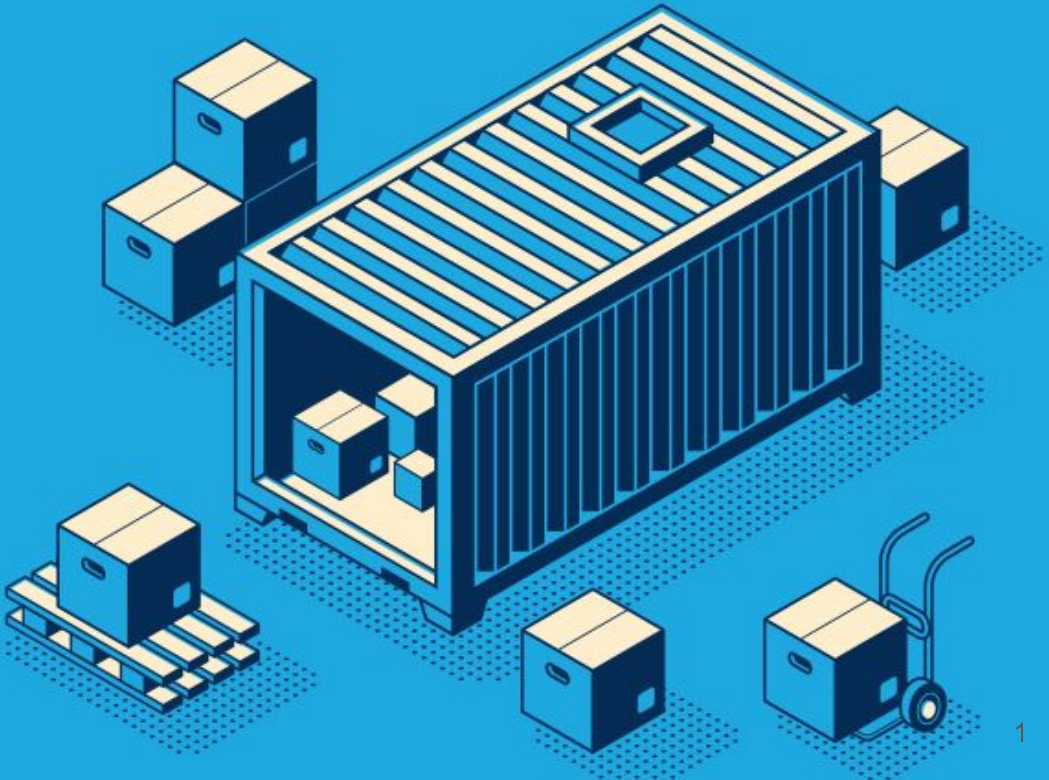
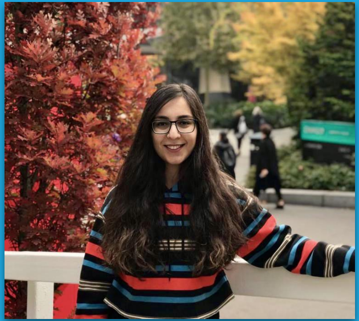
# Kubernetes

## Vertical Pod Autoscaler Operator



# The Team


Selen

Laxmi

Apoorva

Akshay

Sindhu

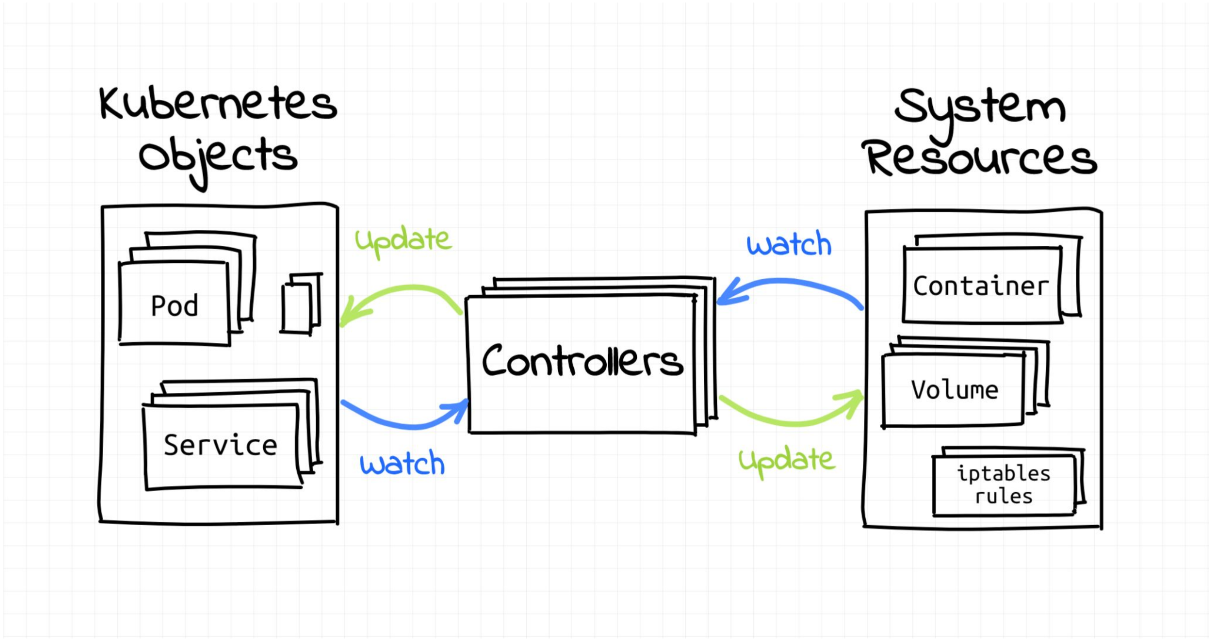
**Mentors (Redhat)**

Humair Khan & Anand Sanmukhani

## Goal of this project

- Avoid **manually** specifying pod resource requirements (CPU/Mem)
- Solution: Use **Vertical Pod Autoscaler** instead
- Question: Why is VPA operator not used **always**?

