Apachecon 2019

THE LAST PICKLE

ANTHONY GRASSO

USING THE TLP TOOLCHAIN AS A CRYSTAL BALL FOR YOUR CLUSTER

THE LAST PICKLE

ABOUT ME

ABOUT THE LAST PICKLE

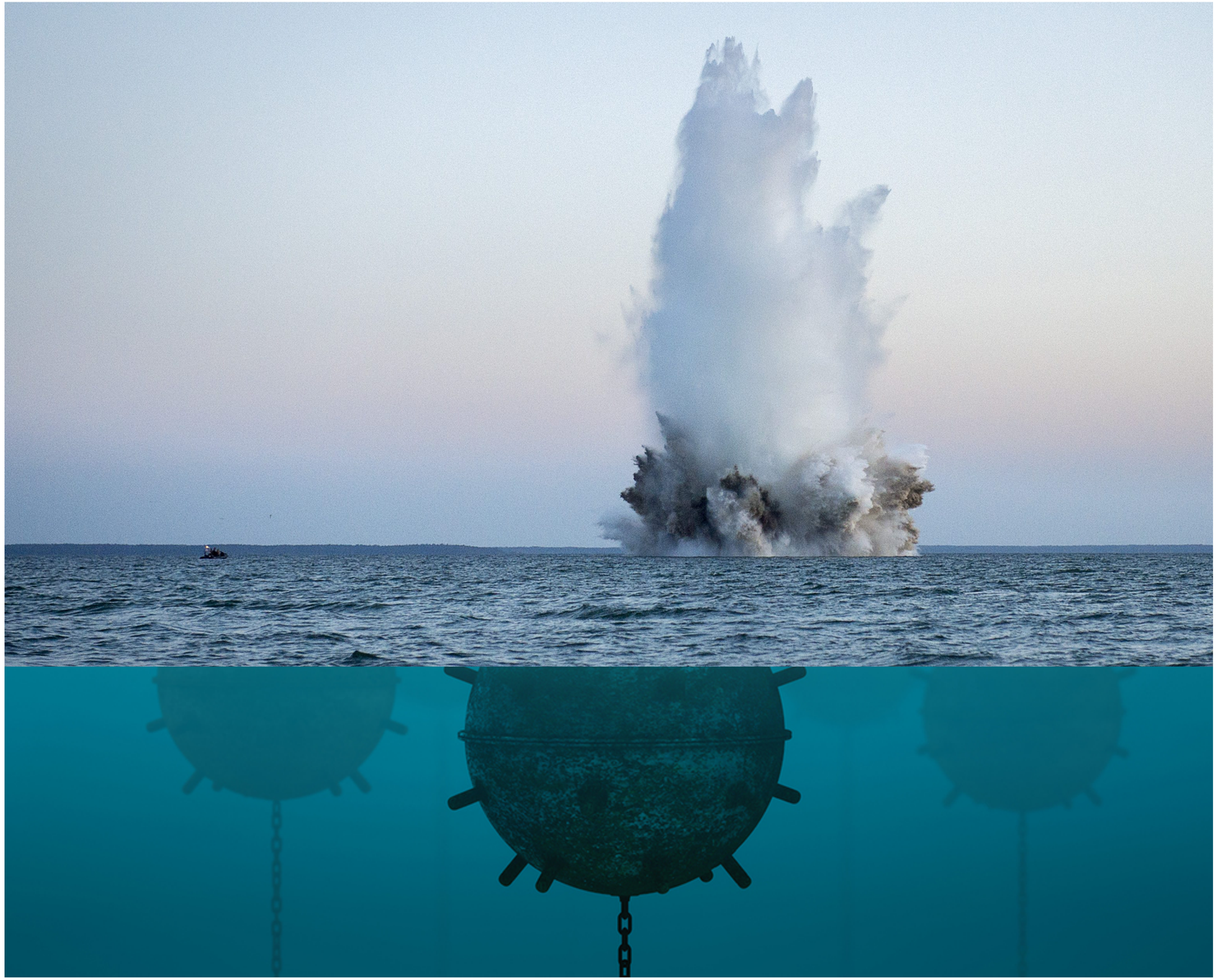
- Specialise in Apache Cassandra
- Help teams with
  - Delivery
  - Improving
- We want you to be successful

TLP TOOL CHAIN

- Performance testing/tuning
- Investigate settings
- Confirm code behaviour
- Compare/test C* versions

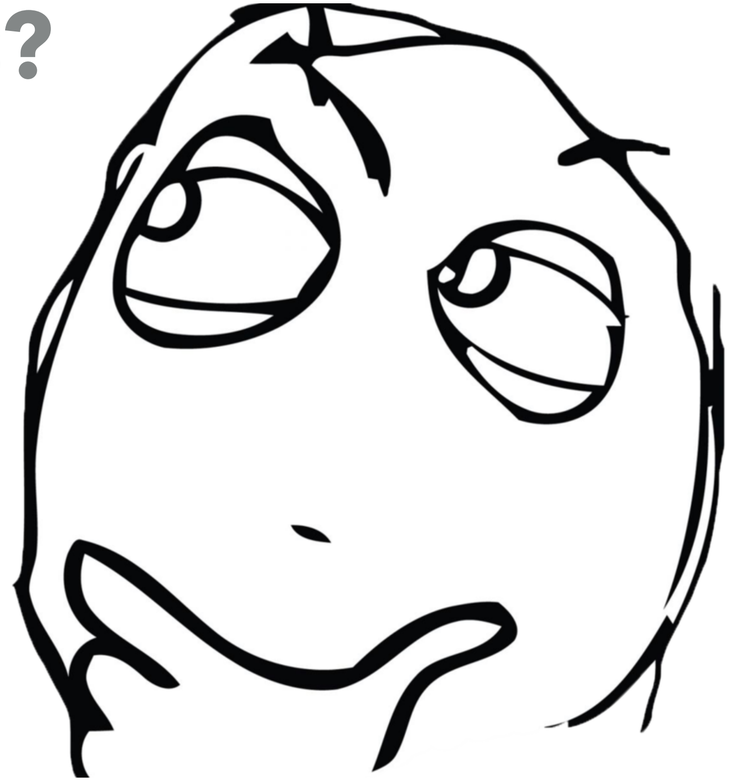
REAL WORLD ™ PROBLEM

- Are we confident that changing a setting will help the cluster?



WE NEED A STRESS TOOL

- Apply production workloads to the cluster
- What about Cassandra stress?



PROBLEMS WITH CASSANDRA-STRESS

- Infrequent updates
- Simple workloads
- Hard to configure and use

NEW STRESS TOOL

- Ships with common workloads
- Workloads configurable
- Easy to add new workloads

TLP-STRESS

- Uses Datastax Java driver
- Metrics use driver instrumentation

TLP-STRESS: WORKLOADS

- BasicTimeSeries
- CountersWide
- KeyValue
- LWT
- Locking
- Maps
- MaterializedViews
- RandomPartitionAccess
- UdtTimeSeries

TLP-STRESS: COMMANDS

- `run` - Run a workload
- `info` - Information about a workload
- `list` - List all known workloads

TLP-STRESS: KEY VALUE WORKLOAD EXAMPLE

- KeyValue workload
- 70% reads
- 100,000 partitions
- 10,000,000 iterations
- 50,000 rows pre-populated

TLP-STRESS: KEY VALUE WORKLOAD EXAMPLE

```
$ tlp-stress run KeyValue \
  --reads 0.7 \
  --partitions 100k \
  --iterations 10M \
  --populate 50k
```

```
Creating tlp_stress:
CREATE KEYSPACE
 IF NOT EXISTS tlp_stress
 WITH replication = {'class': 'SimpleStrategy',
'replication_factor':3 }

Creating schema
Executing 10000000 operations with consistency level LOCAL_ONE
Connected
Creating Tables
CREATE TABLE IF NOT EXISTS keyvalue (
                        key text PRIMARY KEY,
                        value text
                        ) WITH caching = {'keys': 'ALL',
'rows_per_partition': 'NONE'} AND default_time_to_live = 0
```

```
Preparing queries
Initializing metrics
Connecting
Creating generator random
Preparing statements.
1 threads prepared.
Populate Progress 100%
[==============================================================]
50000/50000 (0:00:04 / 0:00:00)

Pre-populate complete.
```

```
Starting main runner
Running
[Thread 0]: Running the profile for 10000000 iterations...
```

