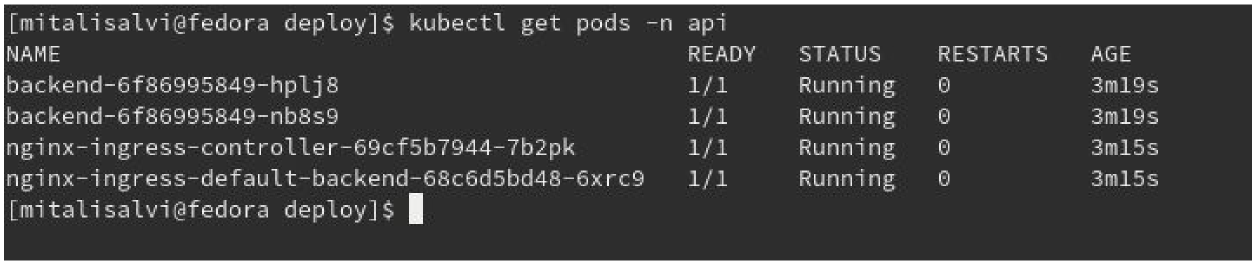
```python
import sys
import logging
import rds_config
import pymysql
#rds settings
rds_host  = "mysql-database.cjlqoxl4pjdi.us-east-1.rds.amazonaws.com"
name = rds_config.db_username
password = rds_config.db_password
db_name = rds_config.db_name

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Open and read the file as a single buffer
fd = open('data.sql', 'r')
sqlFile = fd.read()
fd.close()

# all SQL commands (split on ';')
sqlCommands = sqlFile.split(';')

try:
    conn = pymysql.connect(rds_host, user=name, passwd=password, db=db_name, connect_timeout=5)
except pymysql.MySQLError as e:
    logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
    logger.error(e)
    sys.exit()

logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")
def handler(event, context):
    """
    This function fetches content from MySQL RDS instance
    """

    item_count = 0
```

3. Deploy Backend Application

The backend architecture is divided into 2 microservices - backend and security backend

- Take the latest pull from the following repos:
    - Backend: https://github.com/mitali-salvi/ticket-reservation
    - Security Backend: https://github.com/mitali-salvi/admin-ticket-reservation
- To deploy the backend microservice
    - Go to deploy folder of the repo
    - Execute the deploy-backend shell script. Input the parameters asked for in the prompt

This script will create a deployment, a service account and expose the deployment via a Node Port.
Then the script will create a nginx-ingress controller to handle both the backend microservices

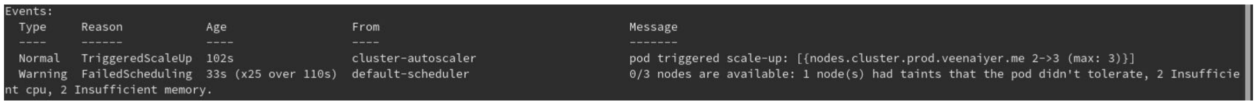


```
[mitalisalvi@fedora deploy]$ kubectl get pods -n api
NAME                                            READY   STATUS    RESTARTS   AGE
backend-6f86995849-hplj8                        1/1     Running   0          3m19s
backend-6f86995849-nb8s9                        1/1     Running   0          3m19s
nginx-ingress-controller-69cf5b7944-7b2pk       1/1     Running   0          3m15s
nginx-ingress-default-backend-68c6d5bd48-6xrc9  1/1     Running   0          3m15s
[mitalisalvi@fedora deploy]$ 
```

