Distributed data stores on Kubernetes * and other things

Alena Hall

@lenadroid

Prerequisites

- Kubernetes cluster
- Basic understanding of Cassandra

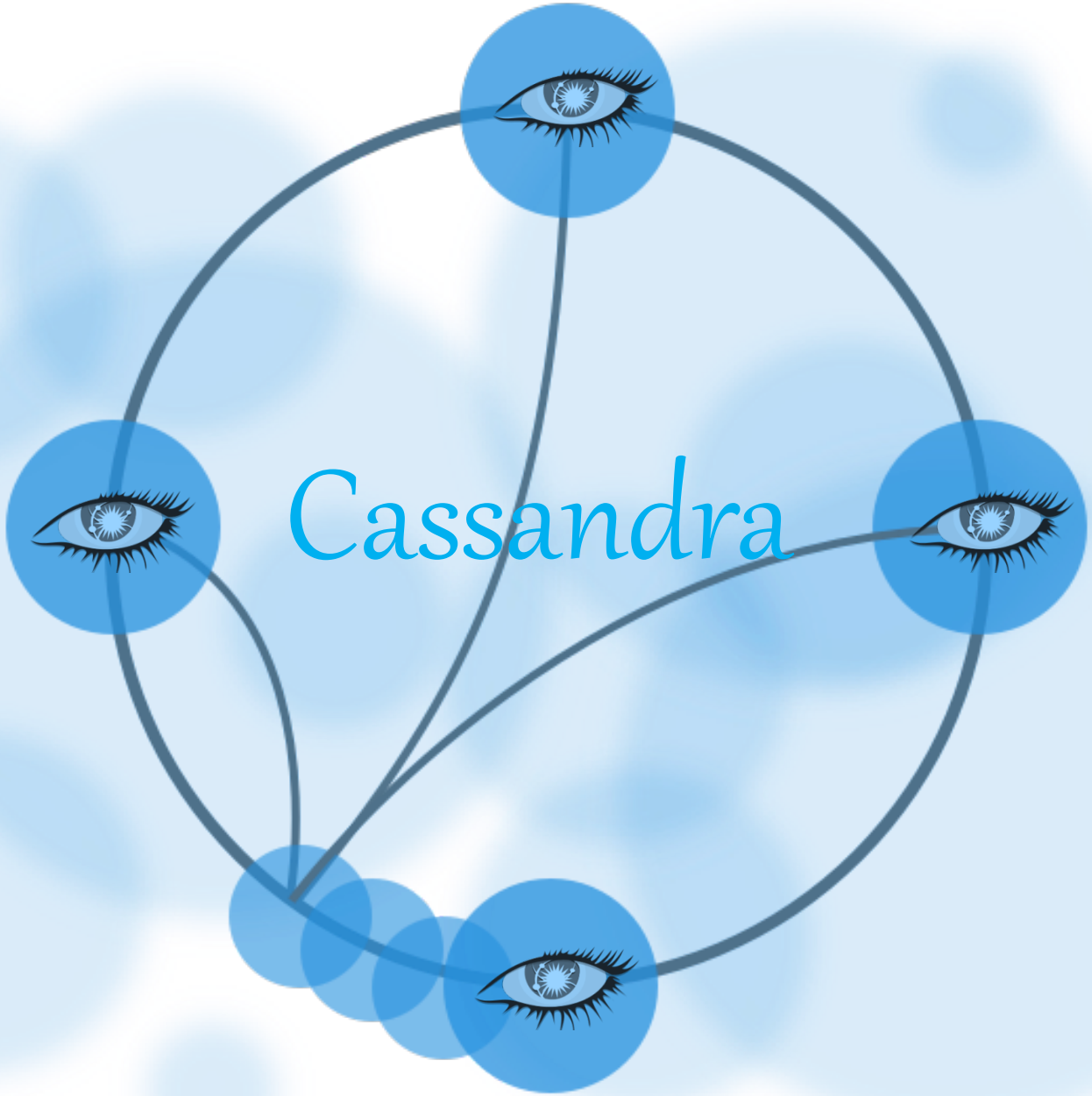
Prerequisites

- Kubernetes cluster
- Basic understanding of Cassandra
- Courage

Motivation

Talk plan

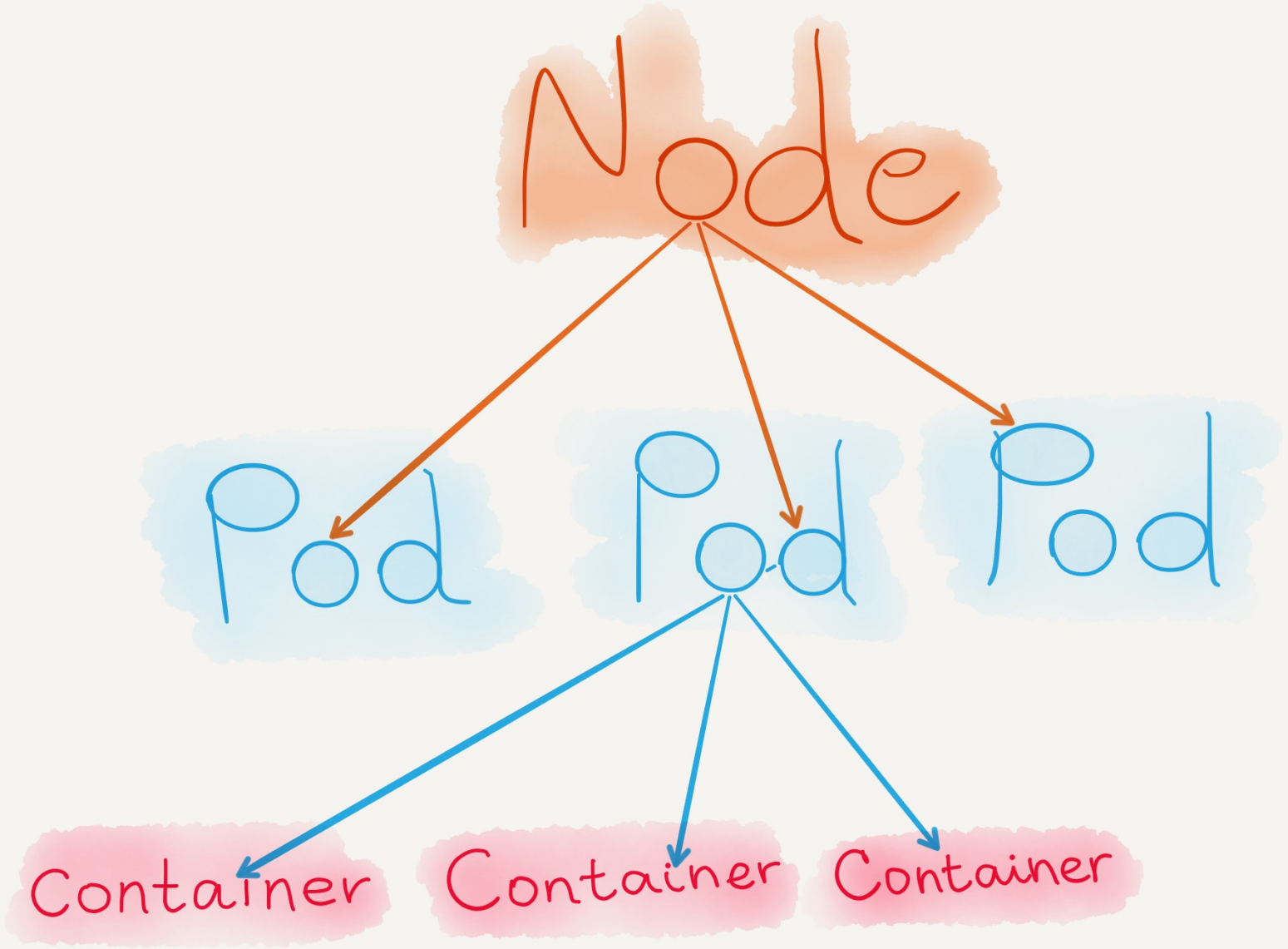
- ✓ Kubernetes and Cassandra refresher
- ✓ Stateful Sets, PVs, PVCs, Storage Classes, and more
- ✓ Example: Cassandra Stateful Set
- ✓ Kubernetes Jobs
- ✓ Example: Writing data into Cassandra using Jobs
- ✓ Spark jobs on Kubernetes
- ✓ Example: Spark and Cassandra on Kubernetes

