**Autotune**

Experiment Manager

# Overview

Experiment manager (EM) enables Autotune to run multiple unique experiments having a variety of deployment configurations along with various metrics collection queries which helps in getting accurate configuration to improve the performance of target application.

# User story and User experience

TBD

# Goals

1. Run multiple experiments per deployment in parallel.
2. Execute different unique trials available inside Experiment.
3. Each trial consists of a Number of iteration having Warm Up cycle and measurement cycle to collect specified metrics.
4. Summarize collected metrics per experiment.
5. Post the summarized result back to the Analyser(HPOasS) for more trials.

# Specifications

TBD

# Milestones

## Architecture Block Diagram.

There are Six main building blocks of EM’s Architecture.

* 1. Parallelization Engine.

This block basically helps in iterating with User or Analyser(HPOasS) for posting results and main goal is to have a quick response to user .This engine will specify following

* Autotune thread pool count
* Async Servlet
* TaskManger which is responsible for creating a queue and executors with another set of thread pools.
  1. Performance Abstraction.

Detail explanation of Task Manager Abstraction which helps in creating task queue and max thread pool count.This abstraction will have following capabilities

* Spawn multiple interactions for a given trial.
* Get repeatable results,compare and check for consistency.
  1. Target Handler.

This abstraction will communicate either with kubernetes or non kubernetes environments but at present Autotune supports only kubernetes.

Kubernetes abstraction should have following features

* + - * Independent of cloud infrastructure like (AWS,AKS,Google cloud, Openshift etc)
      * Should be a common module and should be used across all other autotune modules and loosely coupled.
      * Should have apis to read/write kubernetes components like

Pods , Containers , Services , Deployments etc

* 1. Metric collection Handler.

TBD

* 1. Summarize the result Handler.

TBD

* 1. Log and Exception Handler.