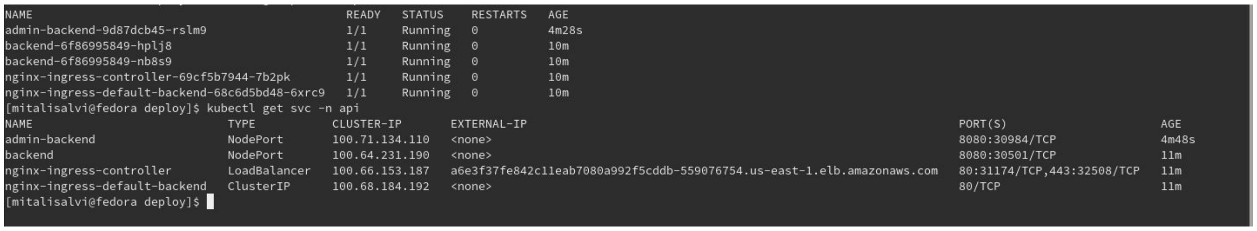
- To deploy the security microservice
- Go to deploy folder of the repo
- Execute the deploy-admin-backend shell script

This script will create a deployment, a service account and expose the deployment via a Node Port
Also this script will trigger the AWS lambda function, since all the backend services are not deployed

(PS: The state of the pods after this may be pending. This is due to the auto scaling functionality kicking into the picture which will spin up a new EC2 instance and deploy the app in that node)



```
Events:
  Type     Reason             Age                  From                                    Message
  ----     ------             ----                 ----                                    -------
  Normal   TriggeredScaleUp  102s                  cluster-autoscaler                      pod triggered scale-up: [{nodes.cluster.prod.veenaiyer.me 2->3 (max: 3)}]
  Warning  FailedScheduling  33s (x25 over 110s)  default-scheduler                       0/3 nodes are available: 1 node(s) had taints that the pod didn't tolerate, 2 Insufficie
nt cpu, 2 Insufficient memory.
```

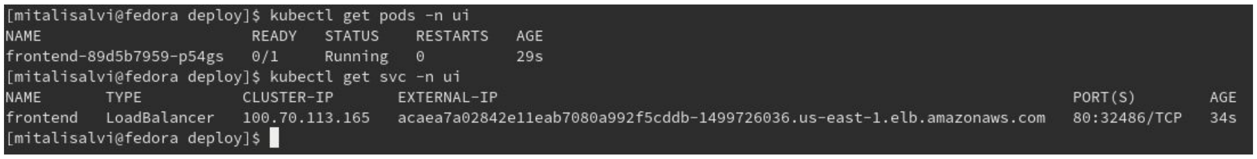


```
NAME                                            READY   STATUS    RESTARTS   AGE
admin-backend-9d87dcb45-rslm9                   1/1     Running   0          4m28s
backend-6f86995849-hplj8                        1/1     Running   0          10m
backend-6f86995849-nb8s9                        1/1     Running   0          10m
nginx-ingress-controller-69cf5b7944-7b2pk       1/1     Running   0          10m
nginx-ingress-default-backend-68c6d5bd48-6xrc9  1/1     Running   0          10m
[mitalisalvi@fedora deploy]$ kubectl get svc -n api
NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)                      AGE
admin-backend                   NodePort       100.71.134.110   <none>                                                                    8080:30984/TCP               4m48s
backend                         NodePort       100.64.231.190   <none>                                                                    8080:30501/TCP               11m
nginx-ingress-controller        LoadBalancer   100.66.153.187   a6e3f37fe842c11eab7080a992f5cddb-559076754.us-east-1.elb.amazonaws.com   80:31174/TCP,443:32508/TCP   11m
nginx-ingress-default-backend   ClusterIP      100.68.184.192   <none>                                                                    80/TCP                       11m
[mitalisalvi@fedora deploy]$
```