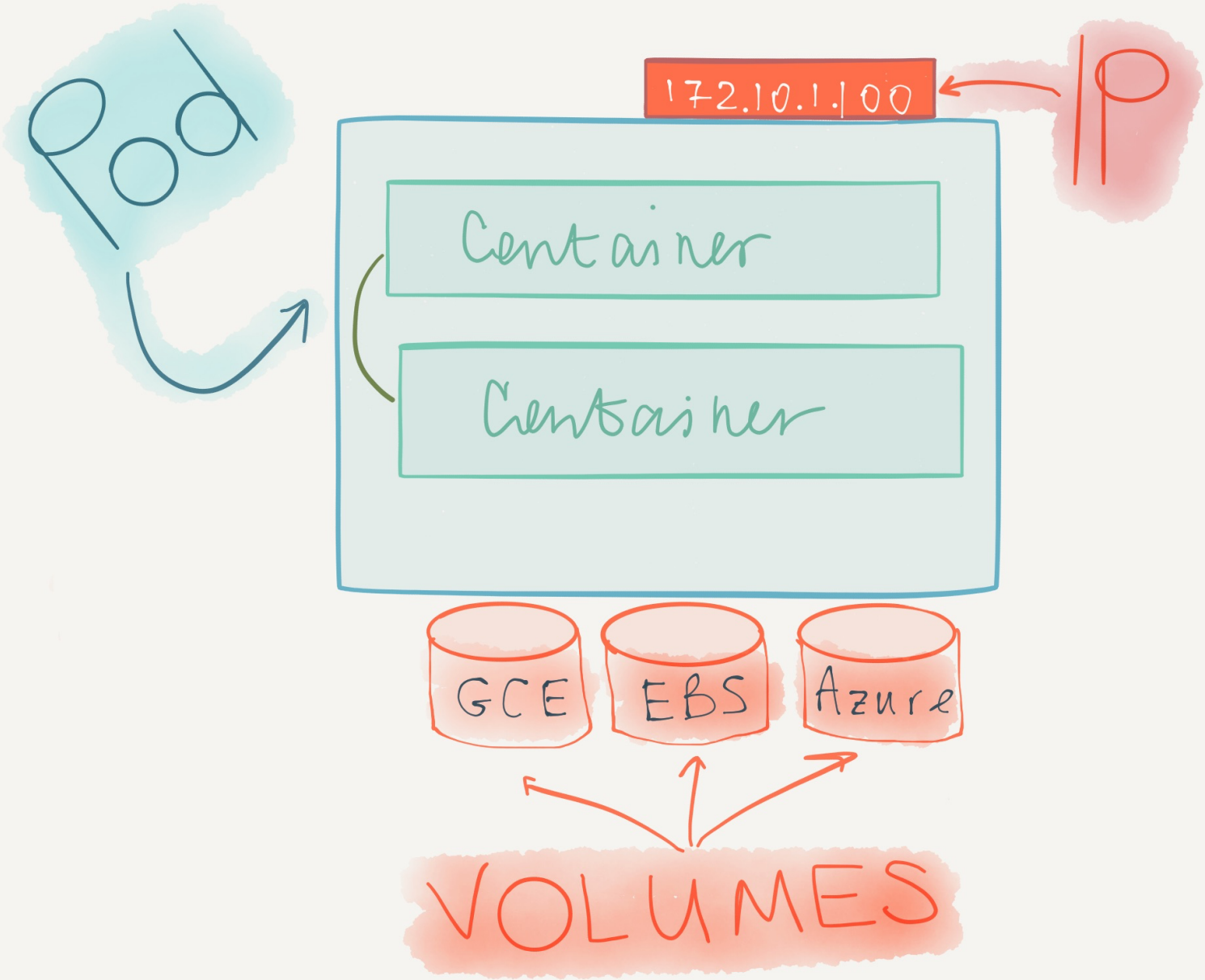

Cassandra

- Distributed database, has a ring topology, uses consistent hashing
- Highly available system with tunable consistency and hinted hand off
- Data is grouped by the partition key and stored together physically
- Stores data in SSTable on disk and RAM and tracks operations in the commit log
- Data can be ordered by a clustering key
- Supports equality, inequality, aggregation queries and custom functions
- Hash ranges can be mapped to Simple Nodes or to Vnodes

Kubernetes Refresher

@lenadroid




Pod
172.10.1.100
IP
Container
Container
GCE
EBS
Azure
VOLUMES

Art by @ashleymcnamara

@lenadroid

Kubernetes Deployments/Replica Sets

Keeps specified number of pod replicas running at any given time

Ephemeral containers

@lenadroid

Distributed databases - systems that require stable persistence

Kubernetes Volumes

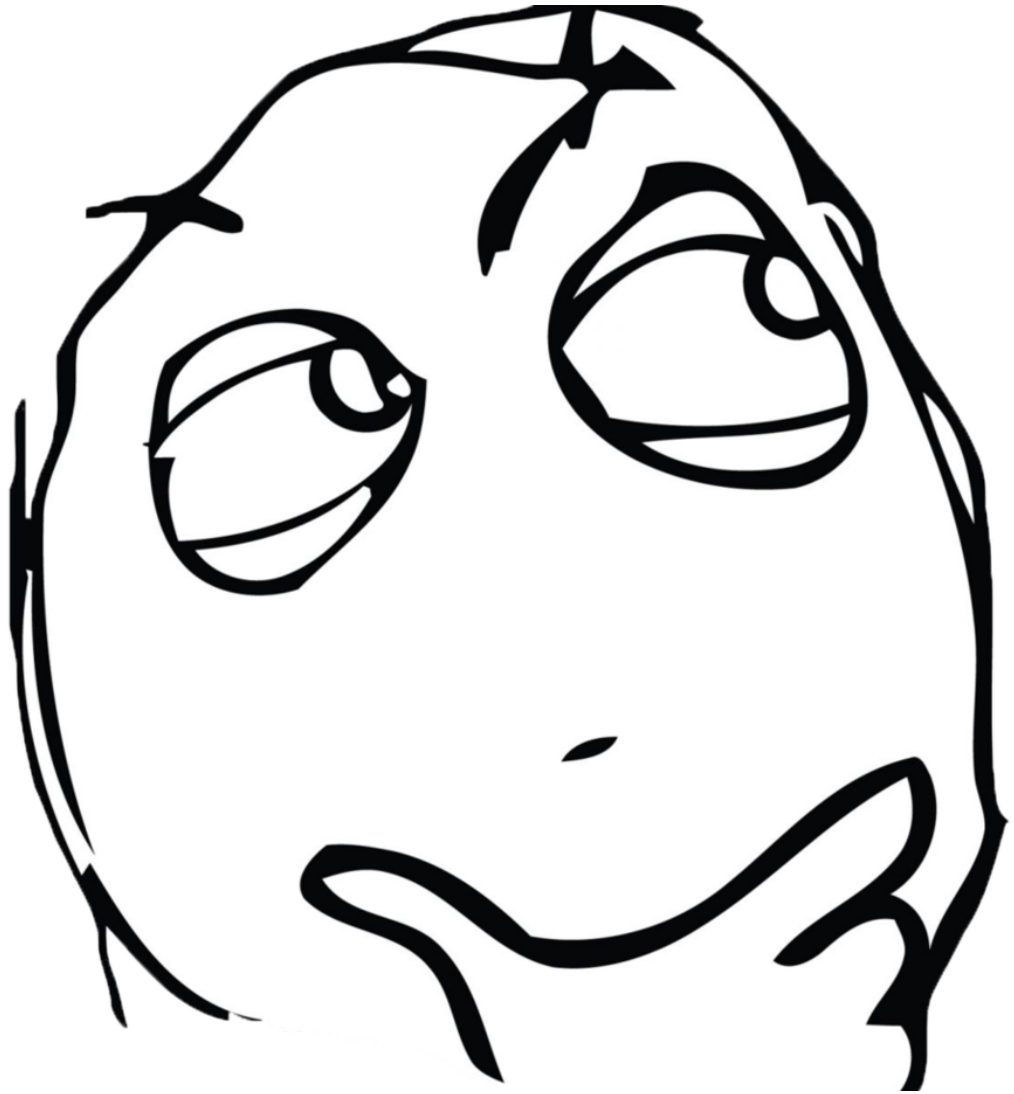
What if we try...
Replicated containers with volumes?

Standard containers are not persistent

Even combined with Kubernetes volumes, there are issues:

- Non-deterministic, **random** names
- Random order in which pods start, scale and terminate