# Kubernetes 101: Controllers



# WHAT IS AN OPERATOR ?

A Kubernetes operator is an **application-specific** controller that extends the functionality of the Kubernetes API to **create**, **configure**, and **manage** instances of complex applications on behalf of a Kubernetes user.

## WHAT IS AN OPERATOR ?

It builds upon the basic Kubernetes resource and controller concepts, but includes **domain or application-specific knowledge** to automate the entire life cycle of the software it manages.

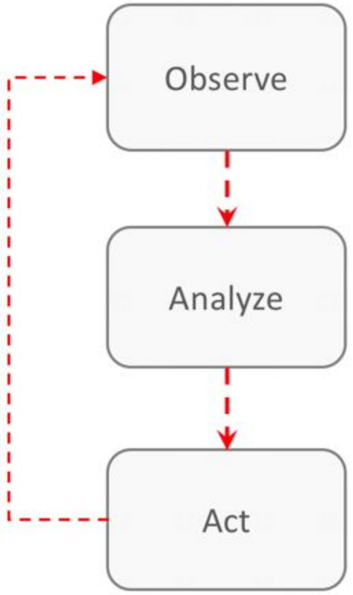
## Why is it a game-changer ?

These Controllers have direct access to Kubernetes API, which means they can

- **monitor** the cluster
- **change** pods/services
- **scale** up/down and call endpoints of the running applications

all according to custom rules written inside those controllers.

## How does it work ?



**Couchbase Cluster is running 2 pods:**

- CB-0001, version 5.0.1
- CB-0002, version 5.0.1

**Differences from desired config:**

- 3 Pods should be running instead of just 2

**How to get to desired config:**

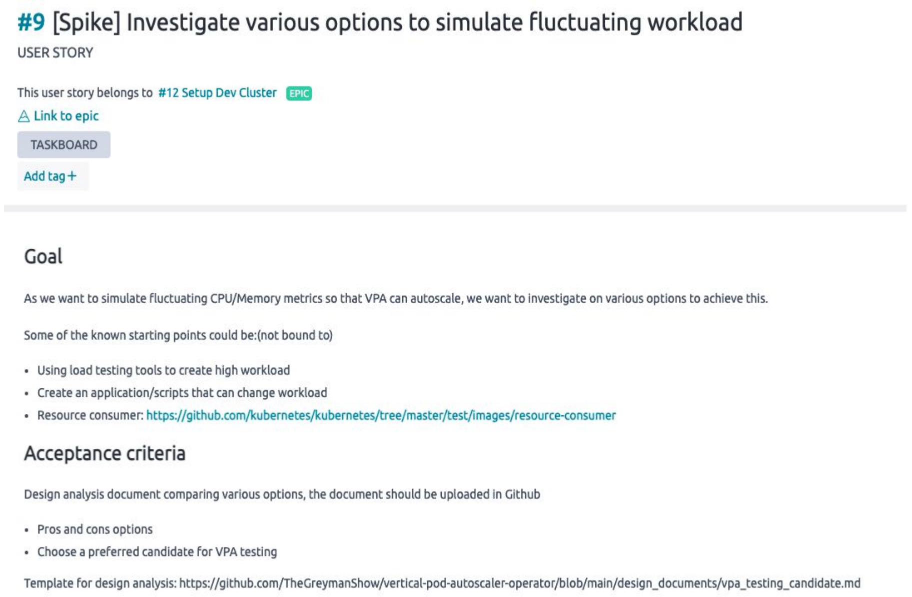
- Start new Pod
- Add new node to the cluster
- Trigger data rebalancing

## WHAT WE PLANNED TO DO IN SPRINT BOSTON

- Setup Openshift cluster on MassOpenCloud(MOC)
- Install Operators on cluster (VPA and Grafana)
- Get access to Grafana metrics from openshift cluster
- [Spike] Investigate options to simulate fluctuating workload
- Understand ArgoCD and GitOps setup in Operate-First
- Sprint demo video, presentation, quiz

## WHAT WE CHANGED

- Assign Product Owner to each Sprint
- Create user tasks with standard format
- Design documents for important decisions



**#9** [Spike] Investigate various options to simulate fluctuating workload

USER STORY

This user story belongs to #12 Setup Dev Cluster EPIC

Link to epic

TASKBOARD

Add tag +

Goal

As we want to simulate fluctuating CPU/Memory metrics so that VPA can autoscale, we want to investigate on various options to achieve this.