4. Deploy Frontend Application

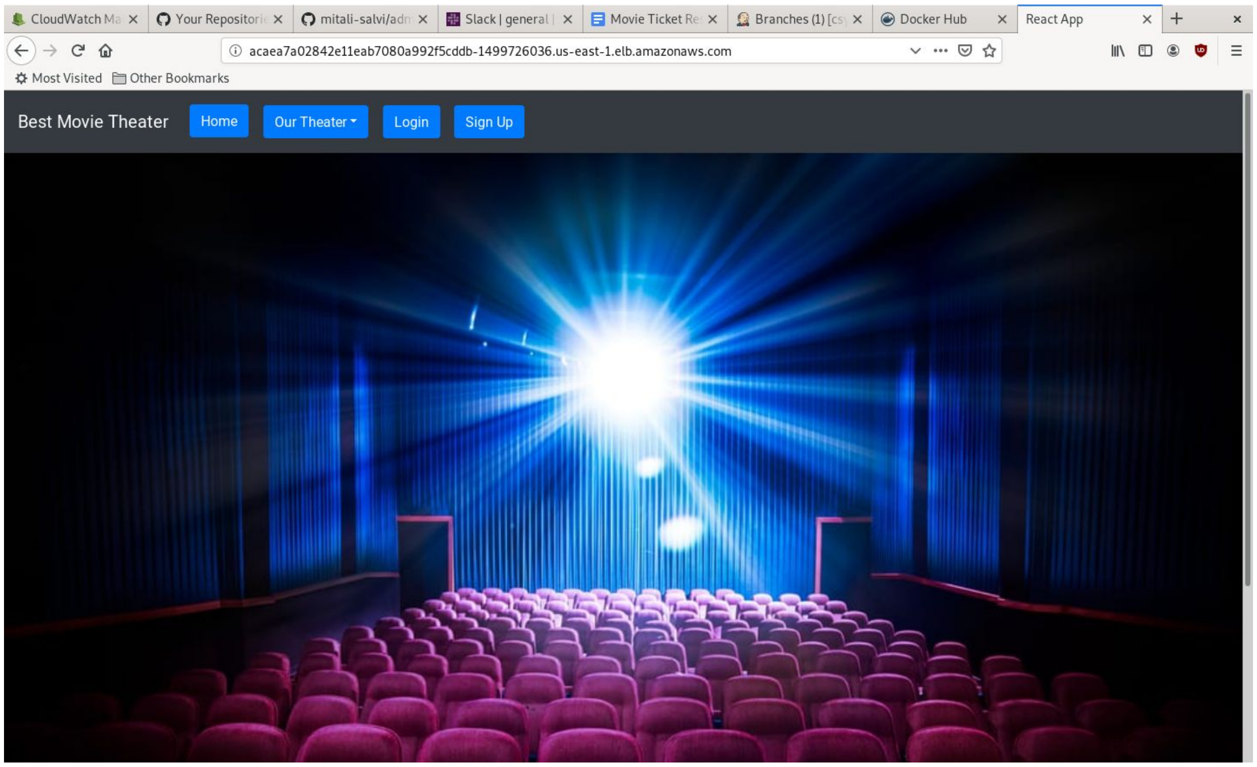
- Take the latest pull from the repo: https://github.com/mitali-salvi/movie-frontend
- Go to the deploy folder in the root of the repo
- Execute the deploy-frontend shell script

This script will create a deployment, a service account and expose the deployment via a Load Balancer to the outside world



```
[mitalisalvi@fedora deploy]$ kubectl get pods -n ui
NAME                        READY   STATUS    RESTARTS   AGE
frontend-89d5b7959-p54gs    0/1     Running   0          29s
[mitalisalvi@fedora deploy]$ kubectl get svc -n ui
NAME       TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
frontend   LoadBalancer   100.70.113.165   acaea7a02842e11eab7080a992f5cddb-1499726036.us-east-1.elb.amazonaws.com   80:32486/TCP   34s
[mitalisalvi@fedora deploy]$
```

The app is now deployed and live !!!!!