TBD

## Use cases

TBD

## API’s

TBD

## UML

TBD

## Development , Review,

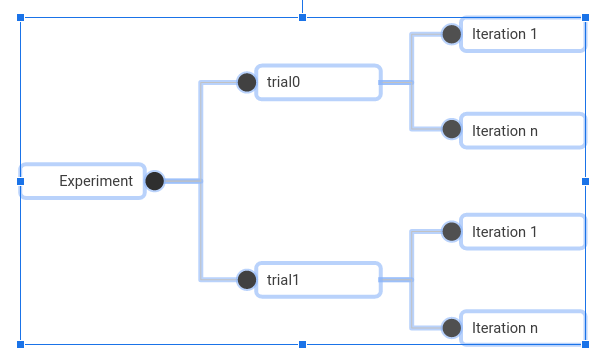
## Testing

TBD

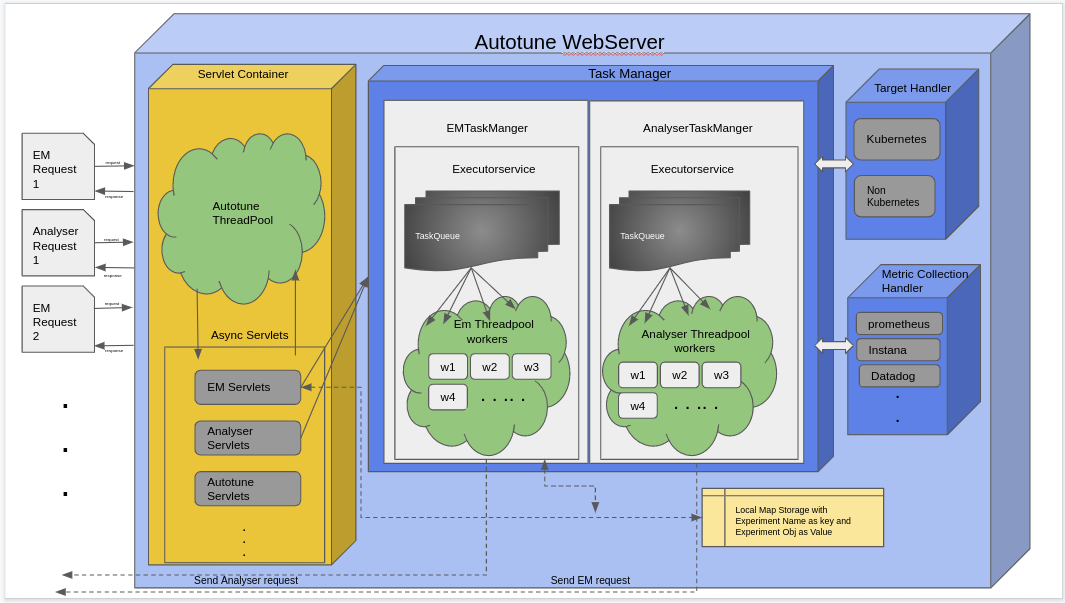
# 

# Architecture Block Diagram

Typically EM receives Configuration from Analyser or from User in the form of JSON request . Json consists of a unique experiment name having multiple trials ranging between 1-n. Each individual trial is differentiated with the help of a unique trial number per experiment. And each trial consists of configuration related to container,runtime etc.



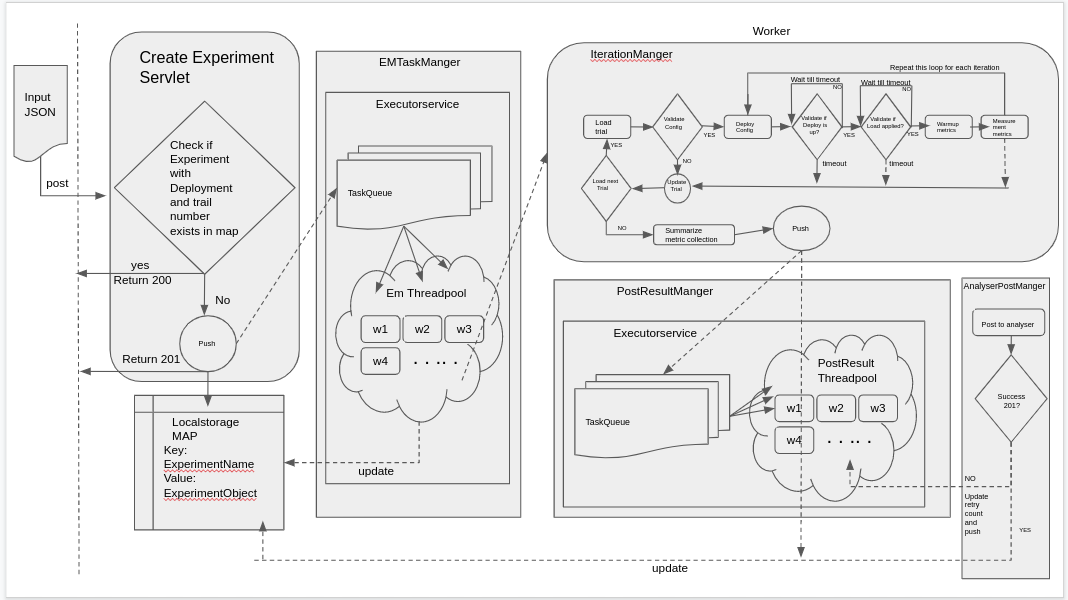
## EM Block Diagram



As shown in the diagram Asynchronous EM servlets get requests in the form of JSON provided by the user or Analyser module. EM servlet will convert JSON into Object and push it into the task queue. Here EM servlets make use of Autotune threadpool + task queue to achieve parallelization and performance goals of EM. Waiting time for the user to get a response is almost nil and tasks are pushed to queue for further processing. Users will have another API ‘checkExperimentSummaryList’ to view the status of trials.

There will be a scheduler process running for every configured time gap to check if any task is inside the queue. If a task is found then the task gets executed using Java Threadpool Executors service using EM threadpool so that multiple experiments can run in parallel.

## EM Architecture



Following Block diagrams will give more information about individual components.