So, replica sets with volumes won't work



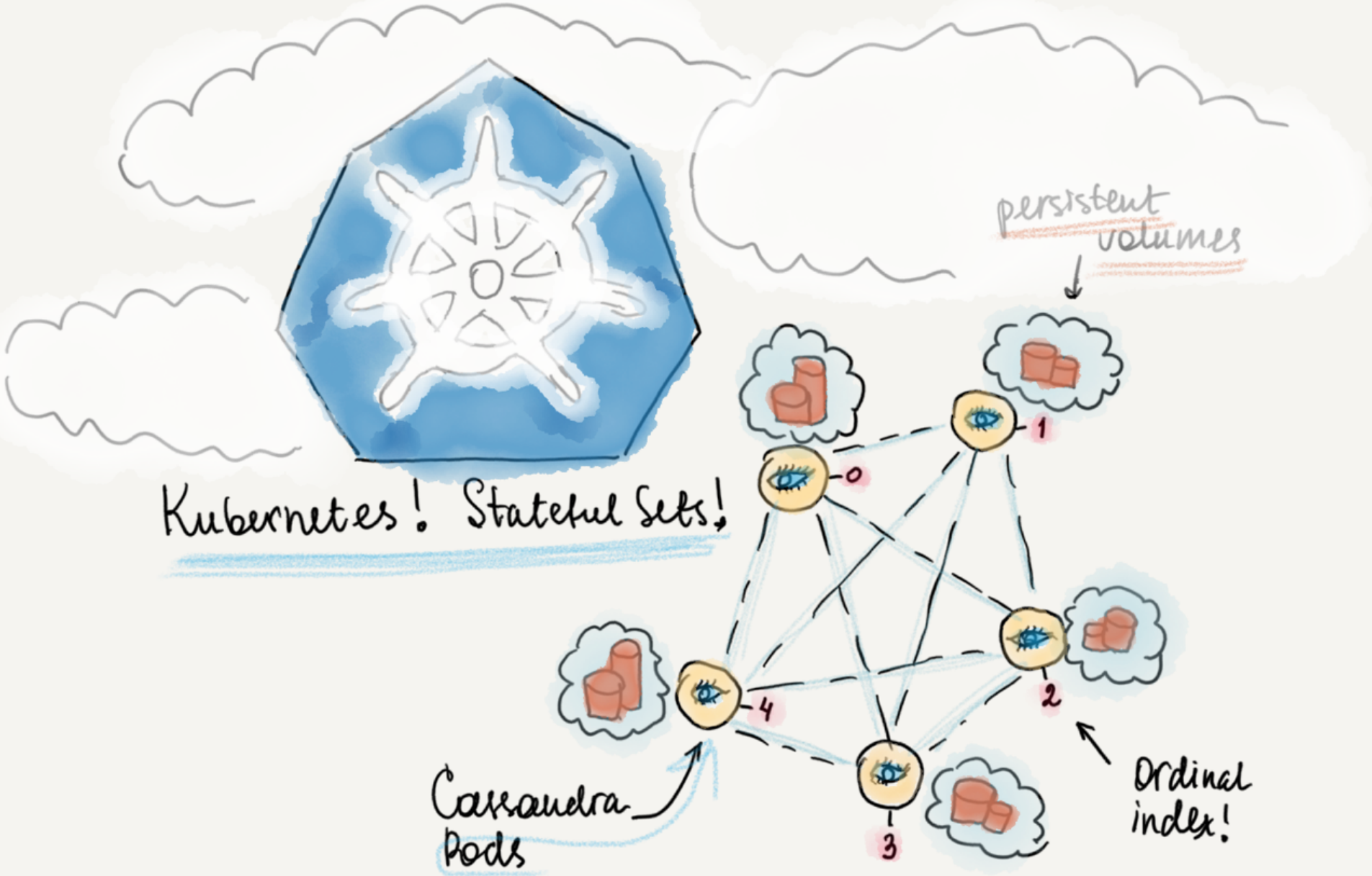
Prior to creating the database cluster :

- We need to have certain **guarantees** about cluster nodes
- Database nodes should have **discoverable names**
- Nodes need to start in **predictable order**
- Stable persistent storage

Stateful Sets

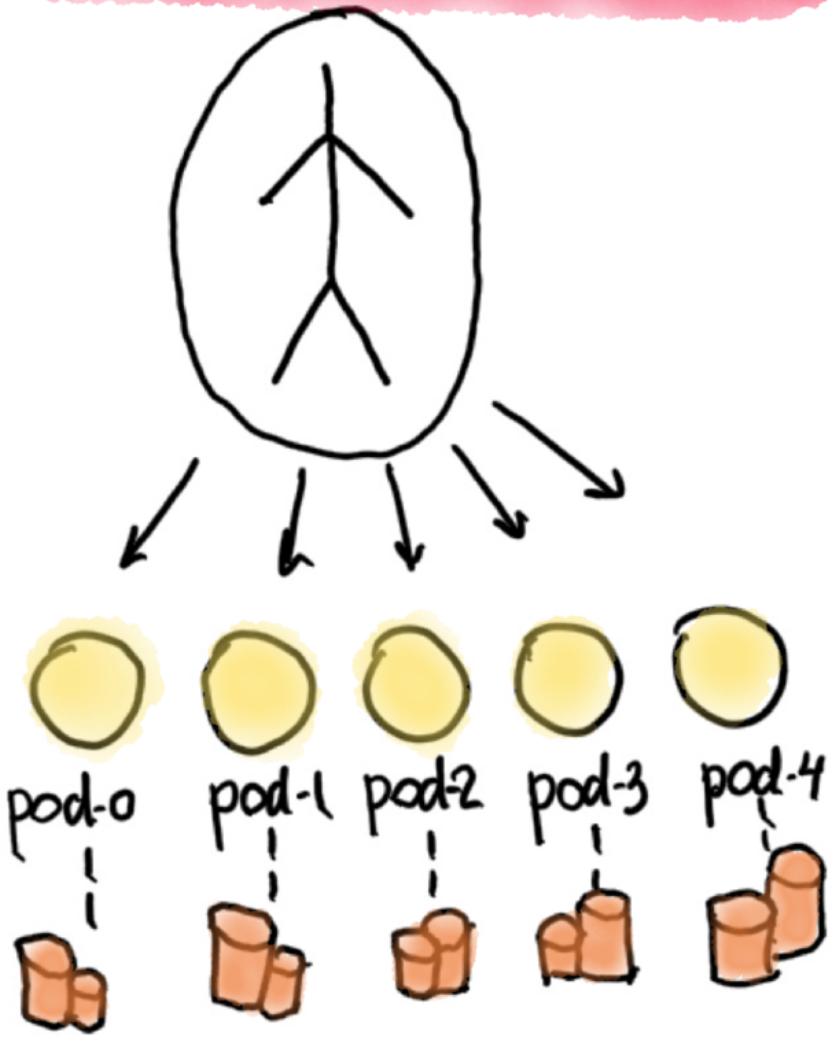
Stateful Set pods have identity ...

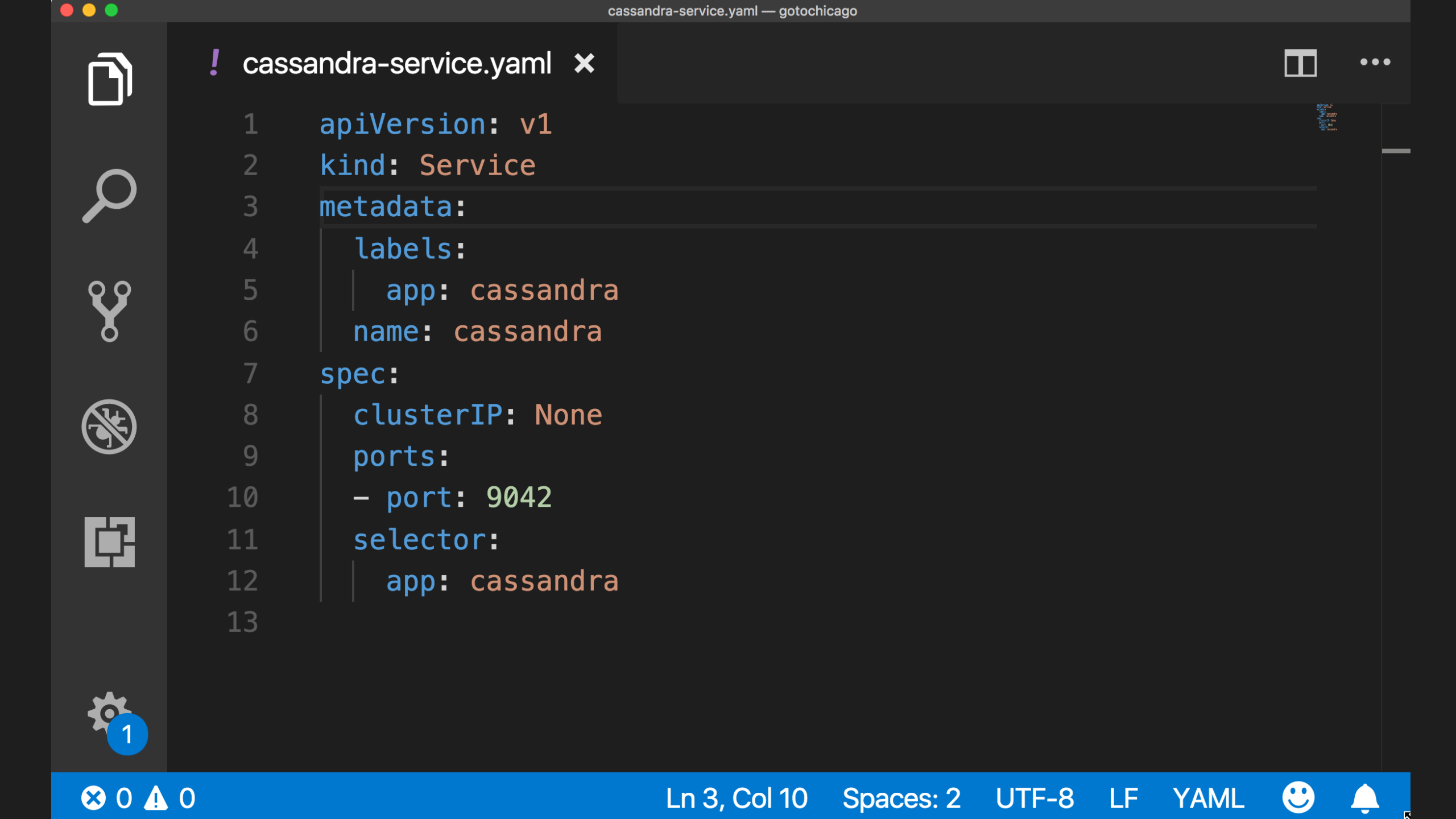
- Stable, **unique** network identifiers
- **Stable**, persistent storage, and link from pod to storage
- **Ordered** deployment, scaling, termination, rolling updates


persistent volumes
Kubernetes! Stateful Sets!
0
1
2
3
4
Cassandra Pods
Ordinal index!