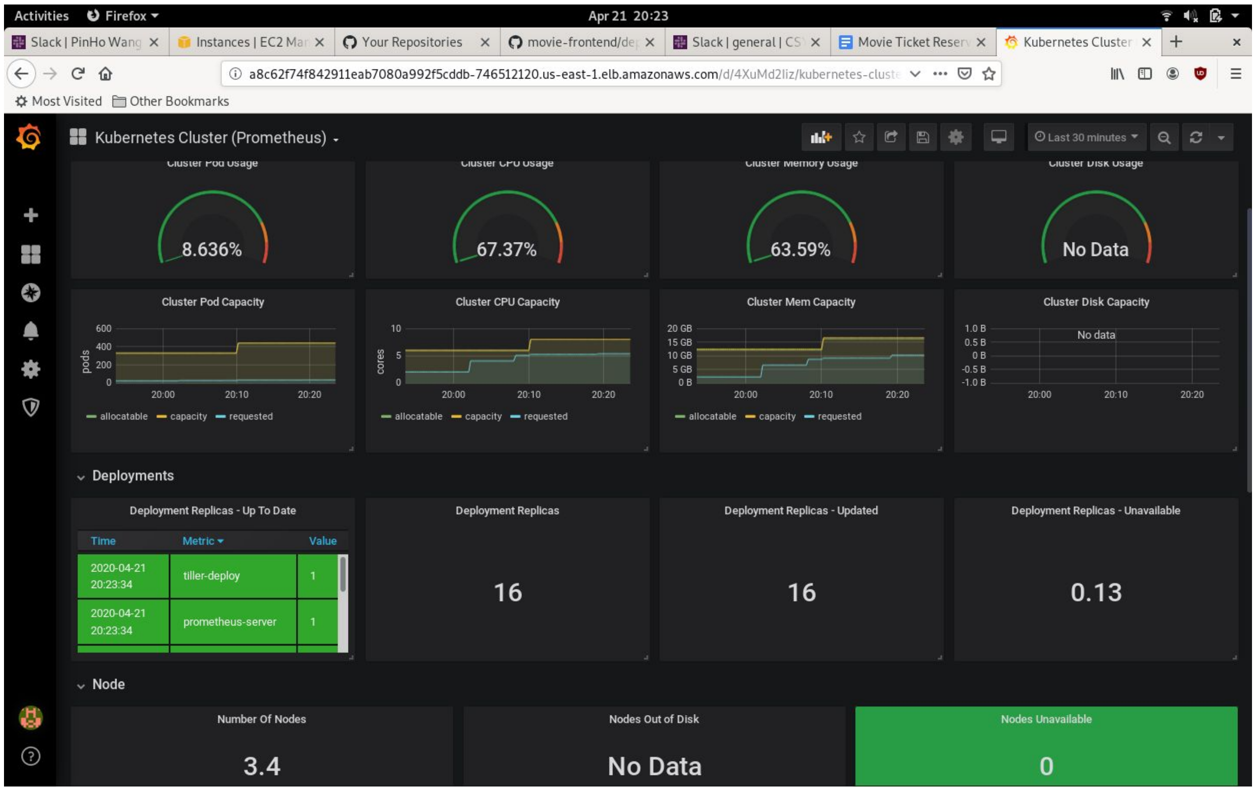
1. Go to the public URL of the frontend service
2. Sign up for the portal using email address and password
3. Login into portal using the above credentials
4. Update your profile with first and last name
5. Add a payment method to book tickets
6. Select the theatre from the dropdown of your choice
7. Select the movies which are playing in that theatre
8. Select the date, time, hall and seat of your choice
9. Click on confirm to book the tickets
10. Go to profile to see your past booked tickets

5. Monitoring Tools

- Grafana
- Get the public IP of the Grafana service by running the following command
- Also get the default password created by Helm to login to the dashboard

```
kubectl get svc -n grafana
kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}"
| base64 --decode ; echo
```

- Go to the public URL of the load balance and login using admin and the credentials