Writes			Reads			Errors	
Count	Latency (p99)	1min (req/s)	Count	Latency (p99)	1min (req/s)	Count	1min (errors/s)
32999	9.39	0	76637	12.04	0	0	0
74482	13.95	883.73	173507	11.14	2049.26	0	0
129948	15.76	883.73	302804	6.88	2049.26	0	0
173410	15.83	2245.13	403228	5.23	5227.09	0	0
230556	15.69	3390.66	537861	5.69	7909.98	0	0
281688	7.99	3390.66	657753	6.29	7909.98	0	0
334738	8.75	4546.51	781929	6.82	10622.34	0	0
366948	8.75	4546.51	857862	7.44	10622.34	0	0
420986	8.75	5222.95	982964	7.11	12205.07	0	0
473449	8.75	6244.76	1105734	5.3	14588.35	0	0

```
...
Stress complete, 1.
```

TLP-STRESS: TIME SERIES WORKLOAD EXAMPLE

- BasicTimeSeries workload
- 50% reads
- 1,000,000 partitions
- Run for 4 days, 13 hours, 21 minutes
- TimeWindowCompactionStrategy
- 6 hour TTL

TLP-STRESS: TIME SERIES WORKLOAD EXAMPLE

```
$ tlp-stress run BasicTimeSeries \
  --reads 0.5 \
  --partitions 1M \
  --duration 4d 13h 21m \
  --compaction "{
      'class':'TimeWindowCompactionStrategy',
      'compaction_window_unit':'HOURS',
      'compaction_window_size':6
    }" \
  --ttl 21600
```

```
Creating tlp_stress:
CREATE KEYSPACE
 IF NOT EXISTS tlp_stress
 WITH replication = {'class': 'SimpleStrategy', 'replication_factor':3 }

Creating schema
Executing 0 operations with consistency level LOCAL_ONE
Connected
Creating Tables
CREATE TABLE IF NOT EXISTS sensor_data (
                            sensor_id text,
                            timestamp timeuuid,
                            data text,
                            primary key(sensor_id, timestamp))
                            WITH CLUSTERING ORDER BY (timestamp DESC) AND compaction =
{
      'class':'TimeWindowCompactionStrategy',
      'compaction_window_unit':'HOURS',
      'compaction_window_size':6
    } AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'} AND
default_time_to_live = 21600
```

```
Preparing queries
Initializing metrics
Connecting
Creating generator random
Preparing statements.
1 threads prepared.
Starting main runner
Running
[Thread 0]: Running the profile for 6561min...
```

Writes			Reads			Errors	
Count	Latency (p99)	1min (req/s)	Count	Latency (p99)	1min (req/s)	Count	1min (errors/s)
24384	10.25	0	24217	7.91	0	0	0
75336	12.89	11407	75039	10.43	11323	0	0
138179	13.87	11407	137140	16.65	11323	0	0
189391	3.78	12062.73	188213	8.74	11977.24	0	0
257294	8.95	12719.65	255883	8.37	12633.45	0	0
322926	4.88	12719.65	321679	2.5	12633.45	0	0
382906	28.59	13356.85	381638	2.66	13280.61	0	0
444251	12	13356.85	442828	2.67	13280.61	0	0
511648	4.88	13975.87	510521	6.79	13898.84	0	0
569022	14.78	14502.66	567793	5.74	14437.87	0	0

```
...
Stress complete, 1.
```

TLP-STRESS: RANDOM PARTITION ACCESS EXAMPLE

- RandomPartitionAccess workload
- 98% reads
- 1,000,000 partitions
- 10,000,000 iterations,
- 1,000,000 rows pre-populated
- Lower index interval

TLP-STRESS: INSPECT WORKLOAD

```
$ tlp-stress info RandomPartitionAccess
CREATE TABLE IF NOT EXISTS random_access (
                           partition_id text,
                           row_id int,
                           value text,
                           primary key (partition_id, row_id)
                       )
Default read rate: 0.01 (override with -r)

Dynamic workload parameters (override with --workload.name=X)

Name   | Description                                          | Type
rows   | Number of rows per partition, defaults to 100        | kotlin.Int
select | Select random row or the entire partition.           | kotlin.String
       | Acceptable values: row, partition                    |
```

TLP-STRESS: RANDOM PARTITION ACCESS EXAMPLE

```
$ tlp-stress run RandomPartitionAccess \
    --reads 0.98 \
    --partitions 1M \
    --iterations 10M \
    --populate 1M \
    --cql "ALTER TABLE tlp_stress.random_access
              WITH max_index_interval = 1024
              AND  min_index_interval = 64" \
    --workload.rows="50" \
    --workload.select="row"
```

```
Creating tlp_stress:
CREATE KEYSPACE
 IF NOT EXISTS tlp_stress
 WITH replication = {'class': 'SimpleStrategy', 'replication_factor':3 }

Creating schema
Executing 10000000 operations with consistency level LOCAL_ONE
Connected
Creating Tables
CREATE TABLE IF NOT EXISTS random_access (
                        partition_id text,
                        row_id int,
                        value text,
                        primary key (partition_id, row_id)
                    ) WITH caching = {'keys': 'ALL', 'rows_per_partition':
'NONE'} AND default_time_to_live = 0
ALTER TABLE tlp_stress.random_access
            WITH max_index_interval = 1024
            AND  min_index_interval = 64
```

```
Preparing queries
Preparing single row reads
Initializing metrics
Connecting
Creating generator random
Preparing statements.
Preparing single row reads
1 threads prepared.
Using 50 rows per partition
Populate Progress 100%
[=============================================================>]
1000000/1000000 (0:00:30 / 0:00:00)

Pre-populate complete.
```

TLP-STRESS: RUN OPTIONS - REPORTING

- Reporting:
  - `csv` - compressed CSV
- Prometheus
  - Exposes metrics

TLP-STRESS: OTHER RUN OPTIONS - CLIENT

- client:
  - `cl` - consistency level
  - `concurrency` - concurrent requests
  - `coordinatoronly` - connect to coordinator node
  - `dc` - datacenter

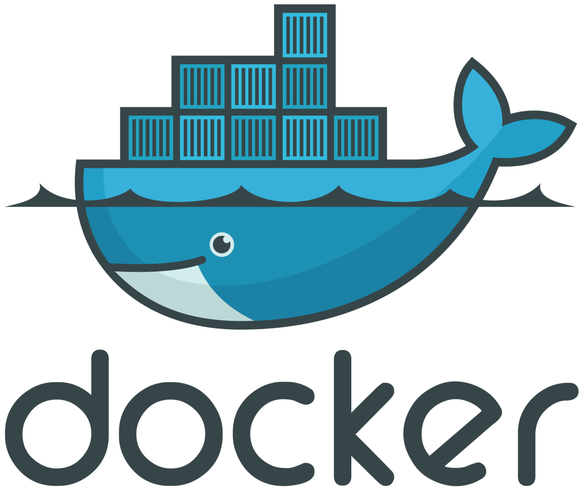
TLP-STRESS: OTHER RUN OPTIONS - TABLE

- table:
  - `compression` - set compression
  - `keycache` - set keycache
  - `rowcache` - set rowcache
  - `replication` - set data replication factor

TLP-STRESS: OTHER RUN OPTIONS - WORKLOAD

- **workload:**
  - `partitiongenerator` - **partition generation method**
    - **random, sequential, gaussian**
  - `rate` - **rate limiter**

TLP-STRESS: 4.0 RELEASE



TLP-STRESS: CODE

- Source: https://github.com/thelastpickle/tlp-stress
- Documentation: http://thelastpickle.com/tlp-stress/
- Mailing List: tlp-dev-tools@googlegroups.com

TLP-STRESS: FUTURE WORK

- Workload customisation
  - Change field size
  - Add fields
- Cluster mode

WE NEED A CLUSTER TO TEST WITH

TEST WITH CCM

```
$ ccm node1 nodetool status

Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load          Tokens  Owns    Host ID    Rack
UN  127.0.0.1   104.25 KiB    16      71.6%   53333cb9   rack1
UN  127.0.0.2   89.58 KiB     16      68.8%   e0a84f16   rack1
UN  127.0.0.3   15.52 KiB     16      59.6%   c6b2c939   rack1
```

```
$ tlp-stress run KeyValue \
    --duration 15m \
    --partitions 10M \
    --reads 0.9 \
    --rate 1000 \
    --populate 10M \
    --compaction lcs \
    --keycache NONE
```

```
Creating schema
Executing 0 operations ...
Connected
Creating Tables

...

Preparing queries
Initializing metrics
Connecting
Creating generator random
Preparing statements.
1 threads prepared.
Populate Progress
0% [>                                                  ] 0/10000000 (0:00:01 / 2:46:02)
```

THE LAST PICKLE

Count	Latency (p99)	1min (req/s)
63644	0.53	100.13
63942	0.54	100.13
64258	5.26	100.12
64555	4.47	100.12
64868	7.52	100.32
65196	7.2	101
65476	8.65	101
65760	8.65	100.52
66064	8.08	100.52
66368	14.27	100.49

Writes

Count	Latency (p99)	1min (req/s)
66663	14.27	100.39
66938	12.35	100.39
67227	12.35	99.88
67509	12.35	99.88
67810	12.24	99.66
68127	12.24	99.72
68441	8.08	99.72
68731	8.08	99.73
69039	7.2	99.73
69346	7.2	100.2

Count	Latency (p99)	1min (req/s)
570329	0.49	899.84
573030	0.53	899.84
575717	**1.69**	899.84
578418	**1.11**	899.84
581102	**1.69**	899.66
583778	[illegible]	898.98
586495	**16.18**	898.98
589211	**5.5**	899.46
591909	**5.5**	899.46
594607	**16.18**	899.46

Reads

Count	Latency (p99)	1min (req/s)
597312	**5.5**	899.59
600020	**5.5**	899.59
602743	**5.5**	900.11
605463	**13.47**	900.11
608166	**13.47**	900.34
610848	**5.5**	900.28
613533	**5.5**	900.28
616245	**4.36**	900.26
618935	**4.36**	900.26
621630	**4.36**	899.8

Count	1min (errors/s)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Errors

Count	1min (errors/s)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Scrolled terminal!