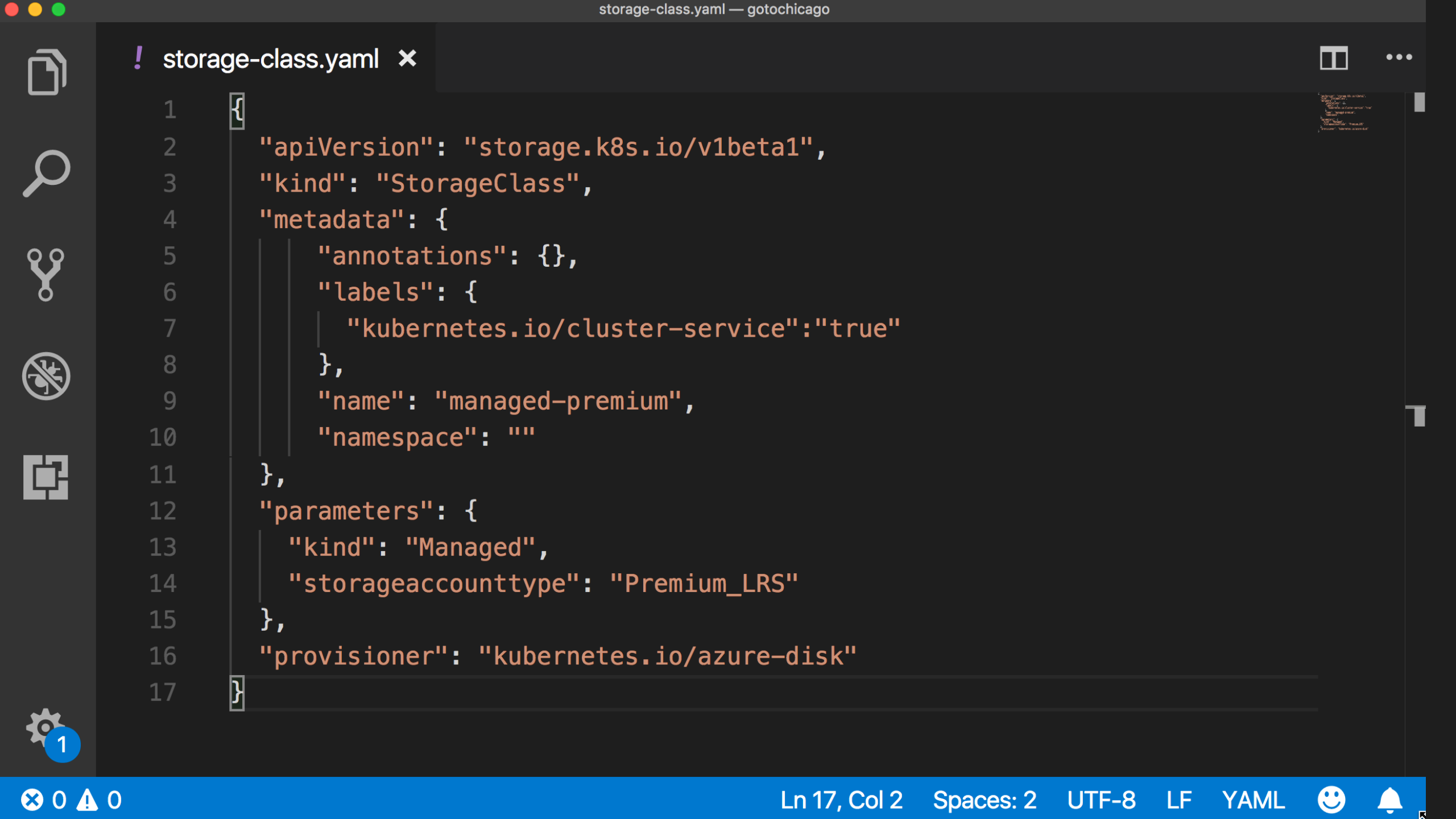
Headless Service

Headless Service




cassandra-service.yaml — gotochicago
cassandra-service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4 labels:
5 app: cassandra
6 name: cassandra
7 spec:
8 clusterIP: None
9 ports:
10 - port: 9042
11 selector:
12 app: cassandra
13
1
0 0
Ln 3, Col 10
Spaces: 2
UTF-8
LF
YAML

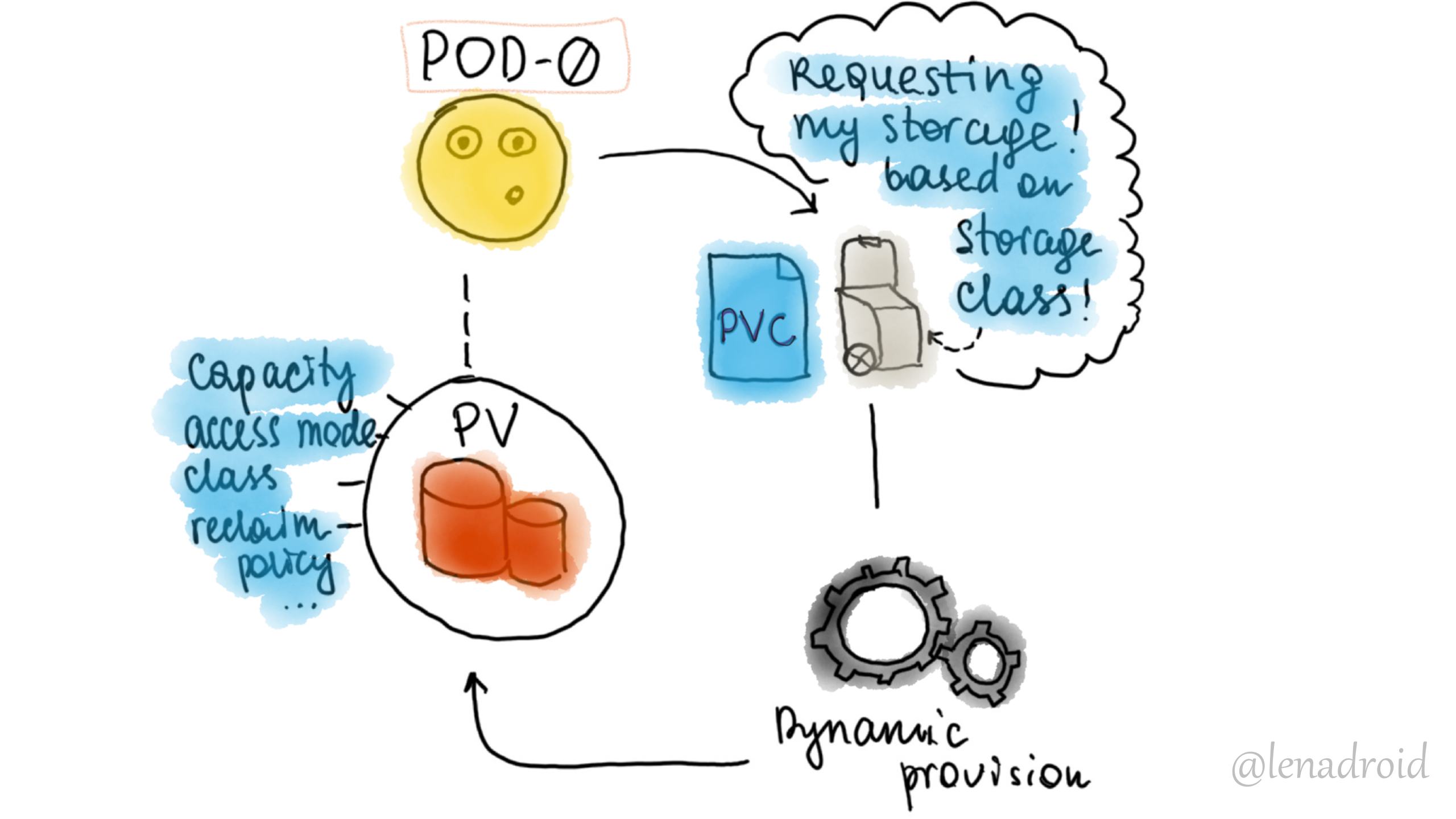
Storage Classes


storage-class.yaml — gotochicago
storage-class.yaml
{
"apiVersion": "storage.k8s.io/v1beta1",
"kind": "StorageClass",
"metadata": {
"annotations": {},
"labels": {
"kubernetes.io/cluster-service":"true"
},
"name": "managed-premium",
"namespace": ""
},
"parameters": {
"kind": "Managed",
"storageaccounttype": "Premium_LRS"
},
"provisioner": "kubernetes.io/azure-disk"
}
1
0 0
Ln 17, Col 2
Spaces: 2
UTF-8
LF
YAML