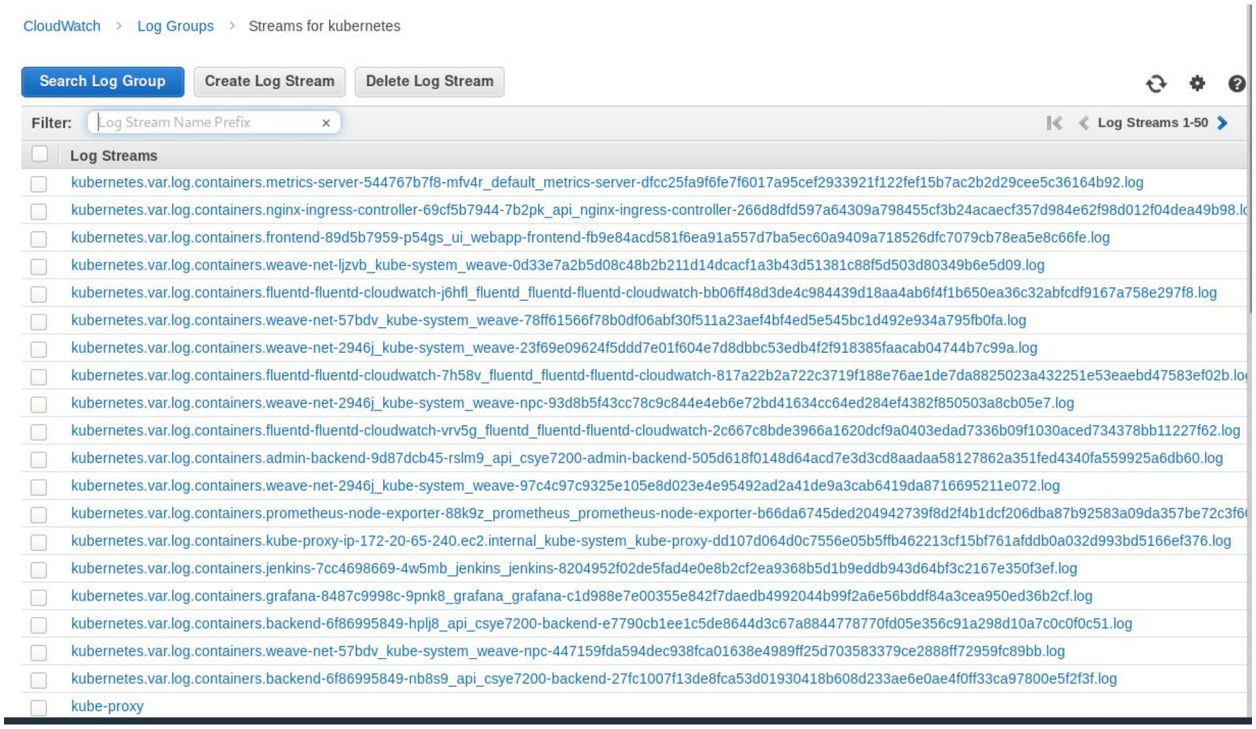


Time	Metric	Value
2020-04-21 20:23:34	tiller-deploy	1
2020-04-21 20:23:34	prometheus-server	1

Also checkout the other dashboards to monitor the backend and frontend architecture

- fluentd

- Login to the AWS console and go to AWS Cloud Watch to monitor the logs of the cluster. This includes all the helm charts installed and the microservices deployed in the cluster