Opened Chrome!!

Browsed NETFLIX!!!

WE NEED A REAL CLUSTER

IT TAKES TIME TO LAUNCHING A CLUSTER

- Launch resources
- Provision nodes
- Configure C*

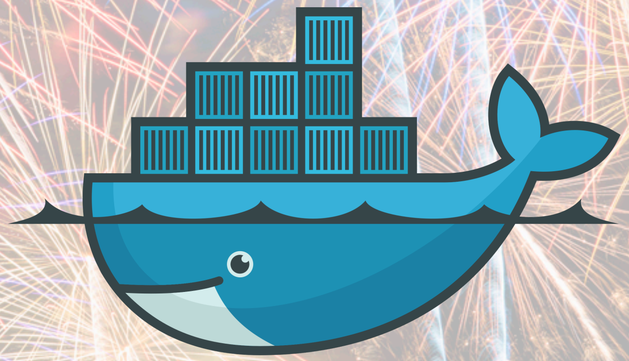
NEW CLUSTER CREATION TOOL

- Similar to CCM
- Cluster < 10mins
- Configurable
- On “real” hardware

TLP-CLUSTER

Kotlin

Terraform

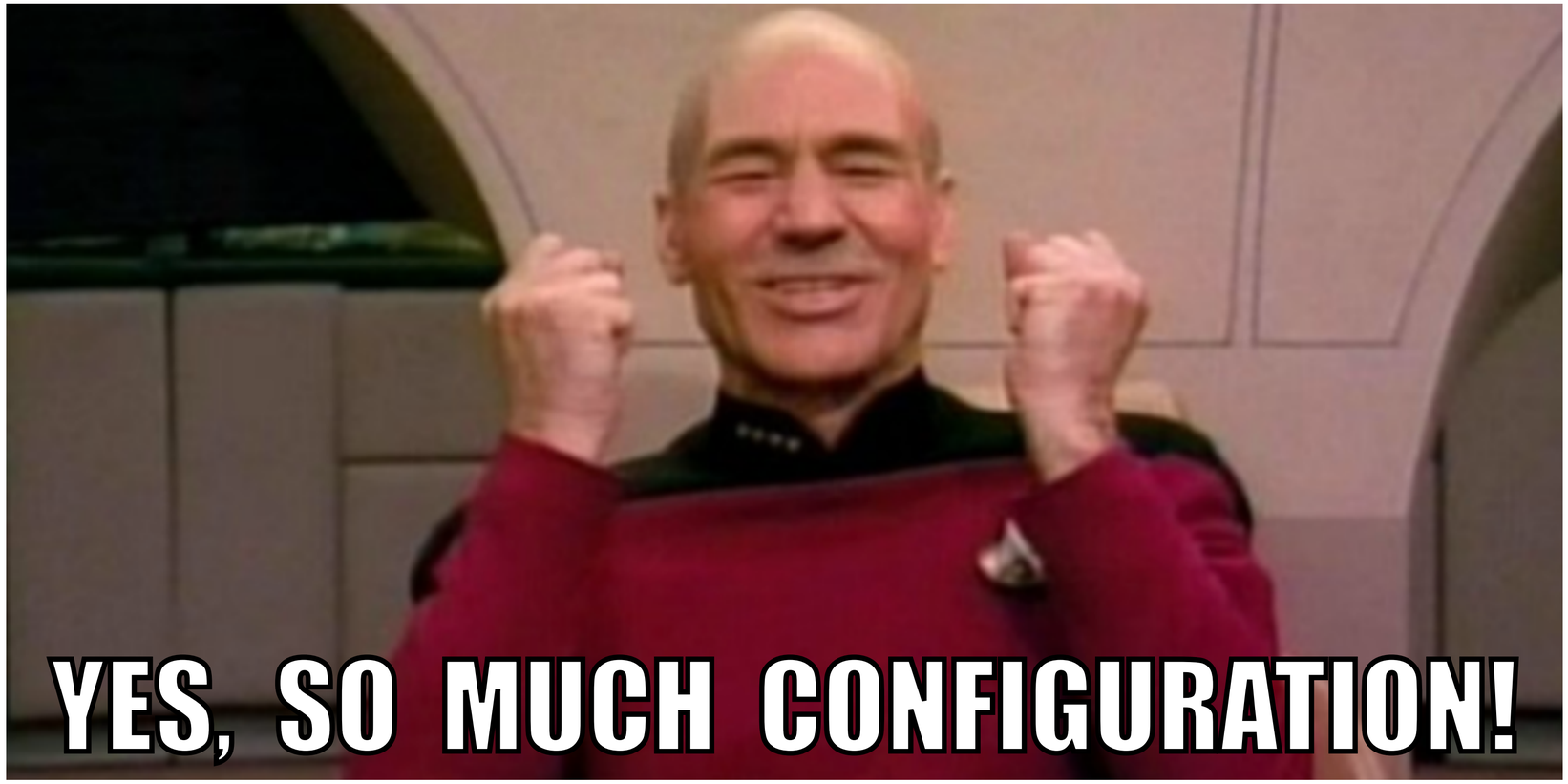


docker

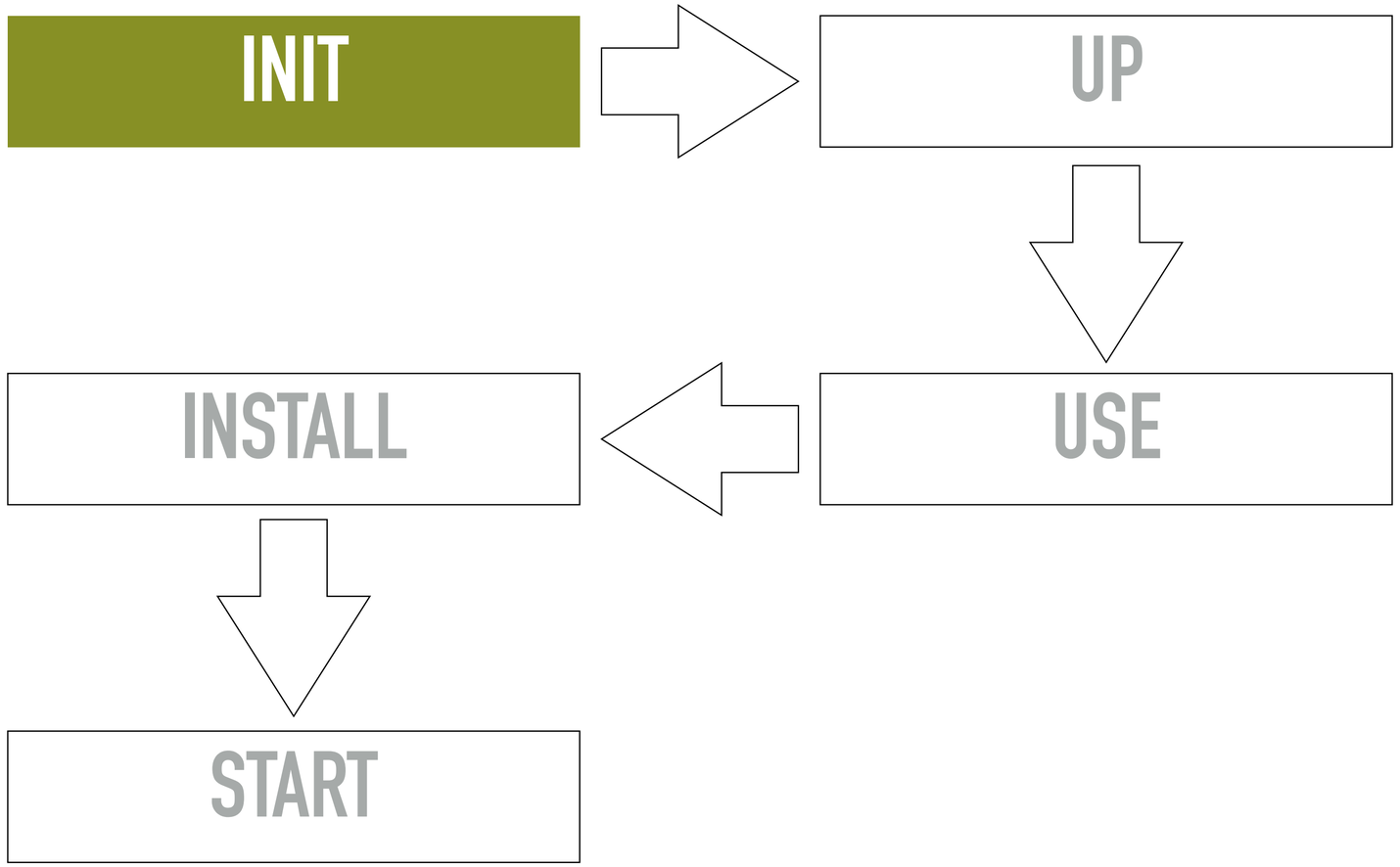
amazon web services™

TLP-CLUSTER: FEATURES

- Instance type
- Cassandra version
- Cassandra settings
- Cluster size
- Installation



TLP-CLUSTER: COMMANDS

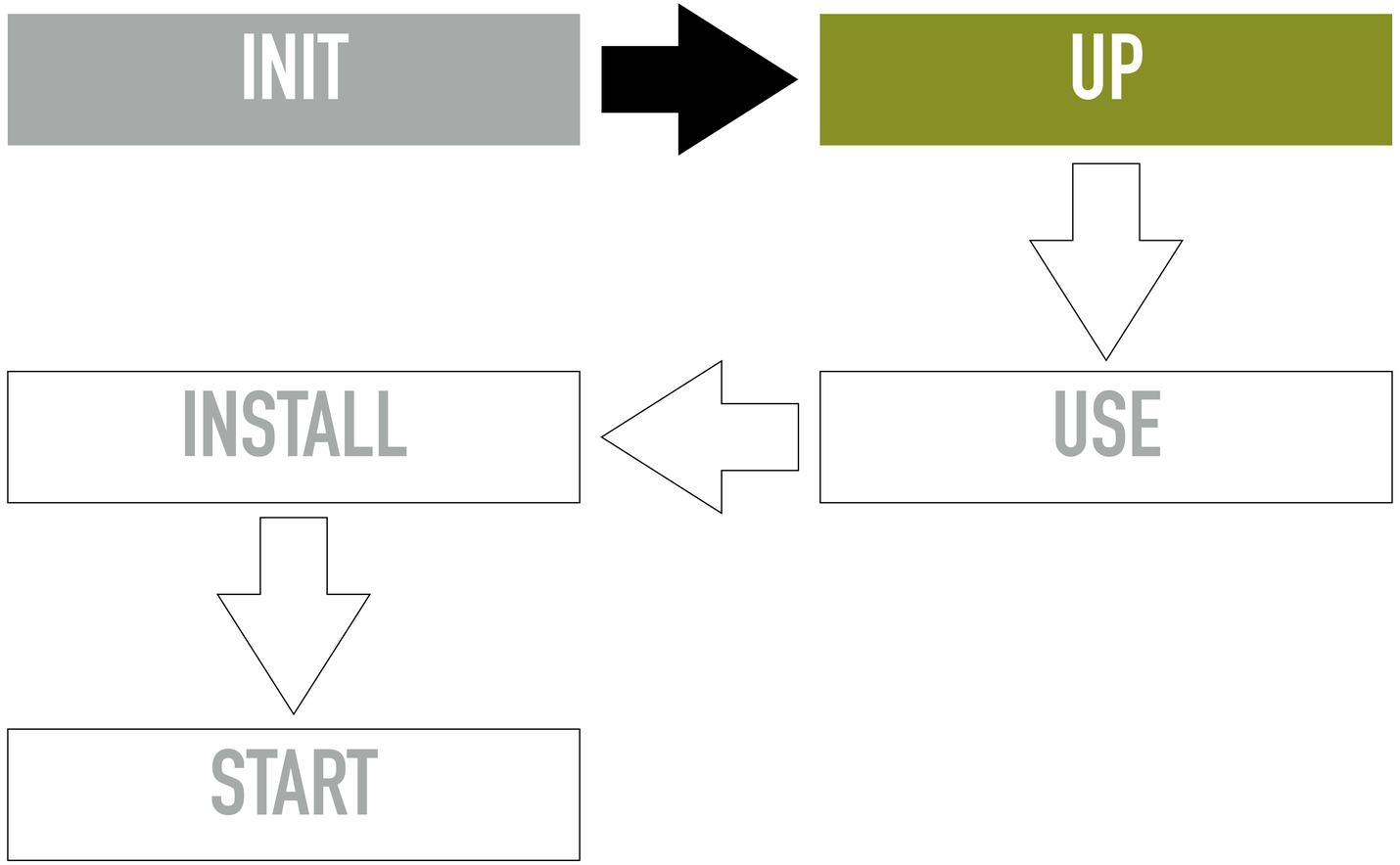


TLP-CLUSTER: EXAMPLE - INIT

```
$ tlp-cluster init \
    --cassandra 6 \
    --instance r5.xlarge \
    --stress 1 \
    --az a,b,c \
  TLP \
  APACHE-CON \
  "Apache Con demo cluster"
```

```
Initializing directory
Copying provisioning files
...
Copying JMX collector
Overriding default az list with [a, b, c]
Setting working directory inside container to /local
Starting hashicorp/terraform:0.11.14 container (dc42026957d6)
...
Terraform has been successfully initialized!
...
Your workspace has been initialized with 6 Cassandra instances
(r5.xlarge) and 1 stress instances in us-west-2
Next you'll want to run tlp-cluster up to start your
instances.
```