Persistent Volume Claims

Persistent Volumes


A story of persistent volumes and persistent volume claims


POD-0
Requesting my storage! based on storage class!
PVC
Capacity
access mode
class
reclaim policy
...
PV
Dynamic provision

cassandra-statefulset.yaml

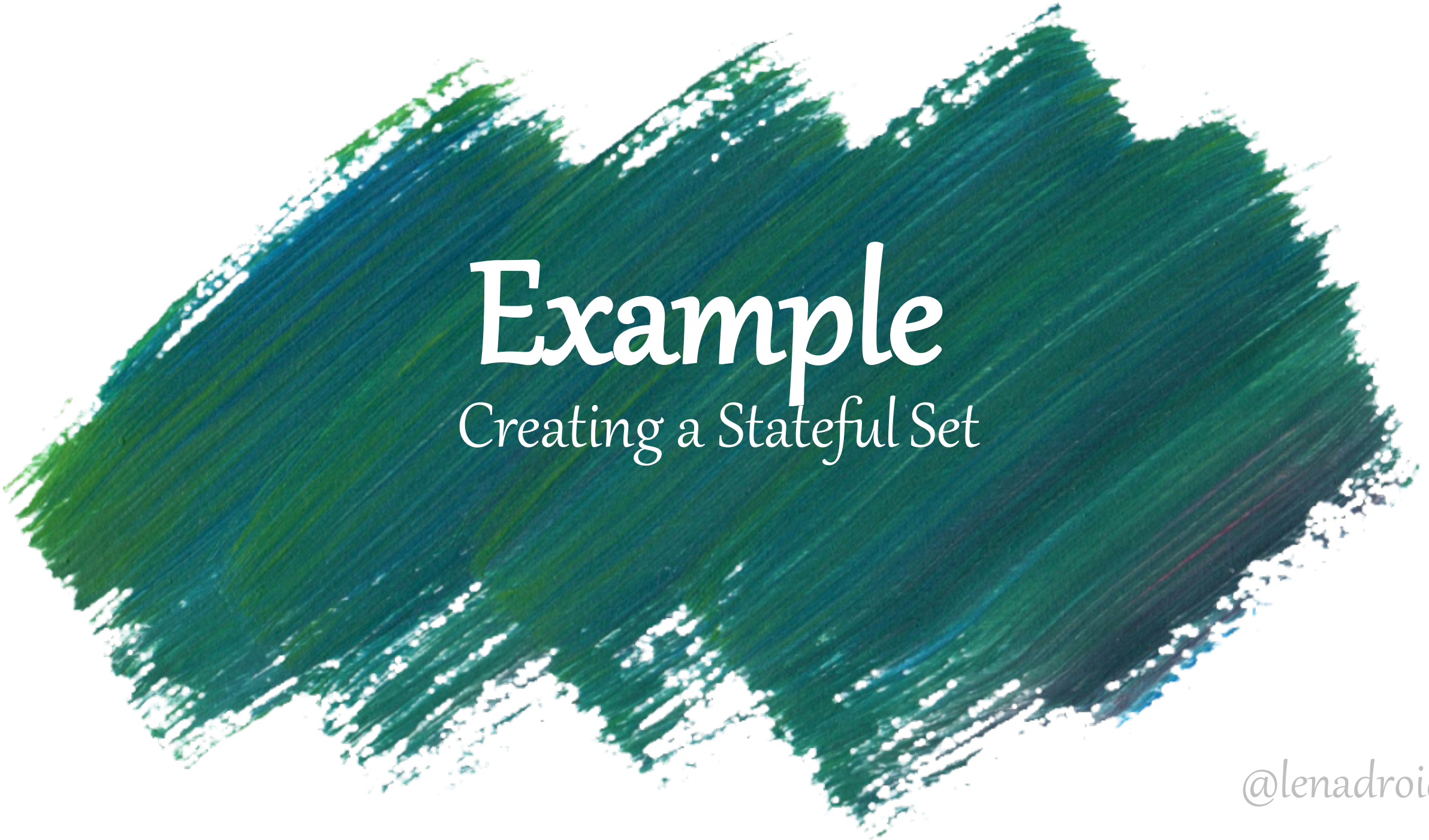
```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cassandra
  labels:
    app: cassandra
spec:
  serviceName: cassandra
  replicas: 5
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      terminationGracePeriodSeconds:
      1800
      containers:
      - name: cassandra
        image:
        gcr.io/google-samples/cassand
        ra:v13
        imagePullPolicy: Always
        ports:
        - containerPort: 7000
          name: intra-node
        - containerPort: 7001
          name: tls-intra-node
        - containerPort: 7199
          name: jmx
        - containerPort: 9042
          name: cql
        resources:
```

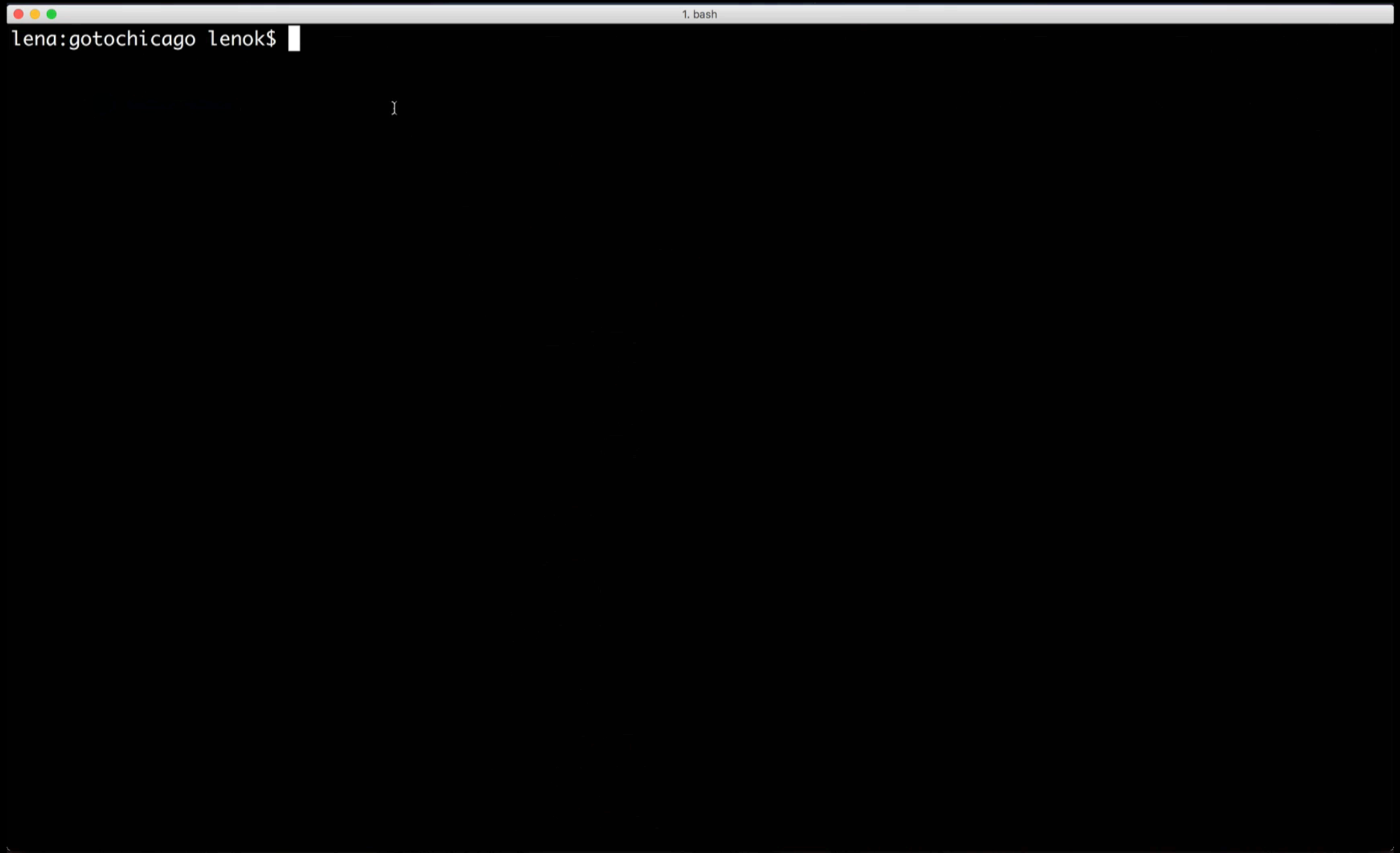
cassandra-statefulset.yaml

```yaml
        resources:
          limits:
            cpu: "2"
            memory: 3.5Gi
          requests:
            cpu: "2"
            memory: 3.5Gi
        securityContext:
          capabilities:
            add:
              - IPC_LOCK
        lifecycle:
          preStop:
            exec:
              command:
              - /bin/sh
              - -c
              - nodetool drain
        env:
          - name: MAX_HEAP_SIZE
            value: 512M
          - name: HEAP_NEWSIZE
            value: 100M
          - name: CASSANDRA_SEEDS
            value:
            "cassandra-0.cassandra.
            default.svc.cluster.
            local"
          - name:
          CASSANDRA_CLUSTER_NAME
            value: "chicago"
          - name: CASSANDRA_DC
            value: "DC1-chicago"
          - name: CASSANDRA_RACK
            value: "Rack1-chicago"
```