Some of the known starting points could be:(not bound to)

- Using load testing tools to create high workload
- Create an application/scripts that can change workload
- Resource consumer: https://github.com/kubernetes/kubernetes/tree/master/test/images/resource-consumer

Acceptance criteria

Design analysis document comparing various options, the document should be uploaded in Github

- Pros and cons options
- Choose a preferred candidate for VPA testing

Template for design analysis: https://github.com/TheGreymanShow/vertical-pod-autoscaler-operator/blob/main/design_documents/vpa_testing_candidate.md

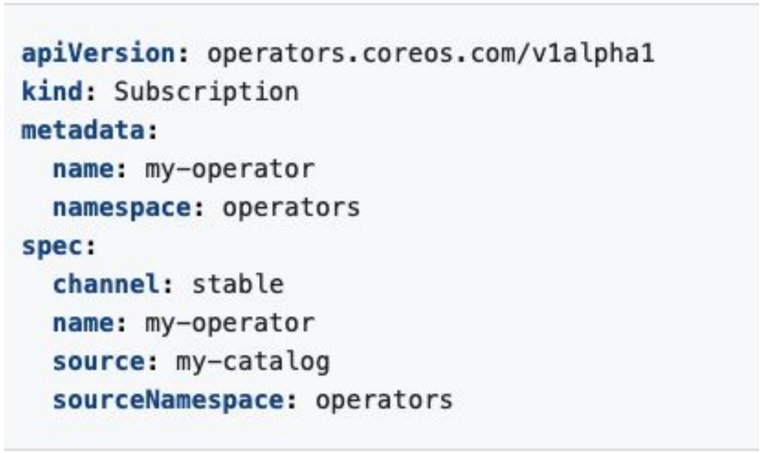
User story format

## Cluster setup

- Namespace assigned on MOC platform
    - Familiarized deployment using standard pet-clinic application
- To be installed:
    - VPA operator
    - Grafana operator
- Configuration for MOC managed clusters
    - https://github.com/CCI-MOC/moc-apps

## Subscription

- A Subscription represents an intention to install an operator.
- It is the CustomResource that relate an operator to a CatalogSource.



```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: my-operator
  namespace: operators
spec:
  channel: stable
  name: my-operator
  source: my-catalog
  sourceNamespace: operators
```

```
kubectl apply -f sub.yaml
```

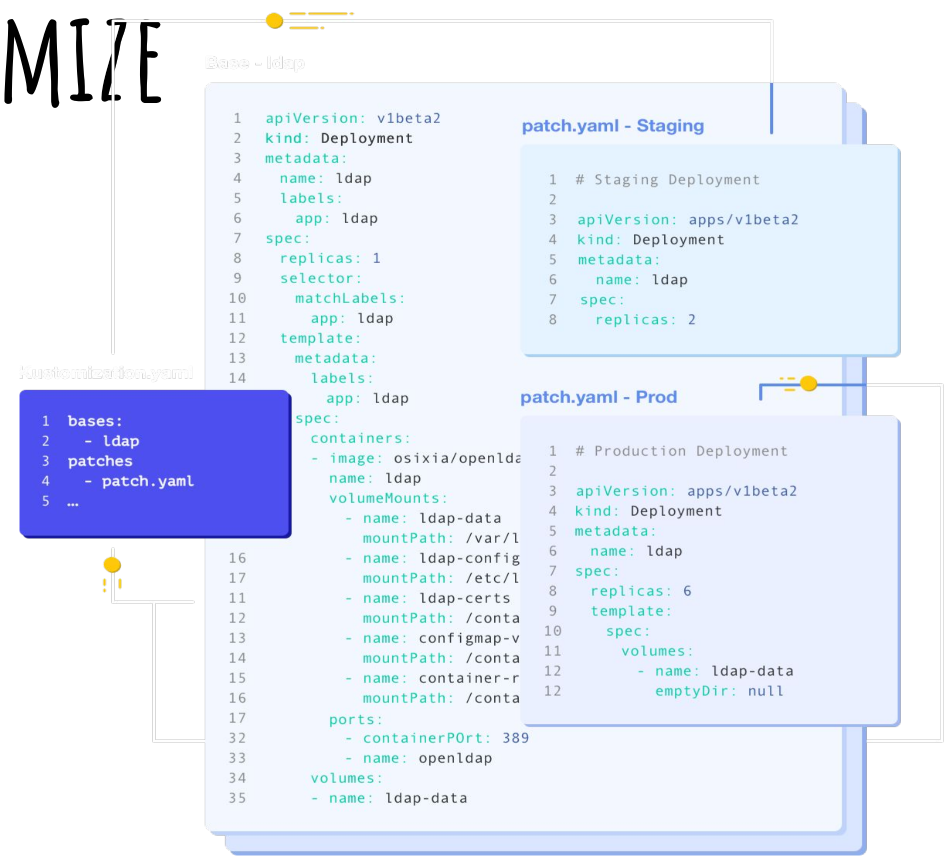
## K8s cluster config

```
kubectl apply -f application/deployment.yaml
```

# KUSTOMIZE

- Kubernetes native configuration management

- Kustomize is meant to build native Kubernetes manifests based on YAML, while leaving the original YAML in tact.



```yaml
apiVersion: v1beta2
kind: Deployment
metadata:
  name: ldap
  labels:
    app: ldap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ldap
  template:
    metadata:
      labels:
        app: ldap
    spec:
      containers:
      - image: osixia/openlda
        name: ldap
        volumeMounts:
          - name: ldap-data
            mountPath: /var/l
          - name: ldap-config
            mountPath: /etc/l
          - name: ldap-certs
            mountPath: /conta
          - name: configmap-v
            mountPath: /conta
          - name: container-r
            mountPath: /conta
        ports:
          - containerPOrt: 389
          - name: openldap
      volumes:
      - name: ldap-data
```

```yaml
bases:
  - ldap
patches
  - patch.yaml
…
```