TLP-CLUSTER: COMMANDS



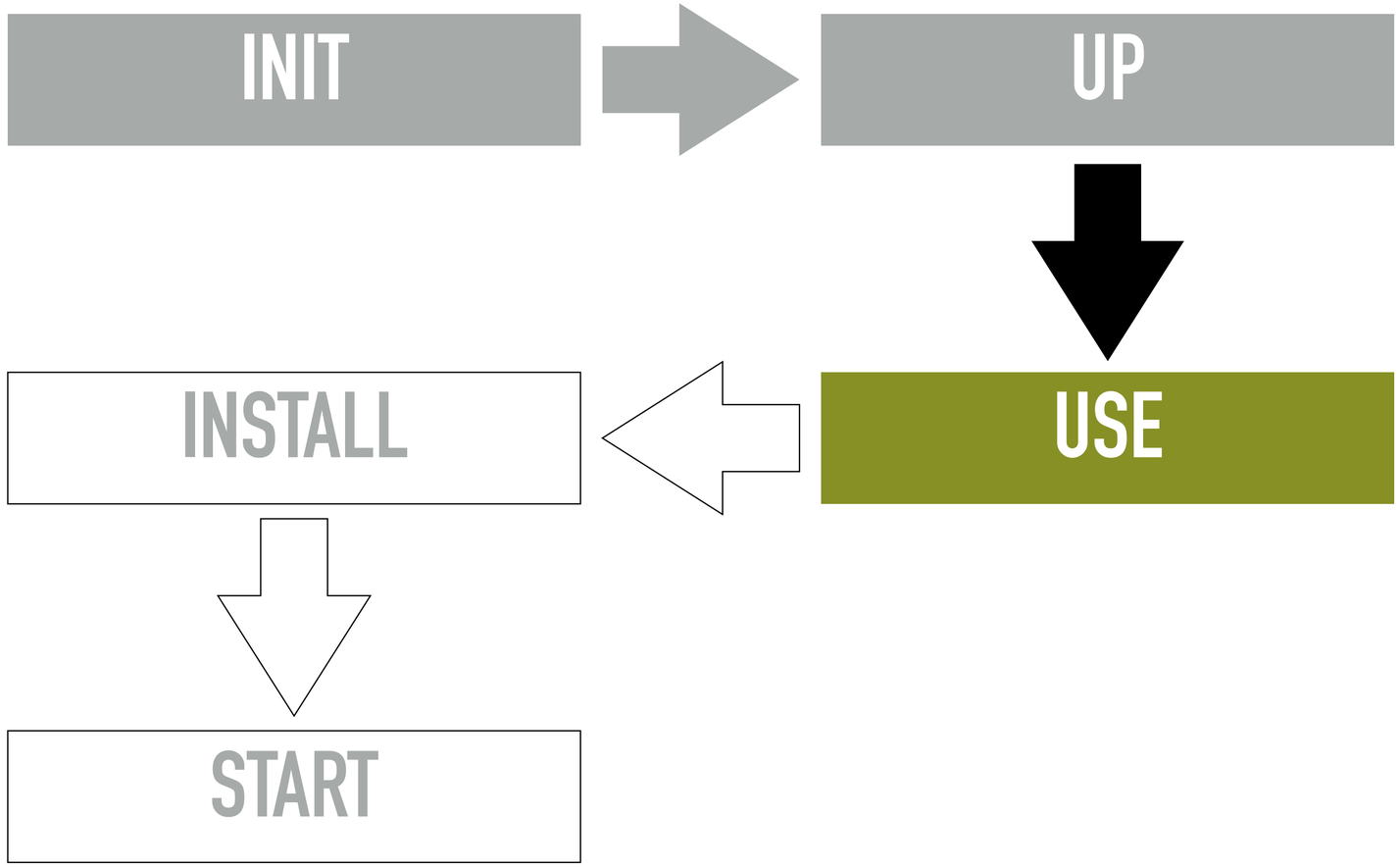
TLP-CLUSTER: EXAMPLE - UP

```
$ tlp-cluster up
```

```
Starting hashicorp/terraform:0.11.14 container (dcb6a78ea2ba)

Terraform will perform the following actions:

  + aws_instance.cassandra[0]
      ...
  + aws_instance.cassandra[5]
      ...
  + aws_instance.monitoring
      ...
  + aws_instance.stress
      ...
  + aws_security_group.NO-TICKET_TlpClusterSG_1567401948
      ...

Plan: 9 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

  Enter a value:
```

```
 Apply complete! Resources: 9 added, 0 changed, 0 destroyed.
 Instances have been provisioned.

 You can edit the provisioning scripts before running them,
they've been copied to ./provisioning.

 Next you'll probably want to run tlp-cluster build to create
a new build, or tlp-cluster use <version> if you already have
a Cassandra build you'd like to deploy.

 Writing ssh config file to sshConfig.
 The following alias will allow you to easily work with the
cluster:

  source env.sh
```

TLP-CLUSTER: COMMANDS

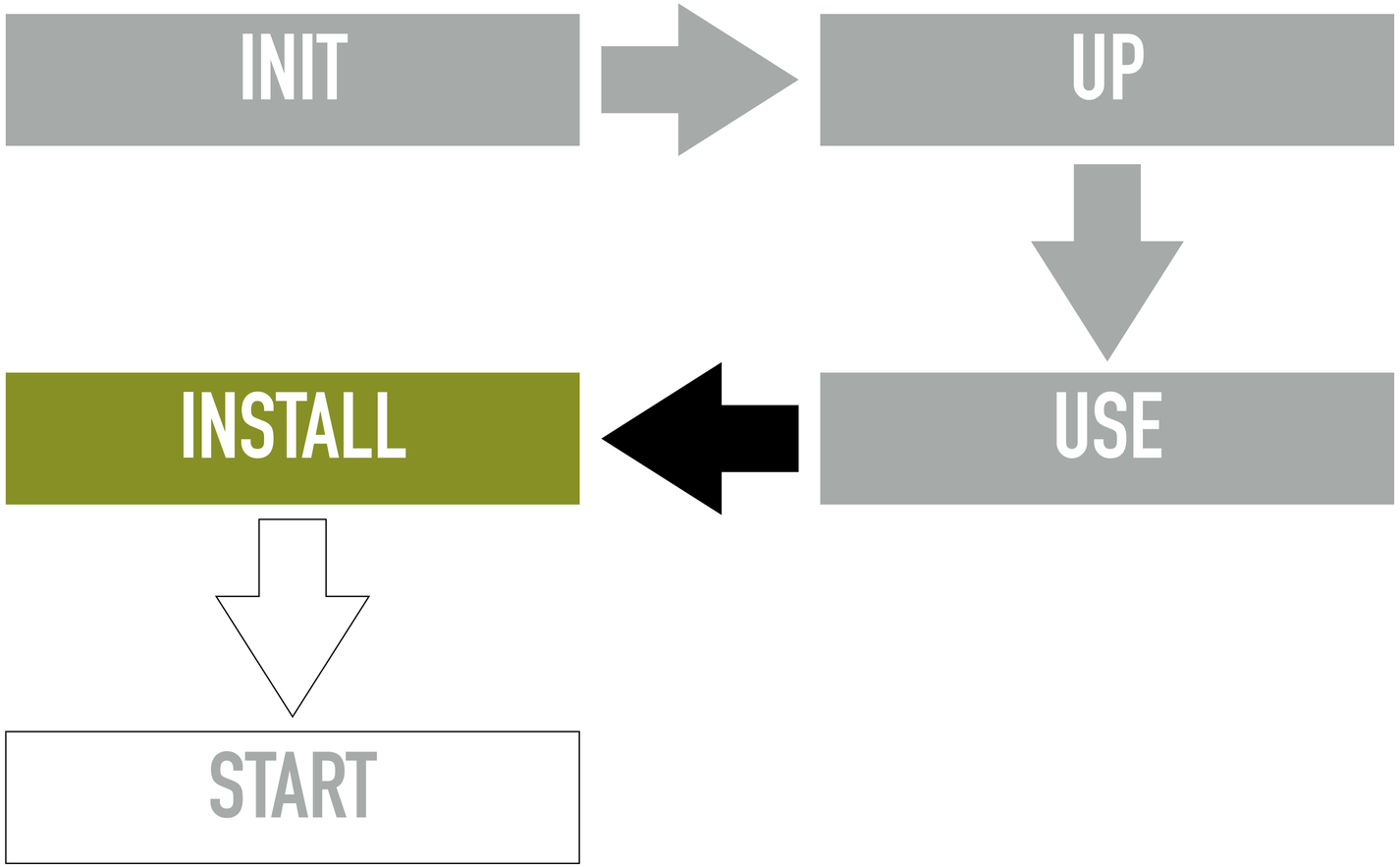


TLP-CLUSTER: EXAMPLE - USE

```
$ tlp-cluster use \
    --config "cluster_name:ApacheCon" \
    --config "num_tokens:16" \
    --config "trickle_fsync:true" \
    --config "dynamic_snitch:false" \
  3.11.4
```

```
Destination artifacts: provisioning/cassandra
Using released version 3.11.4
...
Starting ubuntu:bionic container (da2b2904d653)
...
DONE
Cassandra deb and config copied to
provisioning/.  Config files are located in
provisioning/cassandra.
 Use tlp-cluster install to push the artifacts
to the nodes.
```

