K8s Nemesis

Requirements and analysis model

Product description

Product Description: A cloud-native service designed to enhance application scalability and performance in Kubernetes (K8s) environments, specifically targeting machine learning (ML) applications with unique scaling and resource requirements.

Team: Sergey Lokhmatikov, Andrey Tamplon, Roman Kuzmenko, Vasiliy Lopatkin

Repo: https://github.com/Lokhmat/k8s_nemesis

Report: https://github.com/Lokhmat/k8s_nemesis/blob/main/final_slides.pdf

MLOps engineer

Background: Focuses on deploying, monitoring, and maintaining ML models in production. He's skilled in cloud platforms, CI/CD pipelines, and container tools like Docker and Kubernetes.

Needs: Tools that help with continuous integration and deployment of ML models make it easier to quickly update and improve models without interrupting current services.

Challenges: Making sure models are reliable, keeping consistency across environments, and handling complex ML pipelines and dependencies.

Goal: Automate deployment and monitoring for optimal model performance.

DevOps engineer

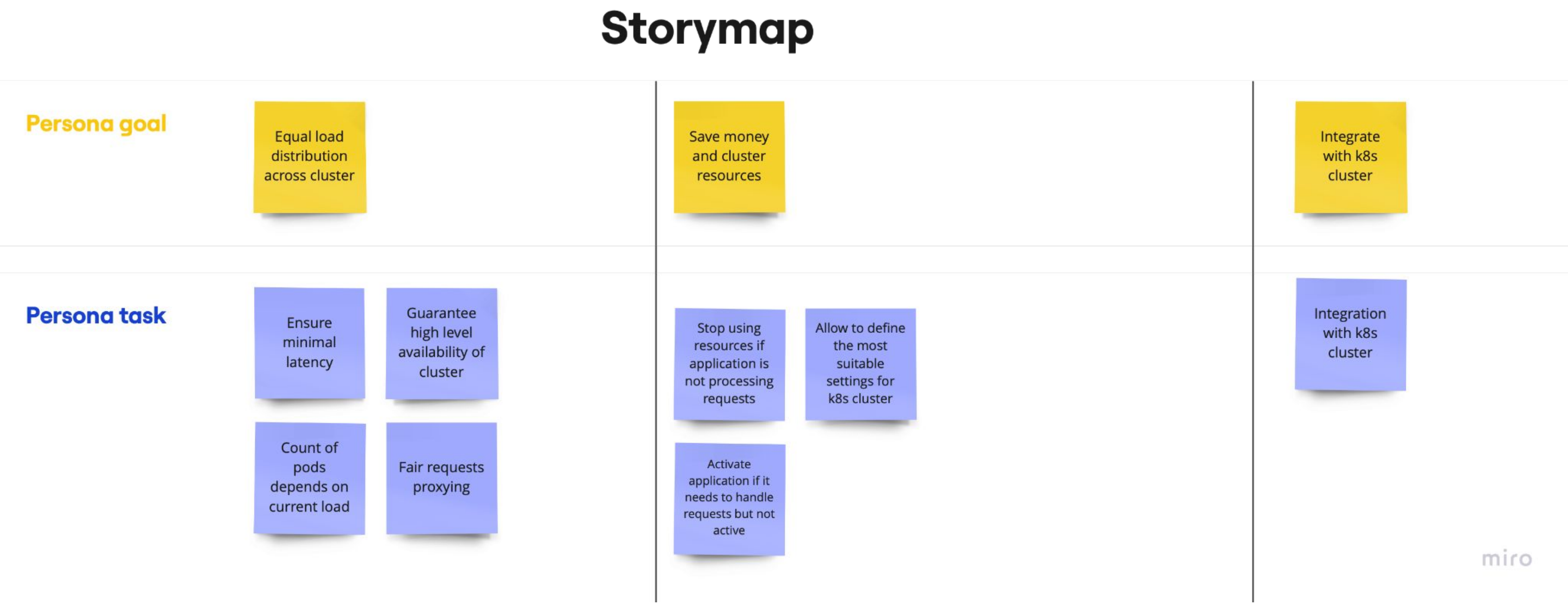
Background: Engineer with a deep understanding of a distributed system. High level expertise of working with Kubernetes, continuous integration and deployment. The one who is responsible for cloud architecture in company

Needs: Elegant and customizable tool for managing k8_s cluster. Tool that can reduce cost of maintenance of a large scale architecture while ensuring high SLA and good performance in a complicated systems.