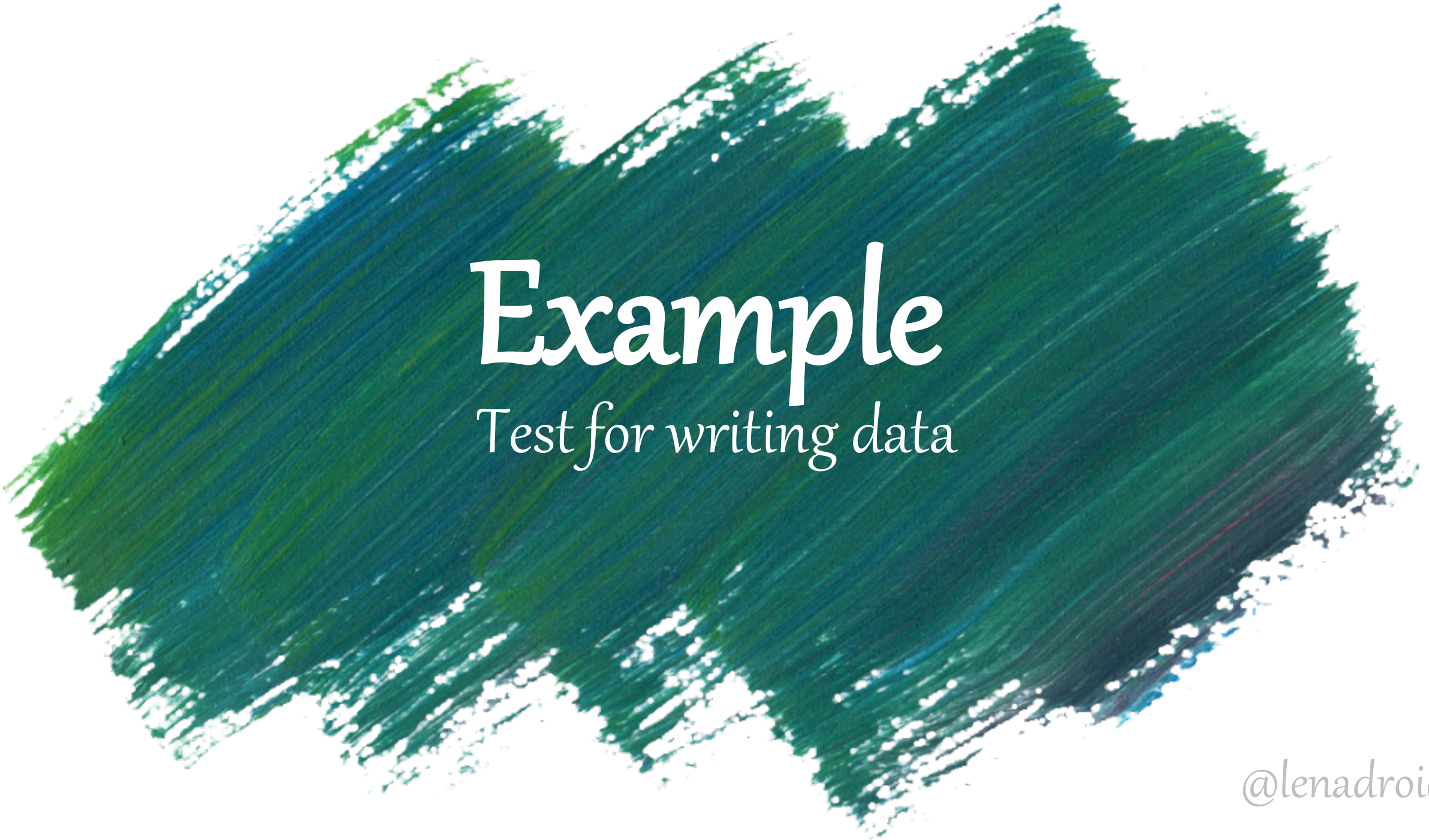
cassandra-statefulset.yaml

```yaml
          - name: POD_IP
            valueFrom:
              fieldRef:
                fieldPath:
                status.podIP
        readinessProbe:
          exec:
            command:
            - /bin/bash
            - -c
            - /ready-probe.sh
          initialDelaySeconds: 15
          timeoutSeconds: 5
        volumeMounts:
        - name: cassandra-data
          mountPath: /cassandra_data
  volumeClaimTemplates:
  - metadata:
      name: cassandra-data
    spec:
      accessModes: [
      "ReadWriteOnce" ]
      storageClassName:
      managed-premium
      resources:
        requests:
          storage: 1Gi
```