TLP-CLUSTER: COMMANDS

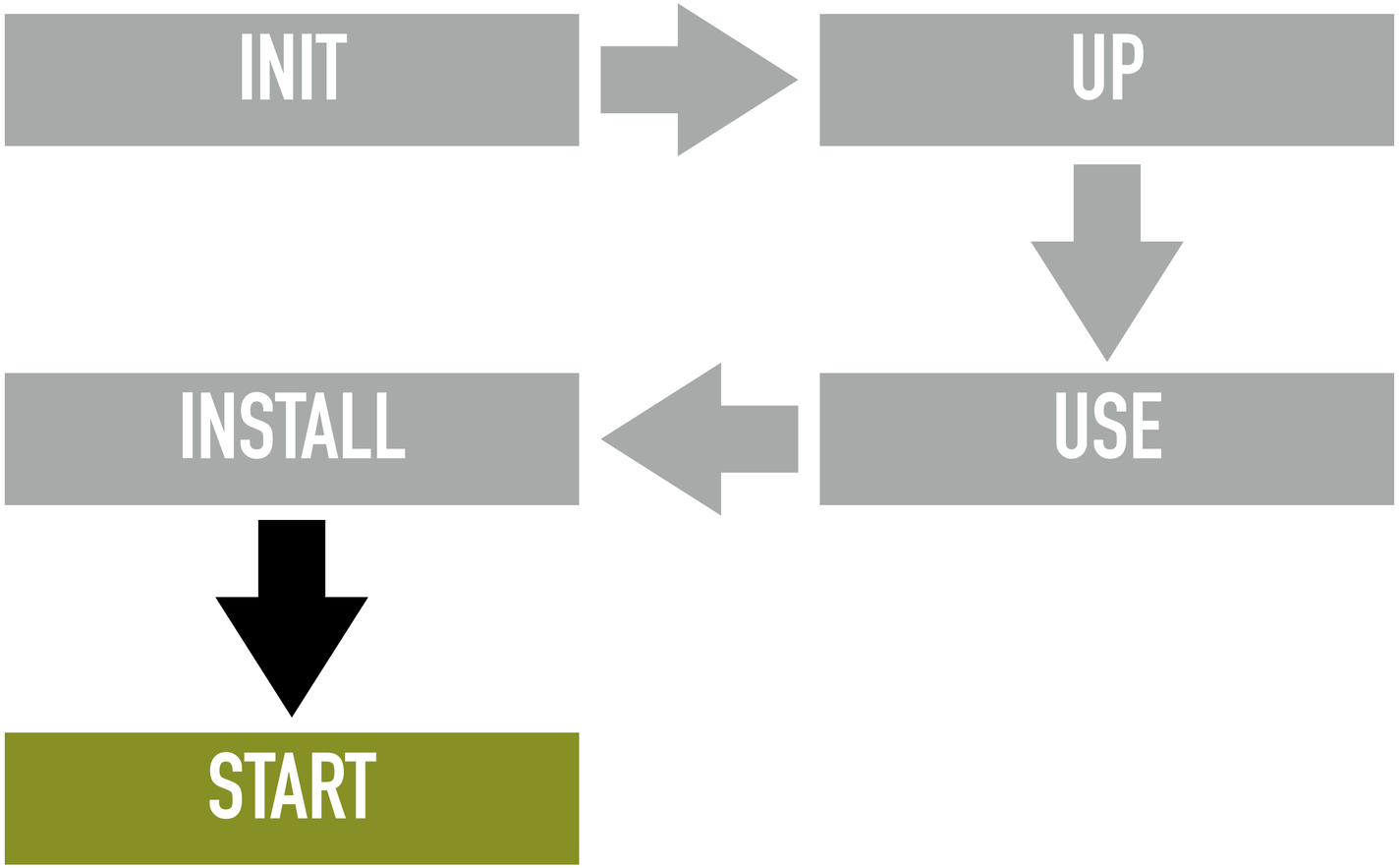


TLP-CLUSTER: EXAMPLE - INSTALL

```
$ tlp-cluster install
```

```
Provisioning cassandra
Starting thelastpickle/tlp-cluster_pssh container (4bb8eff4b8a1)
[1] 03:58:36 [SUCCESS] 54.245.23.137
[2] 03:59:18 [SUCCESS] 54.200.90.149
[3] 03:59:25 [SUCCESS] 52.43.138.110
[4] 03:59:28 [SUCCESS] 35.165.72.245
[5] 03:59:29 [SUCCESS] 34.221.156.217
[6] 03:59:39 [SUCCESS] 35.167.215.231
...
Provisioning stress
Starting thelastpickle/tlp-cluster_pssh container (46415b7cc427)
[1] 04:00:52 [SUCCESS] 54.188.46.250
...
Provisioning monitoring
Starting thelastpickle/tlp-cluster_pssh container (27590d3b7d58)
[1] 04:02:24 [SUCCESS] 18.236.169.66
...
Now run tlp-cluster start to fire up the cluster.
```

TLP-CLUSTER: COMMANDS



TLP-CLUSTER: EXAMPLE - START

```
$ tlp-cluster start
```

```
Starting thelastpickle/tlp-cluster_pssh container (cad5ed86a0ef)
[1] 04:06:47 [SUCCESS] 54.200.90.149
[2] 04:06:47 [SUCCESS] 34.221.156.217
[3] 04:06:47 [SUCCESS] 35.165.72.245
[4] 04:06:47 [SUCCESS] 35.167.215.231
[5] 04:06:47 [SUCCESS] 52.43.138.110
[6] 04:06:47 [SUCCESS] 54.245.23.137
...
cassandra service successfully started

Starting thelastpickle/tlp-cluster_pssh container (85e06a3c3dd7)
[1] 04:06:58 [SUCCESS] 18.236.169.66
● prometheus.service - Prometheus
...
prometheus service successfully started

Starting thelastpickle/tlp-cluster_pssh container (ce68dd41c814)
[1] 04:06:58 [SUCCESS] 18.236.169.66
● grafana-server.service - Grafana instance
...
grafana-server service successfully started
```

```
You can access the monitoring UI using the
following URLs:
 - Prometheus: http://18.236.169.66:9090
 - Grafana:    http://18.236.169.66:3000

 Reaper is available on all Cassandra nodes with
the login 'admin', password 'admin'

 - Reaper:     http://35.167.215.231:8080/webui/
```

TLP-CLUSTER: ENV FILE

- Server names
- Commands

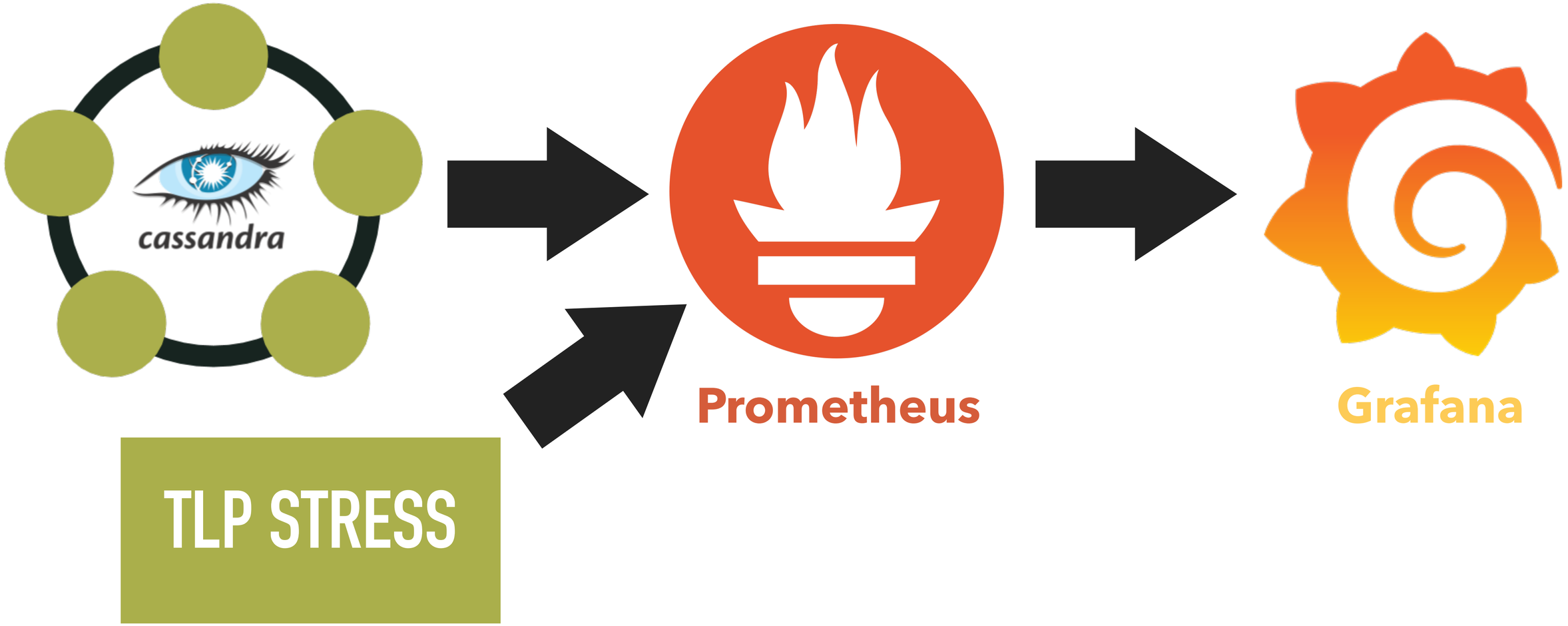
  `ssh, rsync, etc.`
- Rolling operations