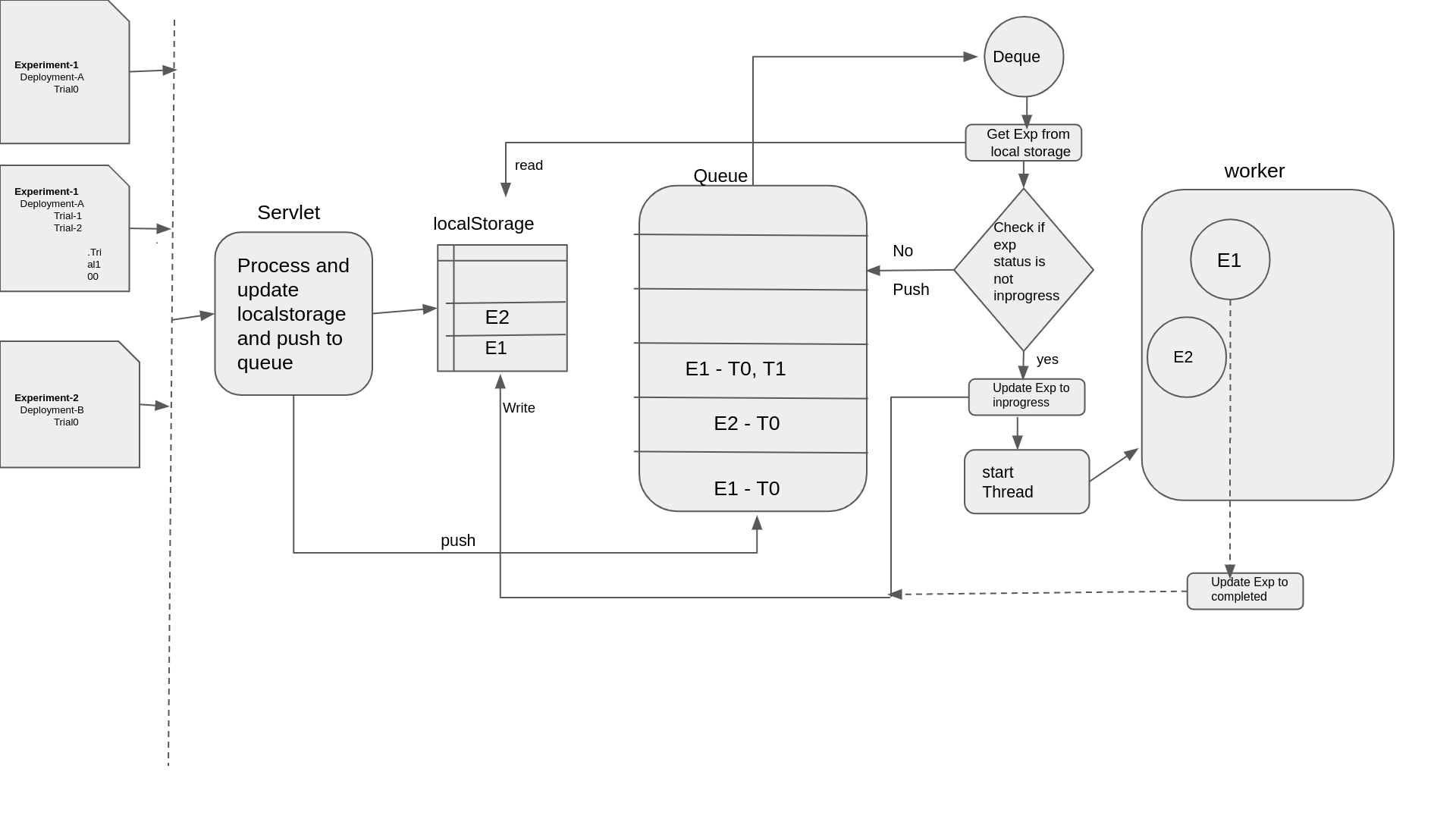
### Create Experiment Servlet

This is an asynchronous servlet meant to receive Experiment trials JSON and perform following granular task

* Convert JSON into ExperimentTrail object. if any exception then return 400.
* Usel service class to check if record exists in existing map using Experiment name.
  + If a record exists check if trail number exists if not then add trial details.
  + If trial number exists check for “retry” flag and update flag to True/False accordingly.
  + If ExperimentName does not exist then create a record.
* If changes are detected then push the ExperimentTrail object into the Task queue and return 201 for new record and return 200 for existing record.