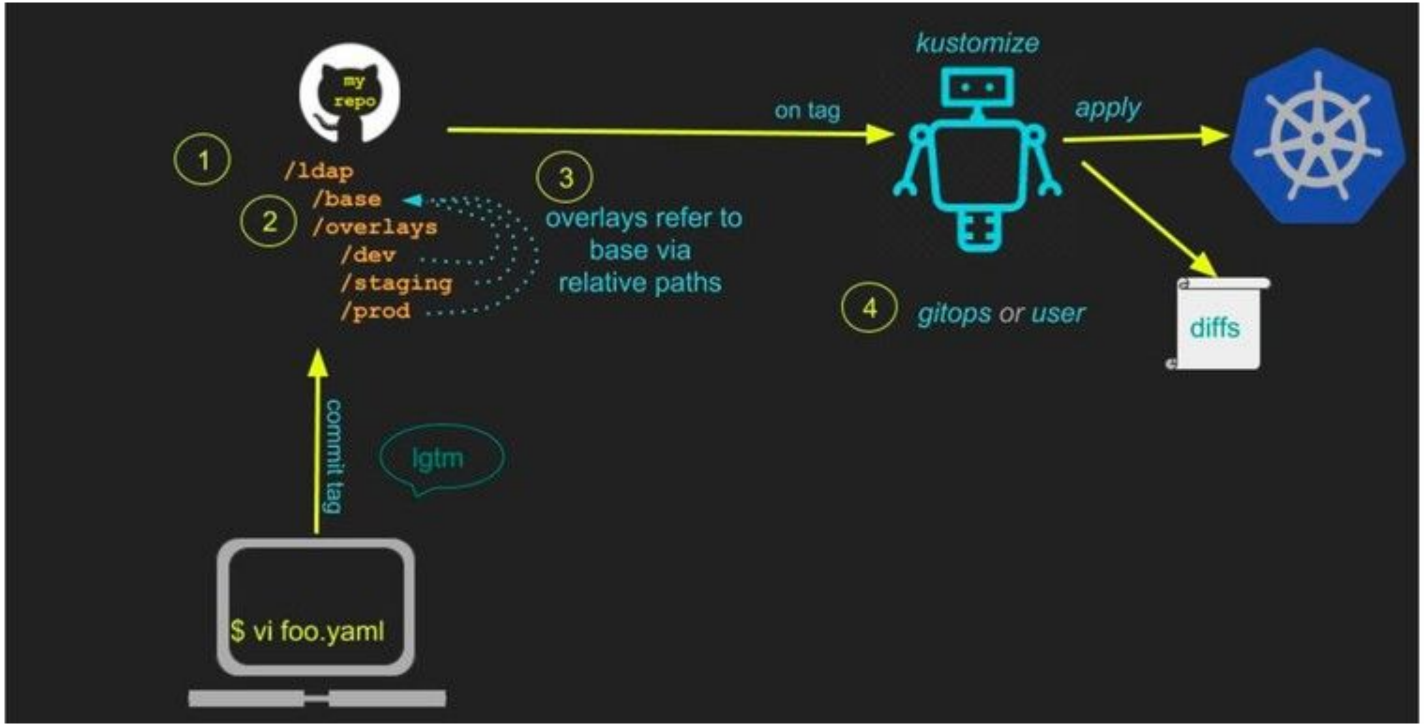
```yaml
# Staging Deployment

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: ldap
spec:
  replicas: 2
```

```yaml
# Production Deployment

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: ldap
spec:
  replicas: 6
  template:
    spec:
      volumes:
        - name: ldap-data
          emptyDir: null
```