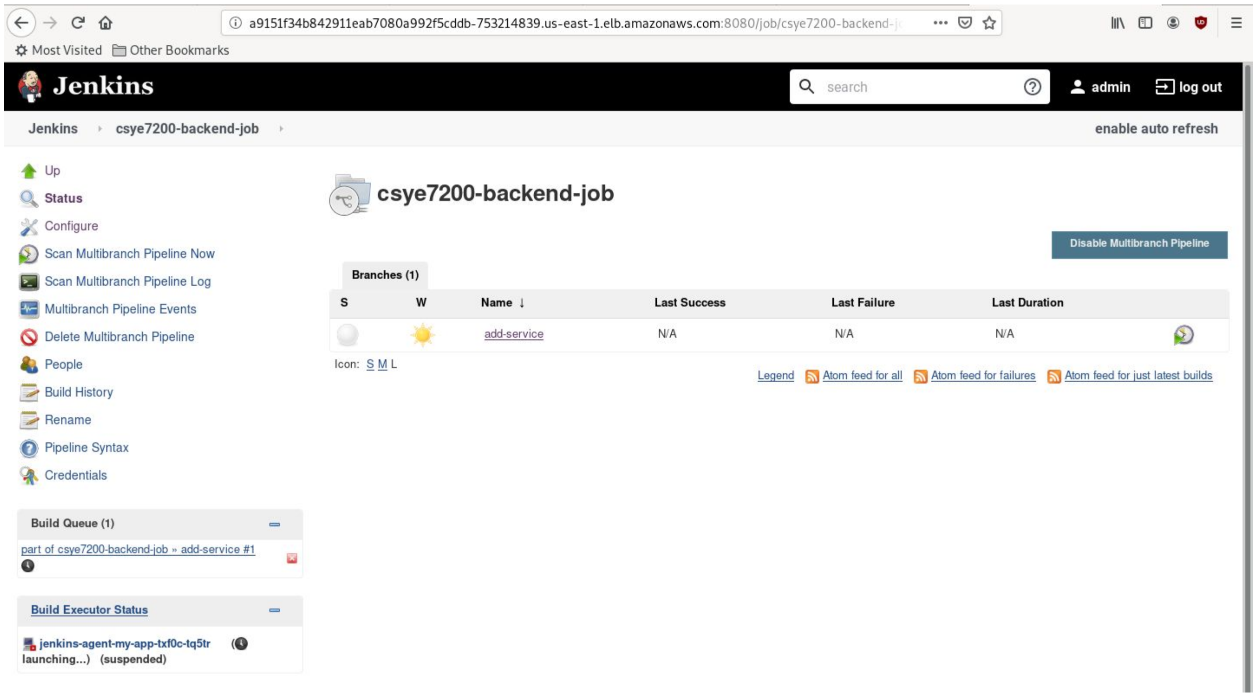
Log Streams
kubernetes.var.log.containers.metrics-server-544767b7f8-mfv4r_default_metrics-server-dfcc25fa9f6fe7f6017a95cef2933921f122fef15b7ac2b2d29cee5c36164b92.log
kubernetes.var.log.containers.nginx-ingress-controller-69cf5b7944-7b2pk_api_nginx-ingress-controller-266d8dfd597a64309a798455cf3b24acaecf357d984e62f98d012f04dea49b98.lo
kubernetes.var.log.containers.frontend-89d5b7959-p54gs_ui_webapp-frontend-fb9e84acd581f6ea91a557d7ba5ec60a9409a718526dfc7079cb78ea5e8c66fe.log
kubernetes.var.log.containers.weave-net-ljzvb_kube-system_weave-0d33e7a2b5d08c48b2b211d14dcacf1a3b43d51381c88f5d503d80349b6e5d09.log
kubernetes.var.log.containers.fluentd-fluentd-cloudwatch-j6hfl_fluentd_fluentd-fluentd-cloudwatch-bb06ff48d3de4c984439d18aa4ab6f4f1b650ea36c32abfcdf9167a758e297f8.log
kubernetes.var.log.containers.weave-net-57bdv_kube-system_weave-78ff61566f78b0df06abf30f511a23aef4bf4ed5e545bc1d492e934a795fb0fa.log
kubernetes.var.log.containers.weave-net-2946j_kube-system_weave-23f69e09624f5ddd7e01f604e7d8dbbc53edb4f2f918385faacab04744b7c99a.log
kubernetes.var.log.containers.fluentd-fluentd-cloudwatch-7h58v_fluentd_fluentd-fluentd-cloudwatch-817a22b2a722c3719f188e76ae1de7da8825023a432251e53eaebd47583ef02b.lo
kubernetes.var.log.containers.weave-net-2946j_kube-system_weave-npc-93d8b5f43cc78c9c844e4eb6e72bd41634cc64ed284ef4382f850503a8cb05e7.log
kubernetes.var.log.containers.fluentd-fluentd-cloudwatch-vrv5g_fluentd_fluentd-fluentd-cloudwatch-2c667c8bde3966a1620dcf9a0403edad7336b09f1030aced734378bb11227f62.log
kubernetes.var.log.containers.admin-backend-9d87dcb45-rslm9_api_csye7200-admin-backend-505d618f0148d64acd7e3d3cd8aadaa58127862a351fed4340fa559925a6db60.log
kubernetes.var.log.containers.weave-net-2946j_kube-system_weave-97c4c97c9325e105e8d023e4e95492ad2a41de9a3cab6419da8716695211e072.log
kubernetes.var.log.containers.prometheus-node-exporter-88k9z_prometheus_prometheus-node-exporter-b66da6745ded204942739f8d2f4b1dcf206dba87b92583a09da357be72c3f6
kubernetes.var.log.containers.kube-proxy-ip-172-20-65-240.ec2.internal_kube-system_kube-proxy-dd107d064d0c7556e05b5ffb462213cf15bf761afddb0a032d993bd5166ef376.log
kubernetes.var.log.containers.jenkins-7cc4698669-4w5mb_jenkins_jenkins-8204952f02de5fad4e0e8b2cf2ea9368b5d1b9eddb943d64bf3c2167e350f3ef.log
kubernetes.var.log.containers.grafana-8487c9998c-9pnk8_grafana_grafana-c1d988e7e00355e842f7daedb4992044b99f2a6e56bddf84a3cea950ed36b2cf.log
kubernetes.var.log.containers.backend-6f86995849-hplj8_api_csye7200-backend-e7790cb1ee1c5de8644d3c67a8844778770fd05e356c91a298d10a7c0c0f0c51.log
kubernetes.var.log.containers.weave-net-57bdv_kube-system_weave-npc-447159fda594dec938fca01638e4989ff25d703583379ce2888ff72959fc89bb.log
kubernetes.var.log.containers.backend-6f86995849-nb8s9_api_csye7200-backend-27fc1007f13de8fca53d01930418b608d233ae6e0ae4f0ff33ca97800e5f2f3f.log
kube-proxy