### Dequeuing using Scheduler

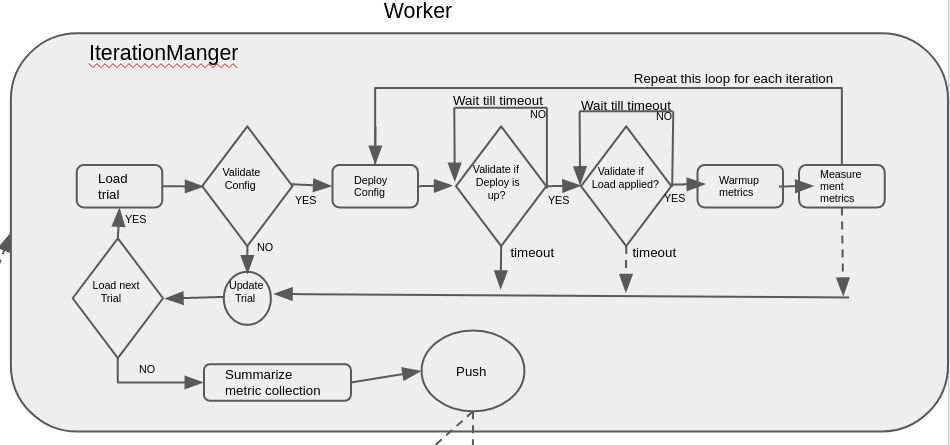
ScheduledThreadPoolExecutor class is used to check if any new task added into queue after specified given delay,If new task determined then it deque as shown in diagram and get status from Localstorage concurrenthashmap. If status is null then update status to new and handover to EM Taskmanger’s threadpool executors to execute the task and update status to “in progress”. If status says “in progress” then reject the task stating Experiment already in progress.



### EM TaskManger

This is an extension of task manager which consists of a queue and threadpool executor. LinkedBlockingqueue is used to store tasks and it maintains order. Maximum number of threads inside the pool will be specified to avoid out of memory issues.IterationManager(Worker) get spawned for each experiments.In case of task rejection if the executor becomes overwhelmed with tasks due to Maximum pool size and defined queue capacity hit their limits or executors are shutting down.Catch those task and report in ExperimentSummary page.

### IterationManager



IterationManager basically a worker that get spawned from threadpoolexecutor, It consists of following stage

#### Loadtrail

This class loads a trails from Experimenttrial object and check if the trial status is new and change the status to inprogress. If trail status is completed then it moves to next trial object.

For each trial run following sub stages

##### ValidateTrial Config

TBD. Raise CustomException if any exception.