## KUStomize



```
kubectl apply -k ./k8s/kustomize/environments/production
```

## Choosing Candidate for Vpa testing (Spike)

## Option 1 - Load Testing tools

SPEEDSCALE

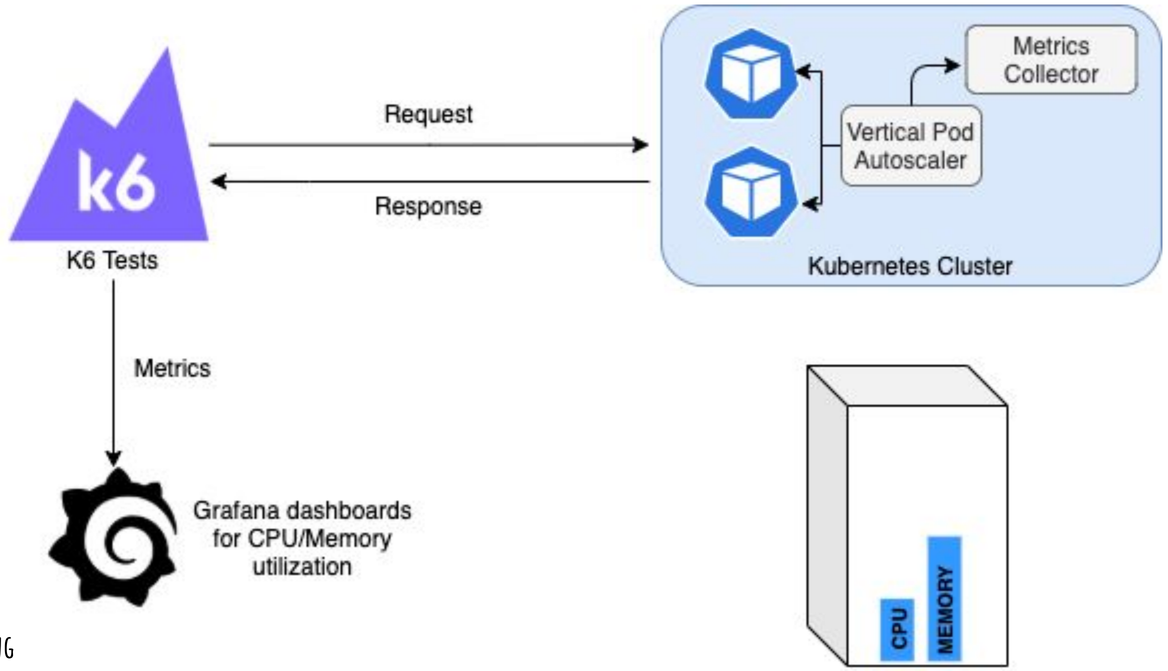
ReadyAPI

k6

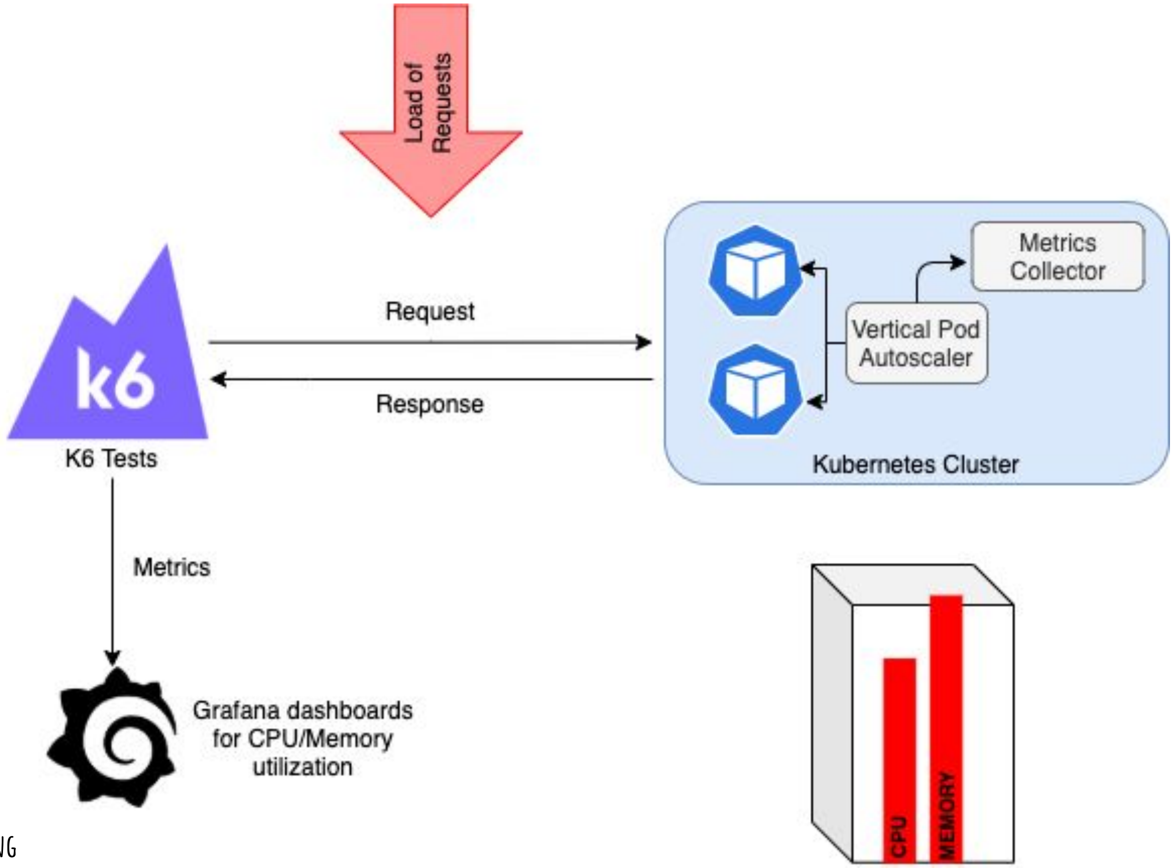
APACHE JMeter™

Gatling

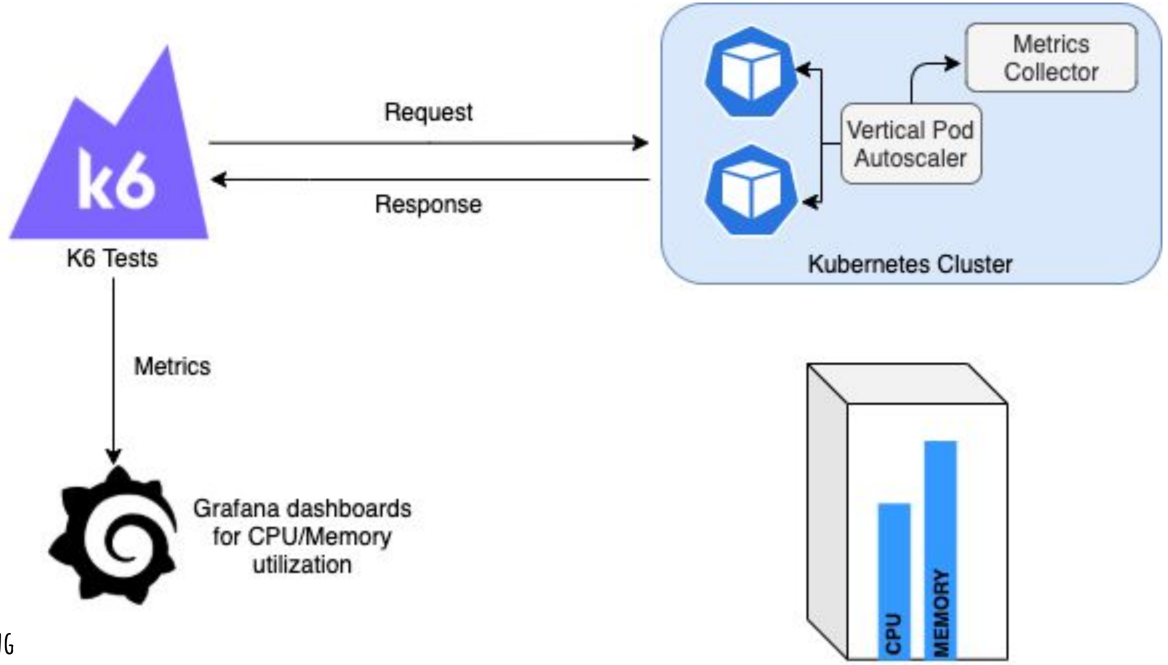
## OVERall test Scenario Design



## OVERall test Scenario Design



# OVERall test Scenario Design



# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | | | |

It is not a trivial task to control CPU consumption per request

## Pros & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | | |

Easily configurable that how much memory will be consumed

# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | ✖ | |

Needs effort to implement & deploy a web server

## PROS & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | ✖ | ✖ |

Easy to integrate with Grafana & K8s via CLI tool but still requires to dealt with other components

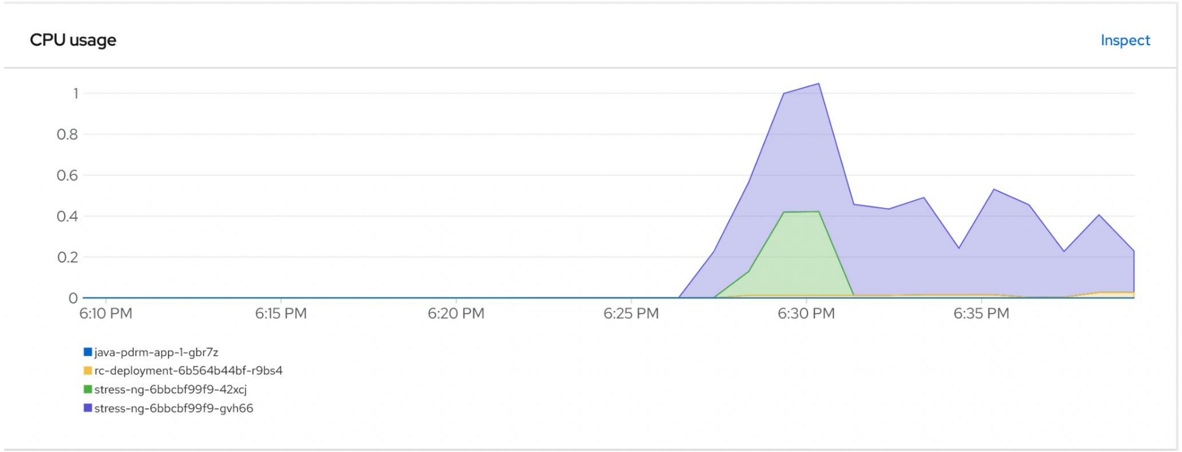
## OPTION 2 - STRESS-NG

- Open source command line tool
- Has multiple stress tests (stressor)

- How to use stress-ng in openshift cluster?
  - Wrapper py script around stress-ng
  - Create a docker image of the wrapper script
  - Deploy docker image in openshift cluster

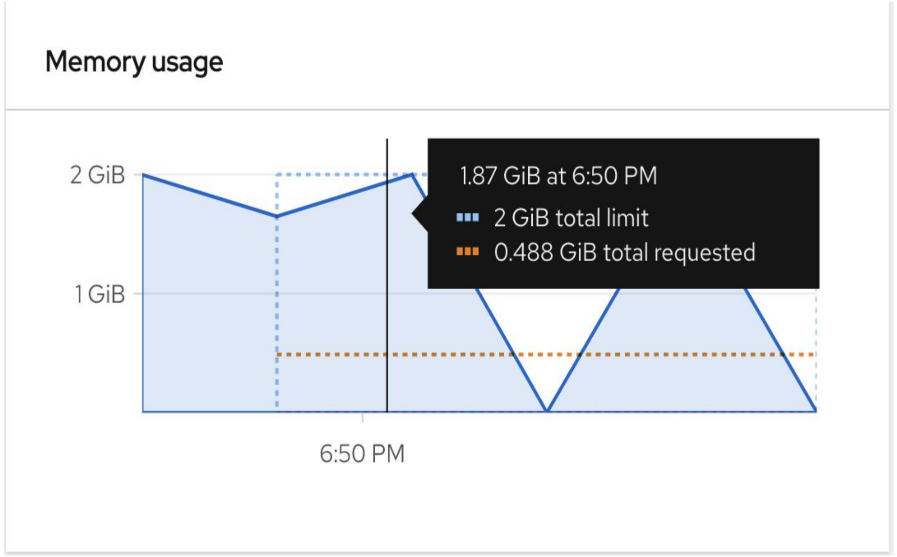
# Stress-ng Example for CPU load

```
docker run -it --rm stress-ng
--cpu 4
--timeout 60s
--metrics-brief
```