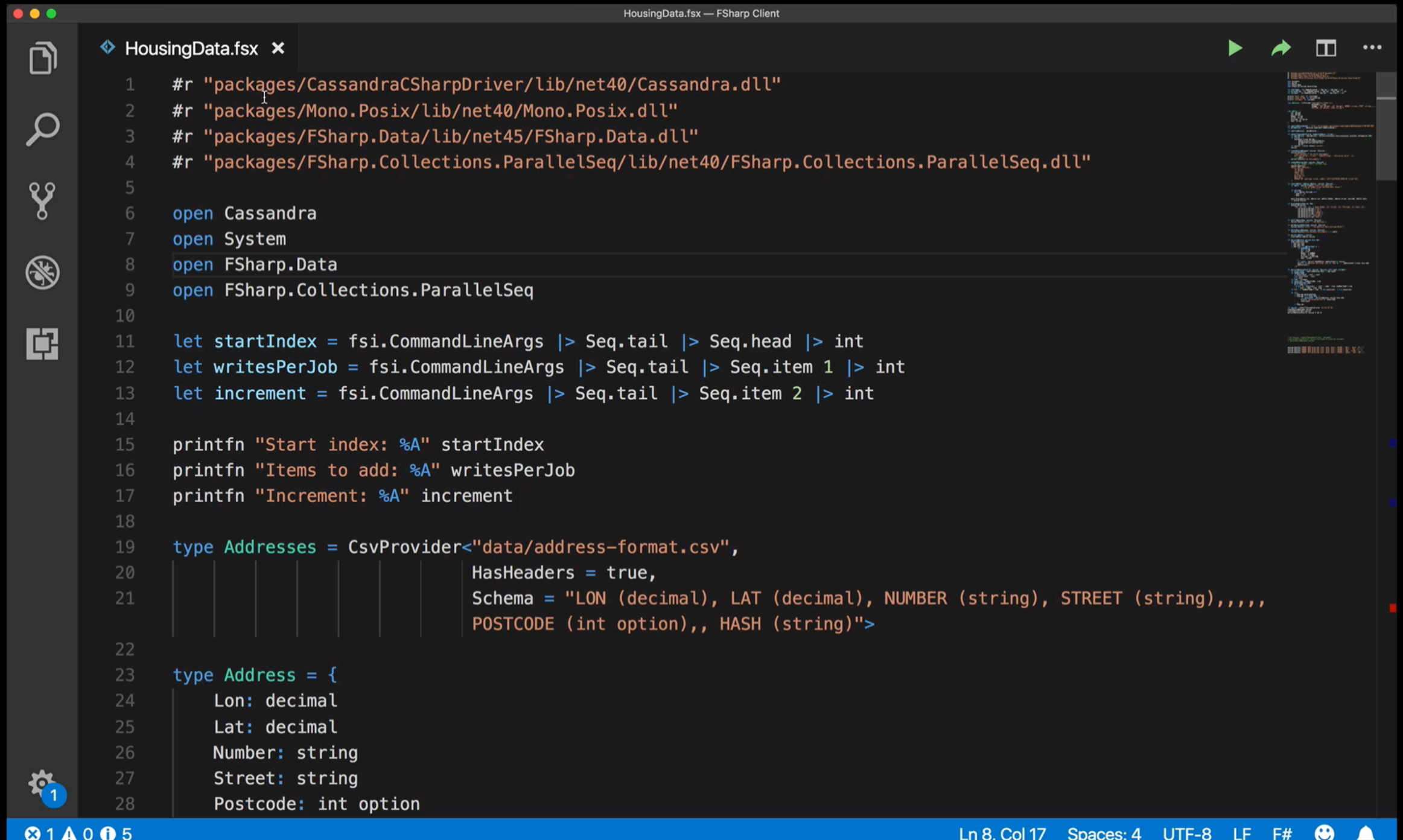



1. bash
lena:gotochicago lenok$

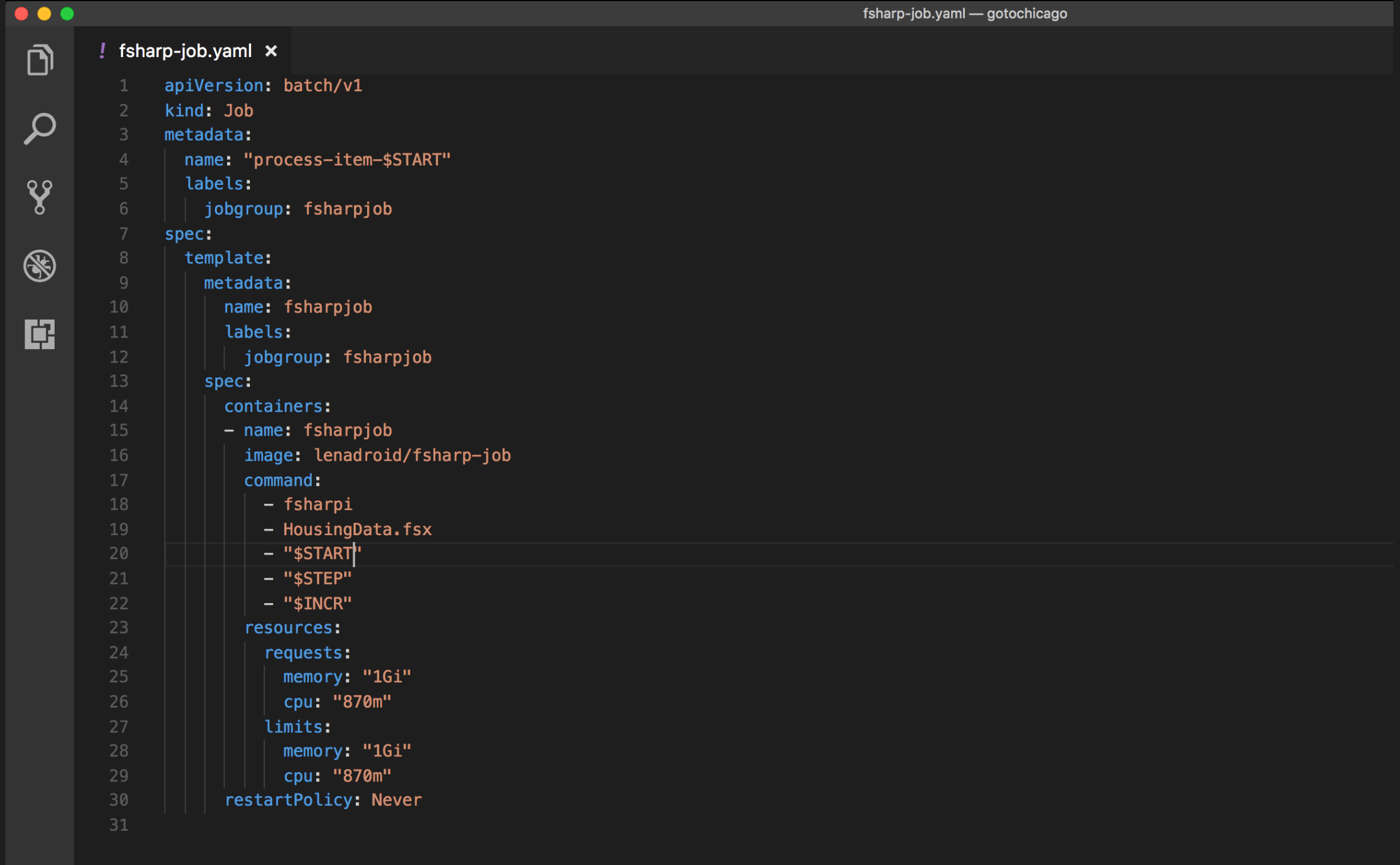
```
lena:gotochicago lenok$ kubectl get services
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
cassandra            ClusterIP      None           <none>          9042/TCP         45m
cassandra-external   LoadBalancer   10.0.186.193   52.234.227.80   9042:30734/TCP   1d
kubernetes           ClusterIP      10.0.0.1       <none>          443/TCP          1d
lena:gotochicago lenok$
```

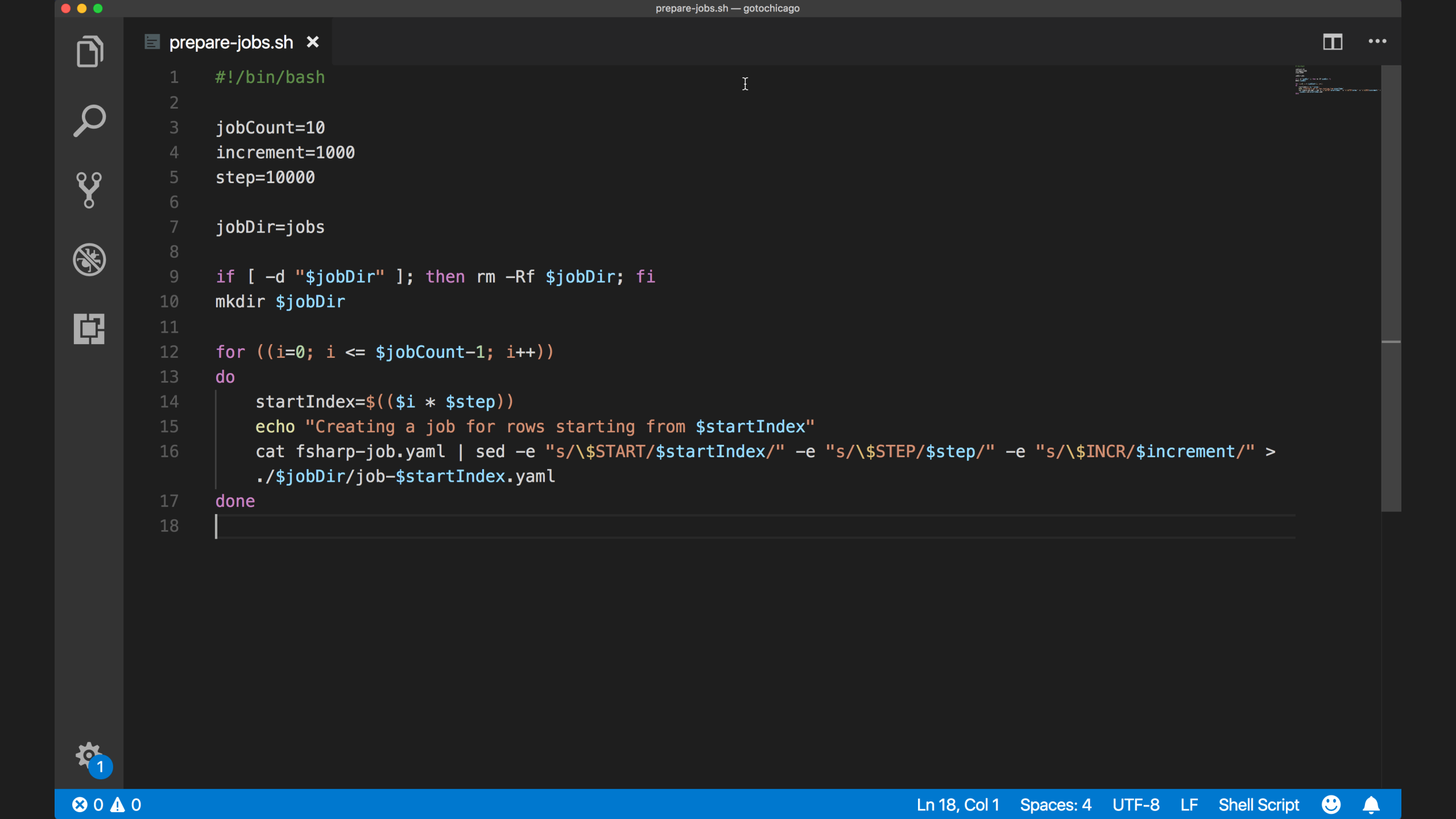


@lenadroid


HousingData.fsx — FSharp Client
HousingData.fsx
#r "packages/CassandraCSharpDriver/lib/net40/Cassandra.dll"
#r "packages/Mono.Posix/lib/net40/Mono.Posix.dll"
#r "packages/FSharp.Data/lib/net45/FSharp.Data.dll"
#r "packages/FSharp.Collections.ParallelSeq/lib/net40/FSharp.Collections.ParallelSeq.dll"
open Cassandra
open System
open FSharp.Data
open FSharp.Collections.ParallelSeq
let startIndex = fsi.CommandLineArgs |> Seq.tail |> Seq.head |> int
let writesPerJob = fsi.CommandLineArgs |> Seq.tail |> Seq.item 1 |> int
let increment = fsi.CommandLineArgs |> Seq.tail |> Seq.item 2 |> int
printfn "Start index: %A" startIndex
printfn "Items to add: %A" writesPerJob
printfn "Increment: %A" increment
type Addresses = CsvProvider<"data/address-format.csv",
HasHeaders = true,
Schema = "LON (decimal), LAT (decimal), NUMBER (string), STREET (string),,,,,
POSTCODE (int option),, HASH (string)">
type Address = {
Lon: decimal
Lat: decimal
Number: string
Street: string
Postcode: int option
1 0 5
Ln 8, Col 17
Spaces: 4
UTF-8
LF
F#

Kubernetes Jobs

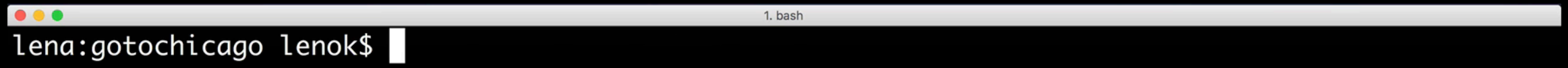

fsharp-job.yaml — gotochicago
fsharp-job.yaml
apiVersion: batch/v1
kind: Job
metadata:
name: "process-item-$START"
labels:
jobgroup: fsharpjob
spec:
template:
metadata:
name: fsharpjob
labels:
jobgroup: fsharpjob
spec:
containers:
- name: fsharpjob
image: lenadroid/fsharp-job
command:
- fsharpi
- HousingData.fsx
- "$START"
- "$STEP"
- "$INCR"
resources:
requests:
memory: "1Gi"
cpu: "870m"
limits:
memory: "1Gi"
cpu: "870m"
restartPolicy: Never


prepare-jobs.sh — gotochicago
prepare-jobs.sh
#!/bin/bash
jobCount=10
increment=1000
step=10000
jobDir=jobs
if [-d "$jobDir"]; then rm -Rf $jobDir; fi
mkdir $jobDir
for ((i=0; i <= $jobCount-1; i++))
do
startIndex=$(($i * $step))
echo "Creating a job for rows starting from $startIndex"
cat fsharp-job.yaml | sed -e "s/\$START/$startIndex/" -e "s/\$STEP/$step/" -e "s/\$INCR/$increment/" >
./$jobDir/job-$startIndex.yaml
done
0 0
Ln 18, Col 1
Spaces: 4
UTF-8
LF
Shell Script