For each iteration run following sub tasks

##### Deploy Trial Config

TBD

##### Validate deployed config

TBD

* Validate if load is applied

TBD

##### Collect Warmup metrics

TBD

##### Collect measurement metrics

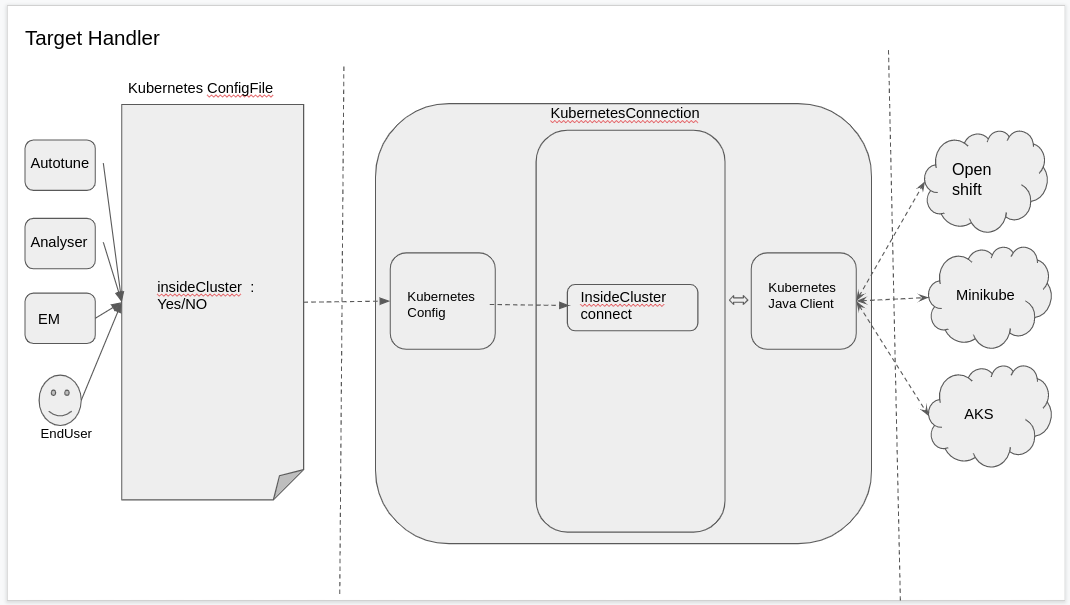
TBD

#### Summarize trial

TBD

## Target Handler Architecture

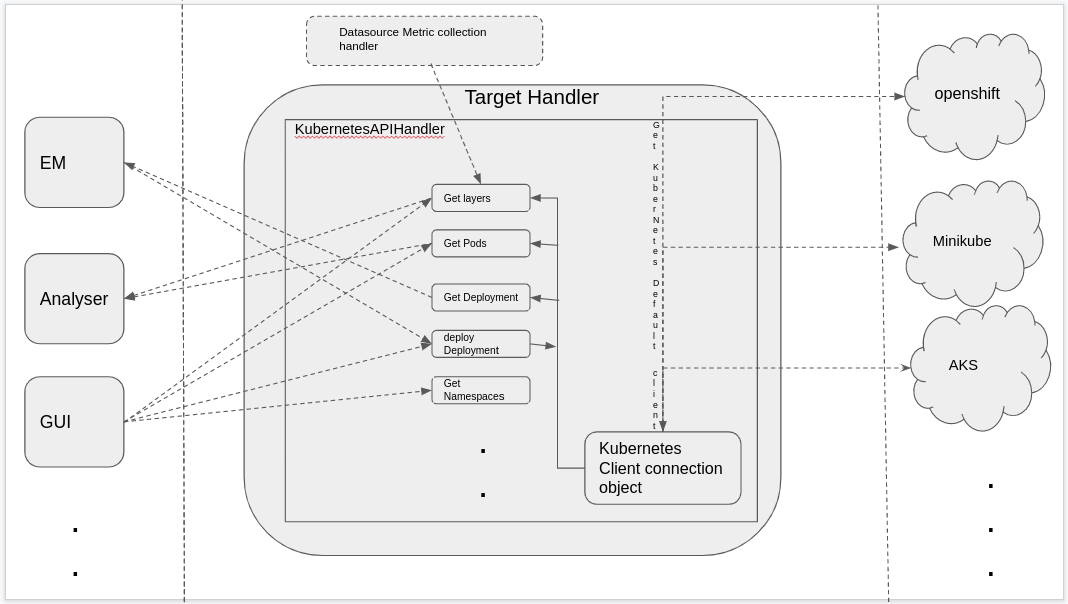
Target is a place where EM deploys and runs Trials suggested by Analyser. And target could be kubernetes or non kubernetes environment. Currently EM supports only the kubernetes environment.



Target Handler is a common connection interface designed to be used across all autotune modules which requires an interaction with kubernetes. Target handler uses Kubernetes java client library (https://github.com/kubernetes-client/java)to interact with kubernetes api. By default all autotune modules will interact with kubernetes client inside cluster and hence insidecluster flag is set to ‘TRUE’ and KuberentesConfig object is used for same, In future if there is a requirement where autotune modules connect to kubernetes outside cluster we can make use of Config object.

KubernetesApi handler

This is a common wrapper class to access some of the common functions like getPods,getDeployments etc.



UML

# 