# Stress-ng Example for memory load

```
docker run stress-ng
--vm 8
--vm-bytes 80%
--timeout 60s
```



## Pros & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | | | |

Not very trivial, stressor may vary between kernels

# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | | |

Yes, easy to specify the % of memory workload to be consumed

# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | ✖ | |

Needs effort to implement wrapper script and creating docker image

# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ? | ✓ | ✖ | ✖ |

It is a CLI, not designed for kubernetes.
Wrapper must be made suitable for kubernetes.

## option 3 - k8s Resource consumer

- Primarily developed to test k8s autoscaling.
- Written in Golang and uses GoRoutines
- Use Cases:
  - ➔ Cluster size autoscaling
  - ➔ Horizontal Pod Autoscaler(HPA)
  - ➔ Vertical Pod Autoscaler(VPA)
- Starts an HTTP server in the consuming container and handles the incoming POST requests.

## Option 3 - K8s Resource consumer

### Setup:

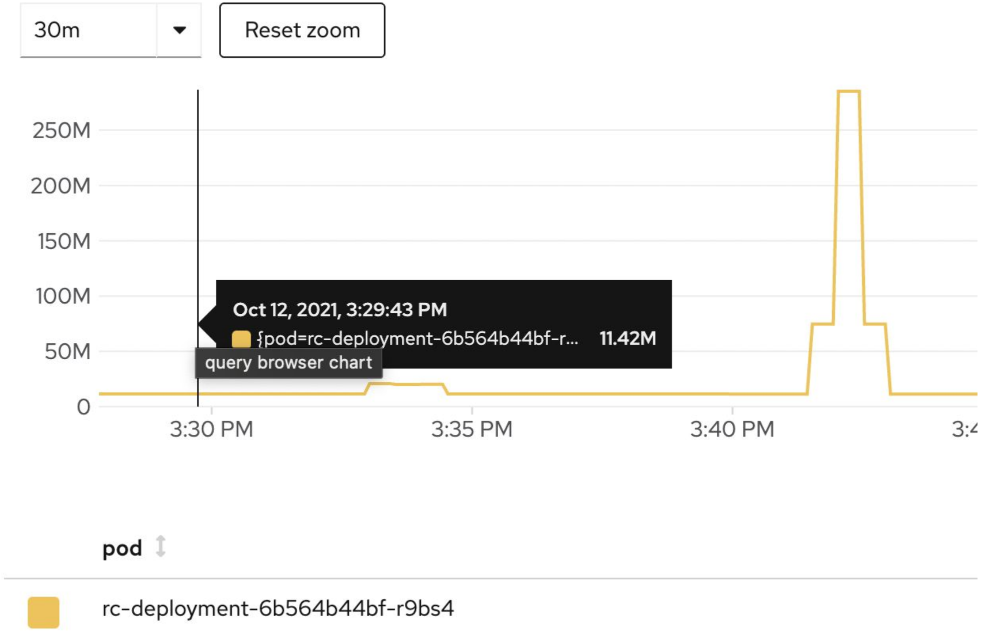
Easy to setup using an existing docker image that can be found in Google Container Registry. It can be easily deployed in a container using *kubectl run*

### Usage:

- Consumes CPU and Memory resources via API.
- HTTP requests to specify CPU and Memory consumption
    - /ConsumeCPU
    - /ConsumeMem

## Example command for Memory Consumption

```
curl --data
"megabytes=250&durationSec
=10"
http://<EXTERNAL_IP>/Cons
umeMem
```