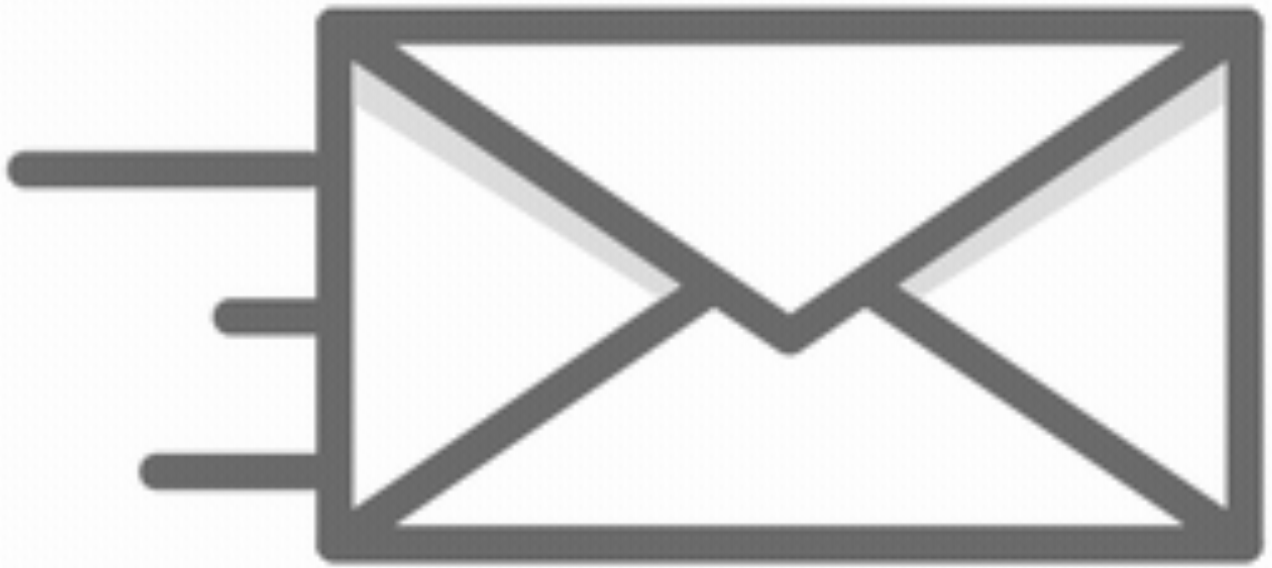
Example

Creating Kubernetes Jobs

@lenadroid


1. bash
lena:gotochicago lenok$

github.com/lenadroid/fsharp-job-kube



Things to note

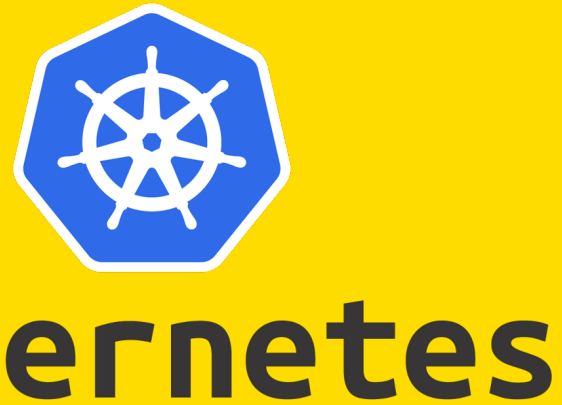
- Persistent Volumes are not deleted automatically
- Stateful Sets were in beta before Kubernetes 1.9
- Node-affinity

Running Spark jobs with Kubernetes scheduler

Art by @ashleymcnamara

@lenadroid

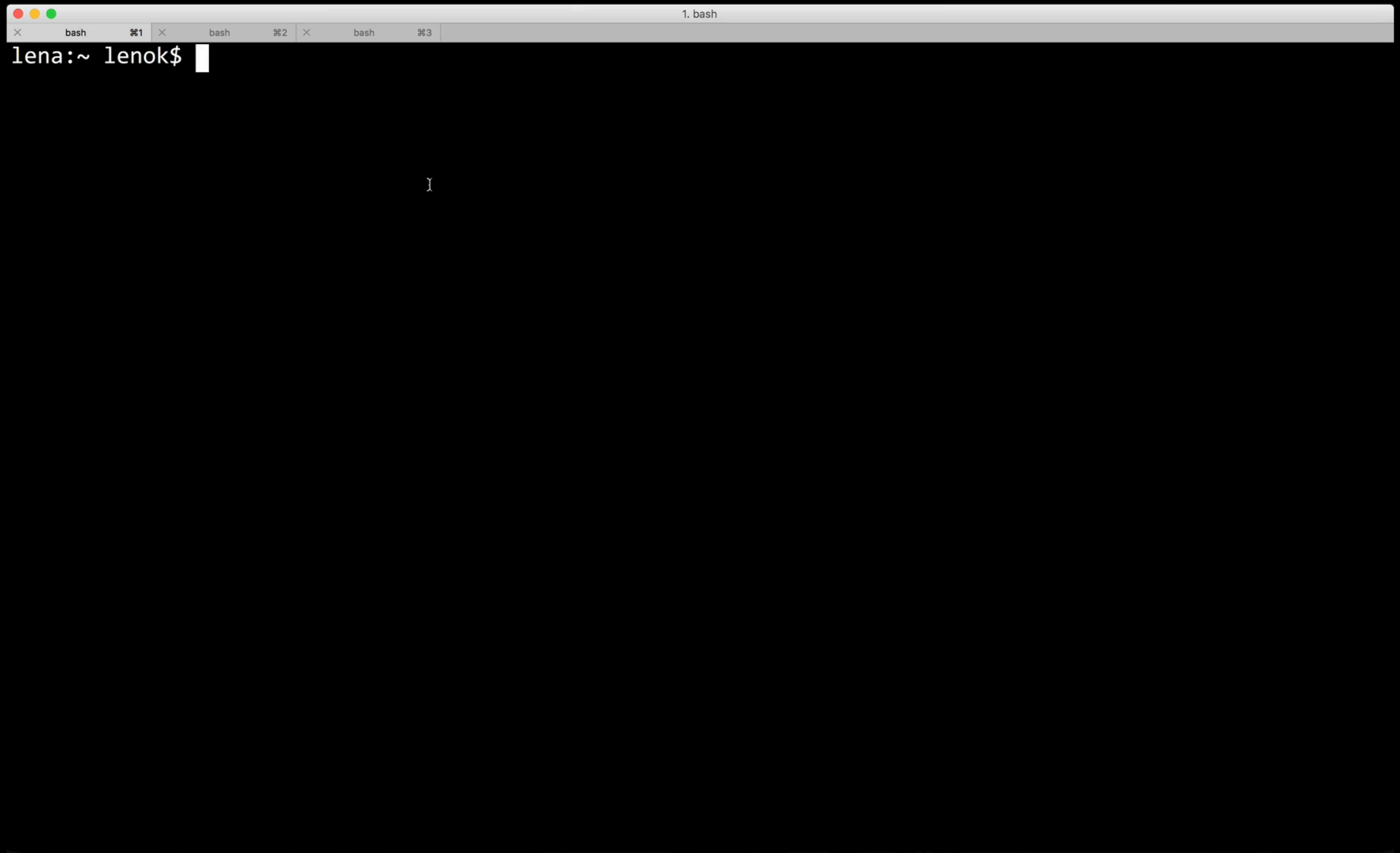
Native Kubernetes support is very new for Spark!



Spark Driver

Spark Executors


Example
Running Spark jobs on Kubernetes
& with Cassandra
@lenadroid


1. bash
bash ⌘1
bash ⌘2
bash ⌘3
lena:~ lenok$

Is running Spark jobs on Kubernetes a good idea?

Maybe

- For a test environment
- To try out a custom version of Spark quickly
- If you already have Kubernetes & want to use spare cycles
- Don't want to have too many different orchestrators

Should I run it in production?

Not yet

BUT IT WILL GET BETTER

1

YOU CAN HELP MAKE IT BETTER

@lenadroid

Note:

This is not the same as running standalone Spark on Kubernetes

These are  -native spark jobs

github.com/lenadroid/goto-cassandra-spark

Resources

- Cassandra blog post:

  `https://lenadroid.github.io/posts/cassandra-docker-fsharp.html`

- Distributed systems algorithms talk:

  `https://www.safaribooksonline.com/library/view/oscon-2017-/9781491976227/video306675.html`

- Stateful Sets article:

  `https://lenadroid.github.io/posts/stateful-sets-kubernetes-azure.html`

- All demos on GitHub @ lenadroid
- Twitter @ lenadroid

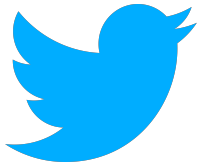
@lenadroid

Alena Hall

- Senior Cloud Developer Advocate, Engineer @ Azure
- Member of F# Software Foundation Board of Trustees
- Lives in beautiful Seattle, WA
- Loves doing data science, machine learning, functional programming and distributed systems at scale

@lenadroid

Thank you!

Alena Hall   @lenadroid