Challenges: Balancing infrastructure maintenance cost vs service uptime. Managing distributed cluster, and event driven scaling based on customizable set of rules and metrics. Ensuring security and stability of a system.

Goal: Deployment and maintenance of a reliable distributed system. Increasing usage efficiency of a current organisation's cloud architecture.

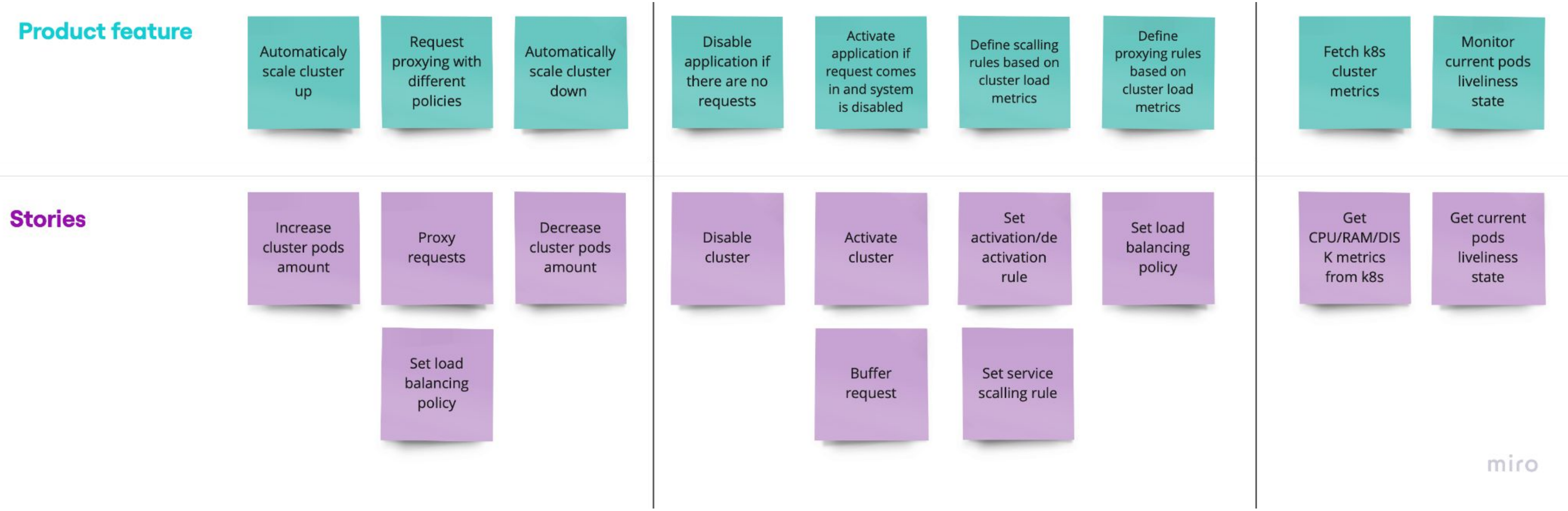
Story Map



Storymap

Persona goal	Equal load distribution across cluster	Save money and cluster resources	Integrate with k8s cluster
Persona task	Ensure minimal latency; Guarantee high level availability of cluster; Count of pods depends on current load; Fair requests proxying	Stop using resources if application is not processing requests; Allow to define the most suitable settings for k8s cluster; Activate application if it needs to handle requests but not active	Integration with k8s cluster