- CI-CD tool

Login to Jenkins by getting the URL and the default password by the following command:

```
kubectl get svc -n jenkins
kubectl get secret --namespace jenkins jenkins -o
jsonpath="{.data.jenkins-admin-password}" | base64 --decode
```

On logging on to the portal, set up the configuration and then run the pipeline.



a9151f34b842911eab7080a992f5cddb-753214839.us-east-1.elb.amazonaws.com:8080/job/csye7200-backend-j

Most Visited Other Bookmarks

Jenkins search admin log out

Jenkins › csye7200-backend-job › enable auto refresh

- Up
- Status
- Configure
- Scan Multibranch Pipeline Now
- Scan Multibranch Pipeline Log
- Multibranch Pipeline Events
- Delete Multibranch Pipeline
- People
- Build History
- Rename
- Pipeline Syntax
- Credentials

Build Queue (1)

part of csye7200-backend-job » add-service #1

Build Executor Status

jenkins-agent-my-app-txf0c-tq5tr (launching...) (suspended)

csye7200-backend-job

Disable Multibranch Pipeline

Branches (1)

S	W	Name ↓	Last Success	Last Failure	Last Duration
○	☀	add-service	N/A	N/A	N/A

Icon: S M L

Legend Atom feed for all Atom feed for failures Atom feed for just latest builds