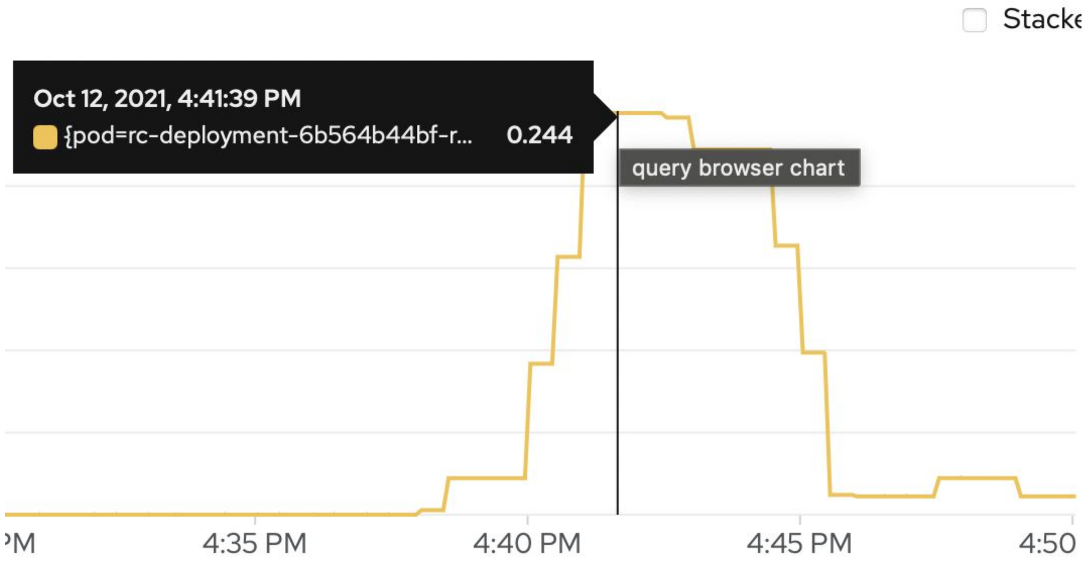


Oct 12, 2021, 3:29:43 PM
{pod=rc-deployment-6b564b44bf-r... 11.42M

rc-deployment-6b564b44bf-r9bs4

## Example command for CPU Consumption

```
curl --data
"millicores=250&durationSec=
100"
http://<EXTERNAL_IP>/Cons
umeCPU
```


Oct 12, 2021, 4:41:39 PM
🟨 {pod=rc-deployment-6b564b44bf-r... 0.244

## Why resource consumer?

- Existing Docker Image.
- Comes with a backend API that handles HTTP requests.
- Controlled consumption of CPU and Memory for a specified duration.
- Easy communication to the consuming container using API calls.

# Pros & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ✓ | | | |

Easy with curl commands

# Pros & Cons

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ✓ | ✓ | | |

Easy with curl commands

## Pros & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ✓ | ✓ | ✓ | |

Easy with existing docker image

# Pros & CONS

| Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ |

Primarily developed to test k8s

## WINNER

## K8s resource consumer

# COMPARISON

| | Ability to Control CPU | Ability to Control Memory Workload | Ease of Setup/Development | Compatibility |
|---|---|---|---|---|
| Load testing tools | ? | ✓ | ✖ | ✖ |
| Scripts/Library | ? | ✓ | ✖ | ✖ |
| Resource Consumer | ✓ | ✓ | ✓ | ✓ |

# WHAT WE ACCOMPLISHED IN THE SPRINT BOSTON

- Setup Openshift cluster on MassOpenCloud(MOC) ✓
- Install Operators on cluster (VPA and Grafana) ⏳
- Get access to Grafana metrics from openshift cluster ⏳
- [Spike] Investigate options to simulate fluctuating workload ✓
- Understand ArgoCD and GitOps setup in Operate-First ✓
- Sprint demo video, presentation, quiz ✓

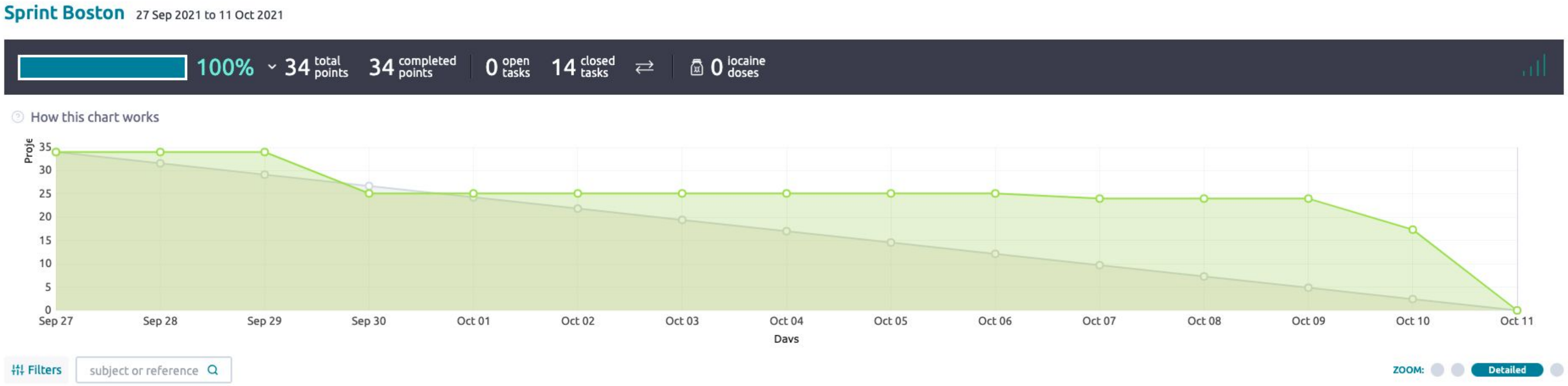
## WHAT WE LEARNED

- For install K8s operator task:
  - turned out to be more effort than installing from OLM.
  - So we need better story estimations.

## Plan for sprint chicago

- Complete operator installation
- Understand the repository pattern required for GitOps
- Install ArgoCD operator and setup Web UI
- Create scheduled workload to show VPA

# Burndown chart



Sprint Boston 27 Sep 2021 to 11 Oct 2021

100% 34 total points 34 completed points 0 open tasks 14 closed tasks 0 iocaine doses