  `c-restart, c-status, etc.`

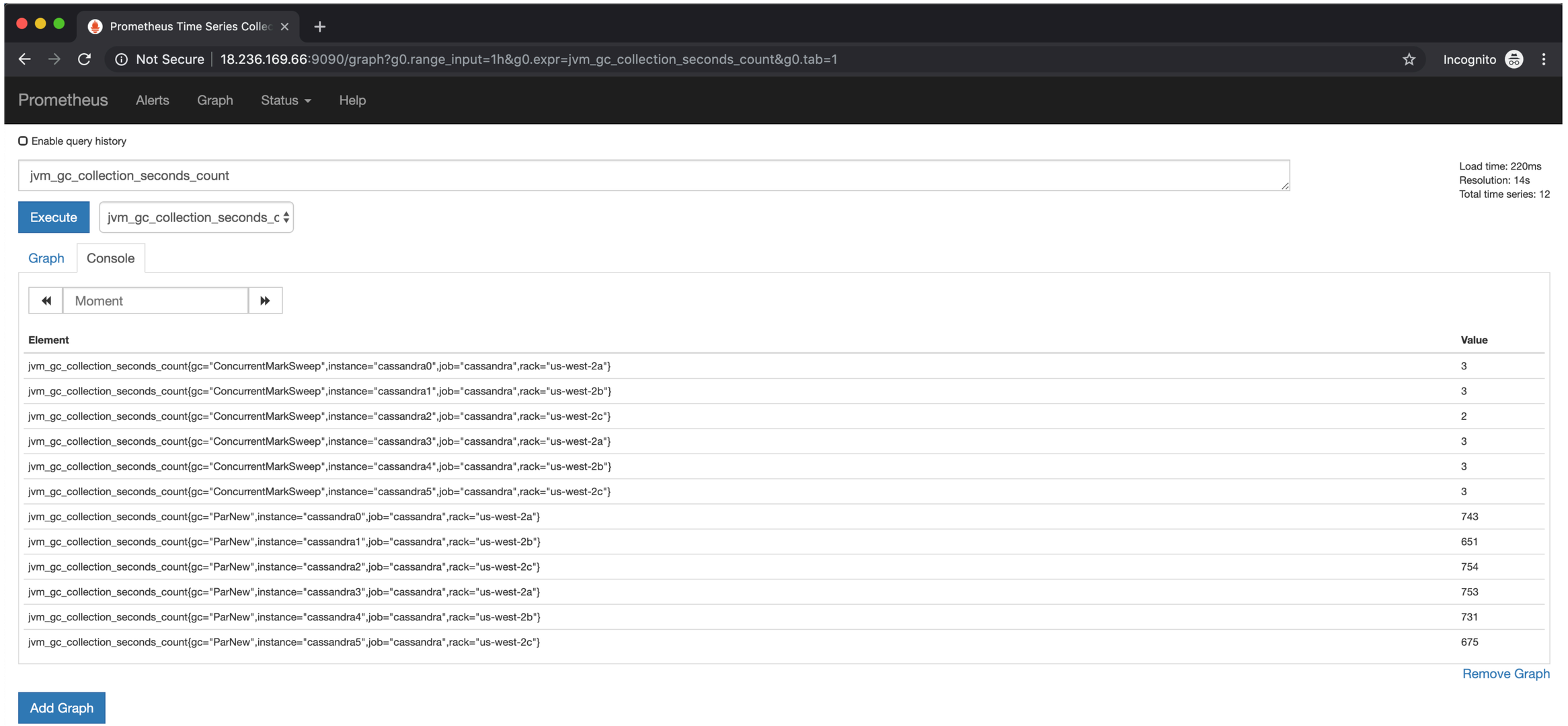
```
$ source env.sh
[WARNING] We are creating aliases which override
these commands:
   ssh
   sftp
   scp
   rsync

The aliases point the commands they override to
your new cluster.
To undo these changes exit this terminal.
```

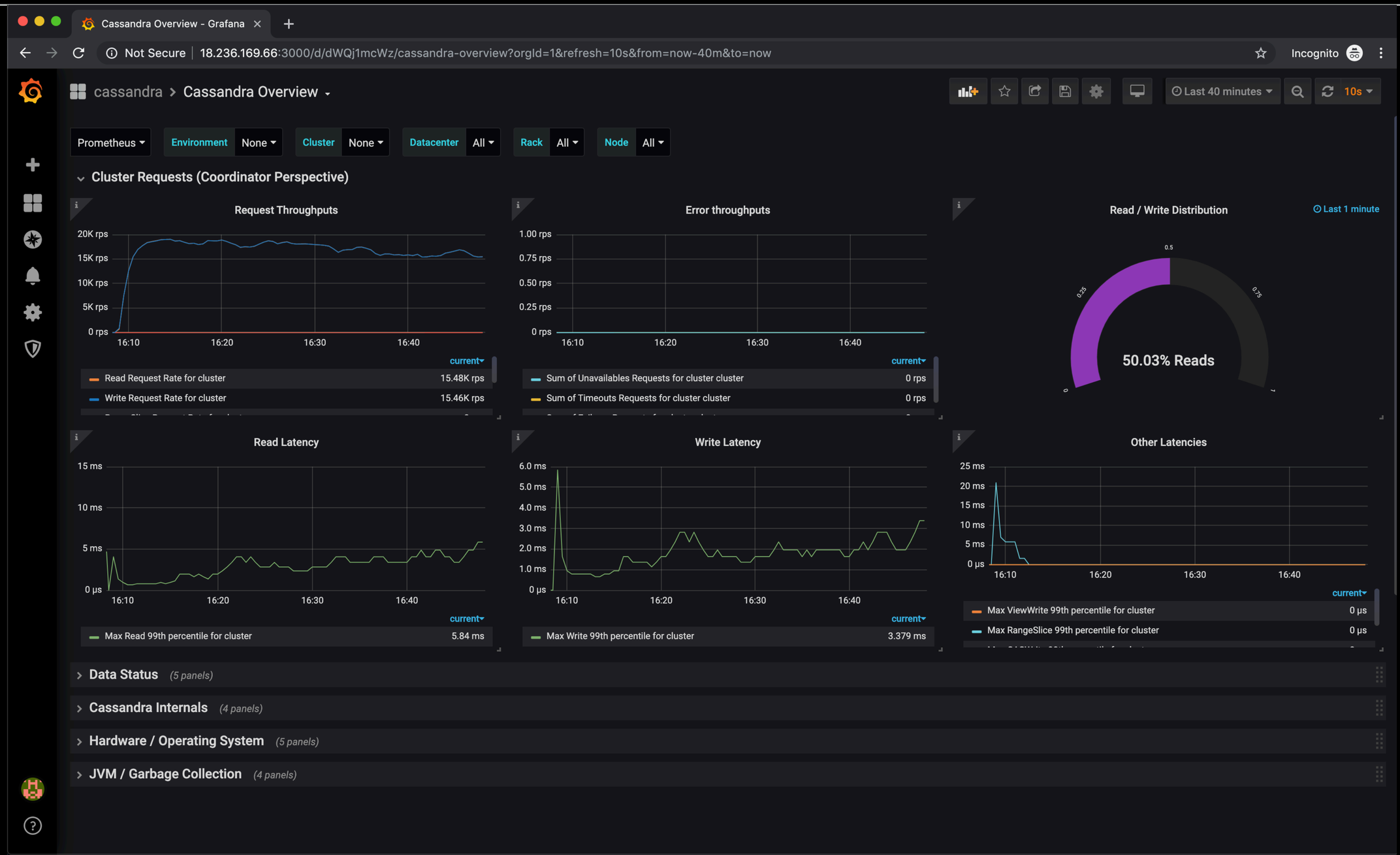
```
$ c-status
Datacenter: us-west-2
=====================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load        Tokens  Owns   Host ID   Rack
UN  172.31.5.26     160.33 KiB  16      59.9%  cffde3b3  2c
UN  172.31.8.56     208.83 KiB  16      45.2%  2c58e7a5  2c
UN  172.31.20.104   179.3 KiB   16      44.7%  4b66ad94  2b
UN  172.31.35.231   183.4 KiB   16      53.0%  be4648fe  2a
UN  172.31.37.101   156.2 KiB   16      46.5%  39b1f3b6  2a
UN  172.31.28.194   188.6 KiB   16      50.7%  57fc3bf7  2b
```

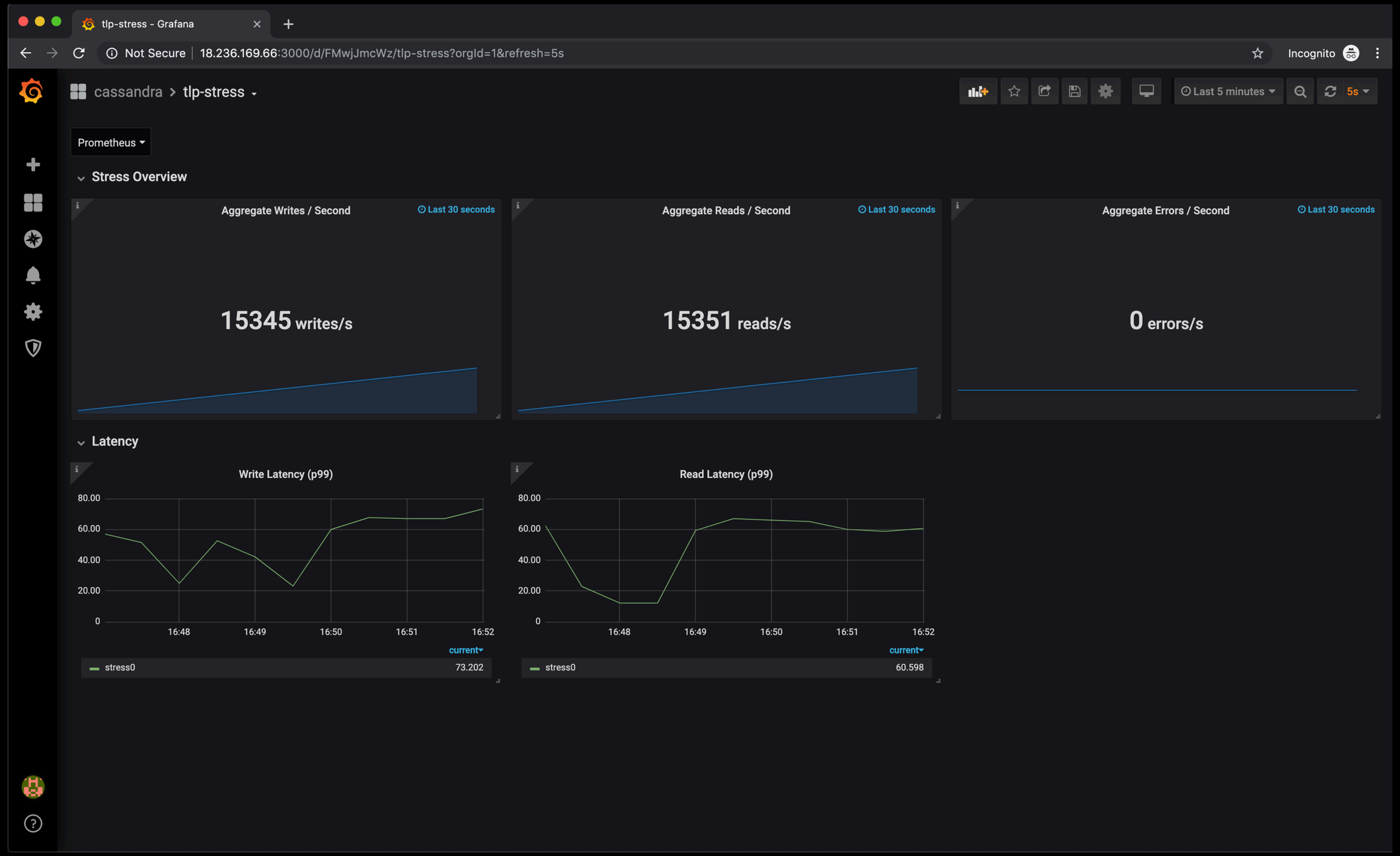
TLP-CLUSTER: METRICS MADE EASY



TLP-CLUSTER: METRICS MADE EASY

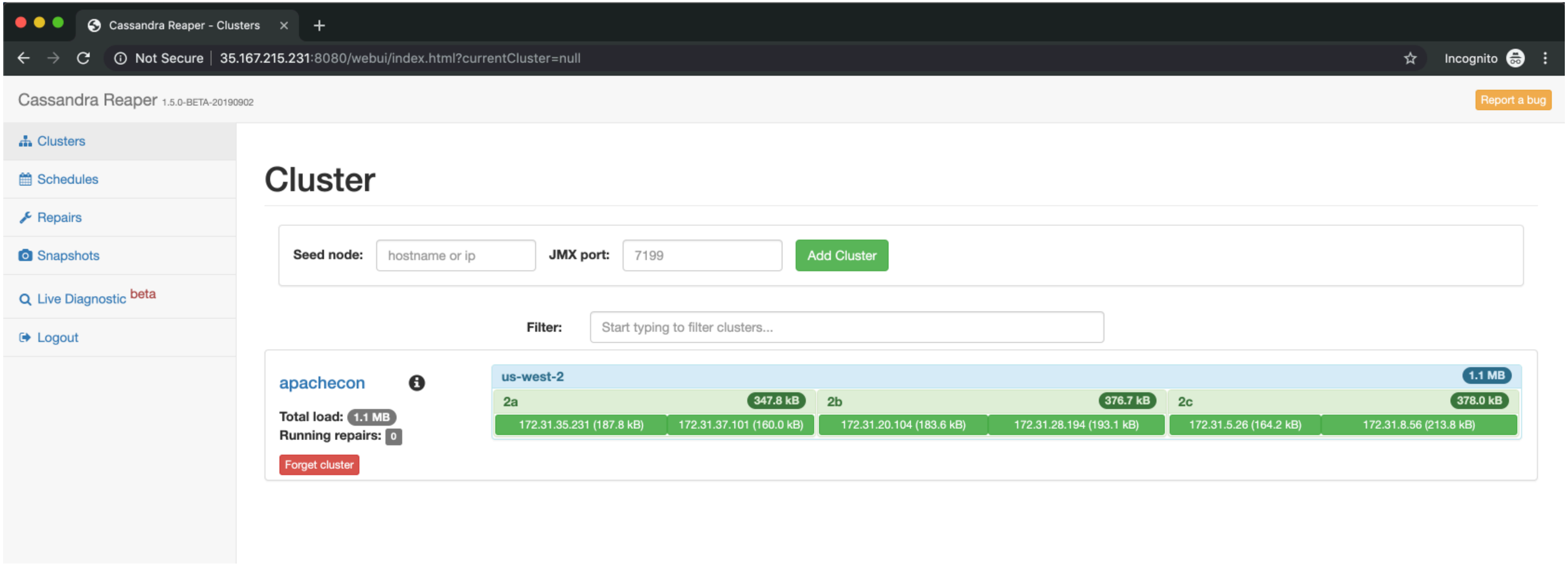



Cassandra Overview - Grafana
Not Secure | 18.236.169.66:3000/d/dWQj1mcWz/cassandra-overview?orgId=1&refresh=10s&from=now-40m&to=now
Incognito
cassandra > Cassandra Overview
Last 40 minutes
10s
Prometheus
Environment None
Cluster None
Datacenter All
Rack All
Node All
Cluster Requests (Coordinator Perspective)
Request Throughputs
20K rps
15K rps
10K rps
5K rps
0 rps
16:10
16:20
16:30
16:40
current
Read Request Rate for cluster 15.48K rps
Write Request Rate for cluster 15.46K rps
Error throughputs
1.00 rps
0.75 rps
0.50 rps
0.25 rps
0 rps
Sum of Unavailables Requests for cluster cluster 0 rps
Sum of Timeouts Requests for cluster cluster 0 rps
Read / Write Distribution
Last 1 minute
50.03% Reads
Read Latency
15 ms
10 ms
5 ms
0 µs
Max Read 99th percentile for cluster 5.84 ms
Write Latency
6.0 ms
5.0 ms
4.0 ms
3.0 ms
2.0 ms
1.0 ms
Max Write 99th percentile for cluster 3.379 ms
Other Latencies
25 ms
20 ms
15 ms
10 ms
5 ms
Max ViewWrite 99th percentile for cluster 0 µs
Max RangeSlice 99th percentile for cluster 0 µs
Data Status (5 panels)
Cassandra Internals (4 panels)
Hardware / Operating System (5 panels)
JVM / Garbage Collection (4 panels)


tlp-stress - Grafana
Not Secure | 18.236.169.66:3000/d/FMwjJmcWz/tlp-stress?orgId=1&refresh=5s
Incognito
cassandra > tlp-stress
Last 5 minutes
5s
Prometheus
Stress Overview
Aggregate Writes / Second
Last 30 seconds
15345 writes/s
Aggregate Reads / Second
Last 30 seconds
15351 reads/s
Aggregate Errors / Second
Last 30 seconds
0 errors/s
Latency
Write Latency (p99)
80.00
60.00
40.00
20.00
0
16:48
16:49
16:50
16:51
16:52
current
stress0
73.202
Read Latency (p99)
80.00
60.00
40.00
20.00
0
16:48
16:49
16:50
16:51
16:52
current
stress0
60.598

THE LAST PICKLE

TLP-CLUSTER: REPAIR MADE EASY



TLP-CLUSTER: 0.3 RELEASE

TLP-CLUSTER: CODE

- Source: https://github.com/thelastpickle/tlp-cluster
- Documentation: http://thelastpickle.com/tlp-cluster/
- Mailing List: tlp-dev-tools@googlegroups.com

TLP-CLUSTER: FUTURE WORK

- Project organisation
- Security
- Advanced topology
- More tests

THE LAST PICKLE

QUESTIONS?