Story Map



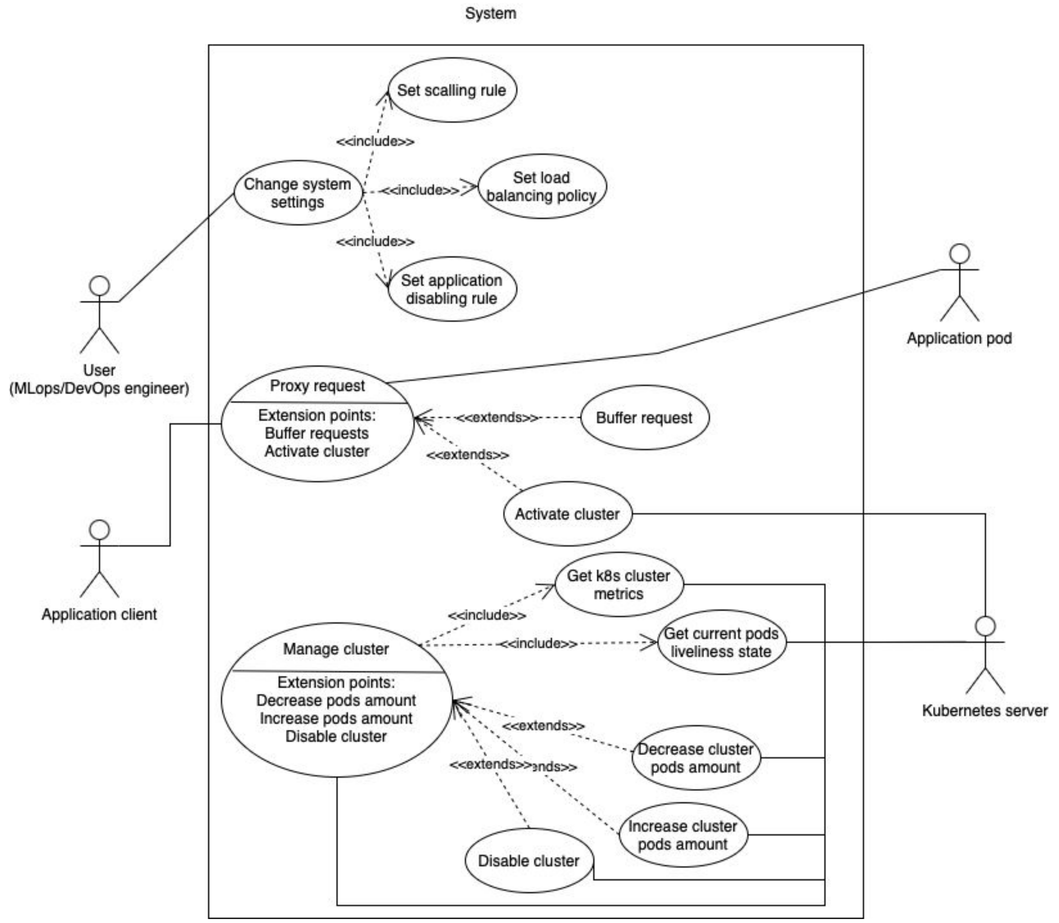
Product feature	Automaticaly scale cluster up	Request proxying with different policies	Automatically scale cluster down	Disable application if there are no requests	Activate application if request comes in and system is disabled	Define scalling rules based on cluster load metrics	Define proxying rules based on cluster load metrics	Fetch k8s cluster metrics	Monitor current pods liveliness state
Stories	Increase cluster pods amount	Proxy requests	Decrease cluster pods amount	Disable cluster	Activate cluster	Set activation/deactivation rule	Set load balancing policy	Get CPU/RAM/DISK metrics from k8s	Get current pods liveliness state
		Set load balancing policy			Buffer request	Set service scalling rule			

miro

Use case diagram

Textual use case scenarios:

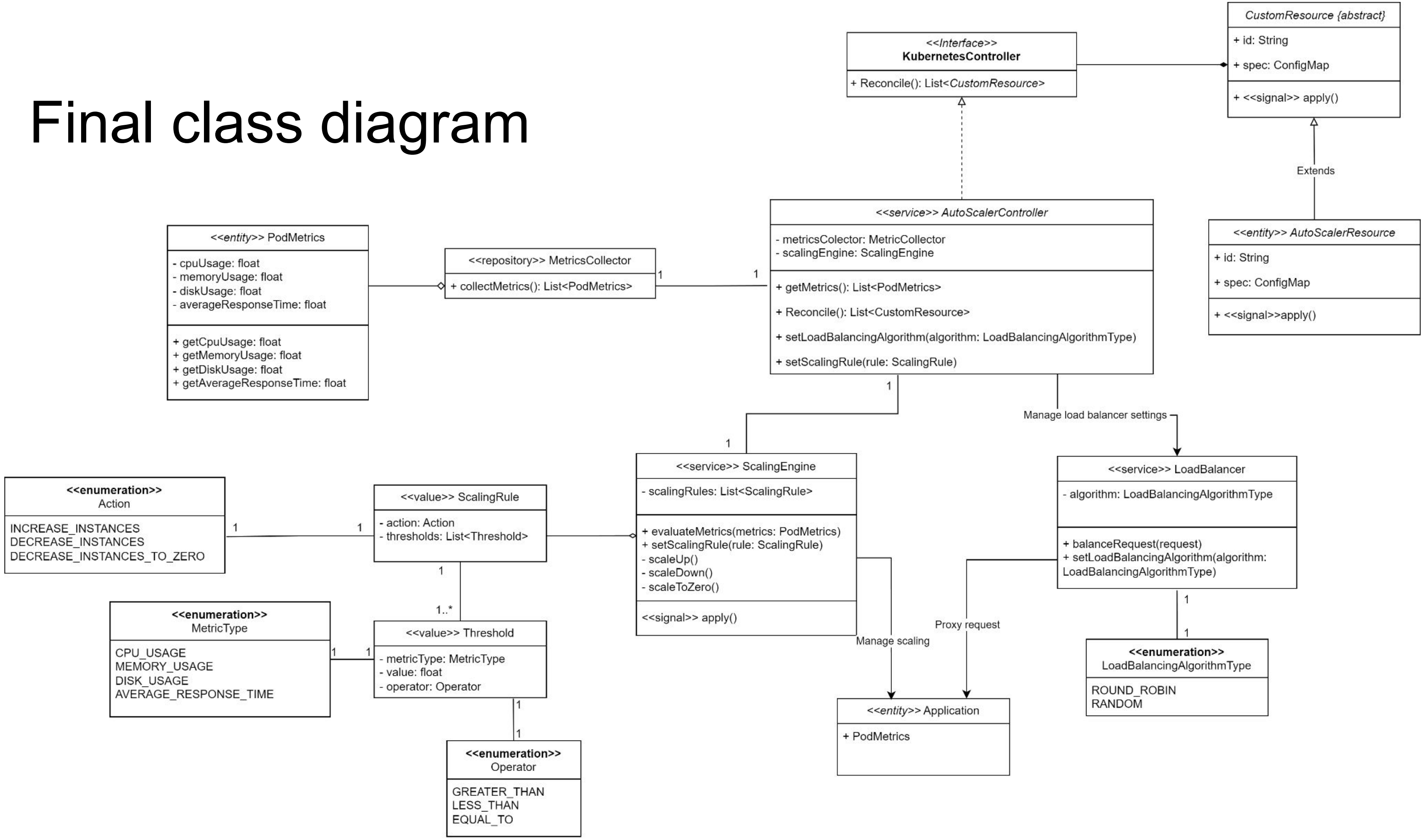
https://github.com/Lokhmat/k8s_nemesis/blob/main/final_task_materials/textual_use_cases.md



Interaction analysis

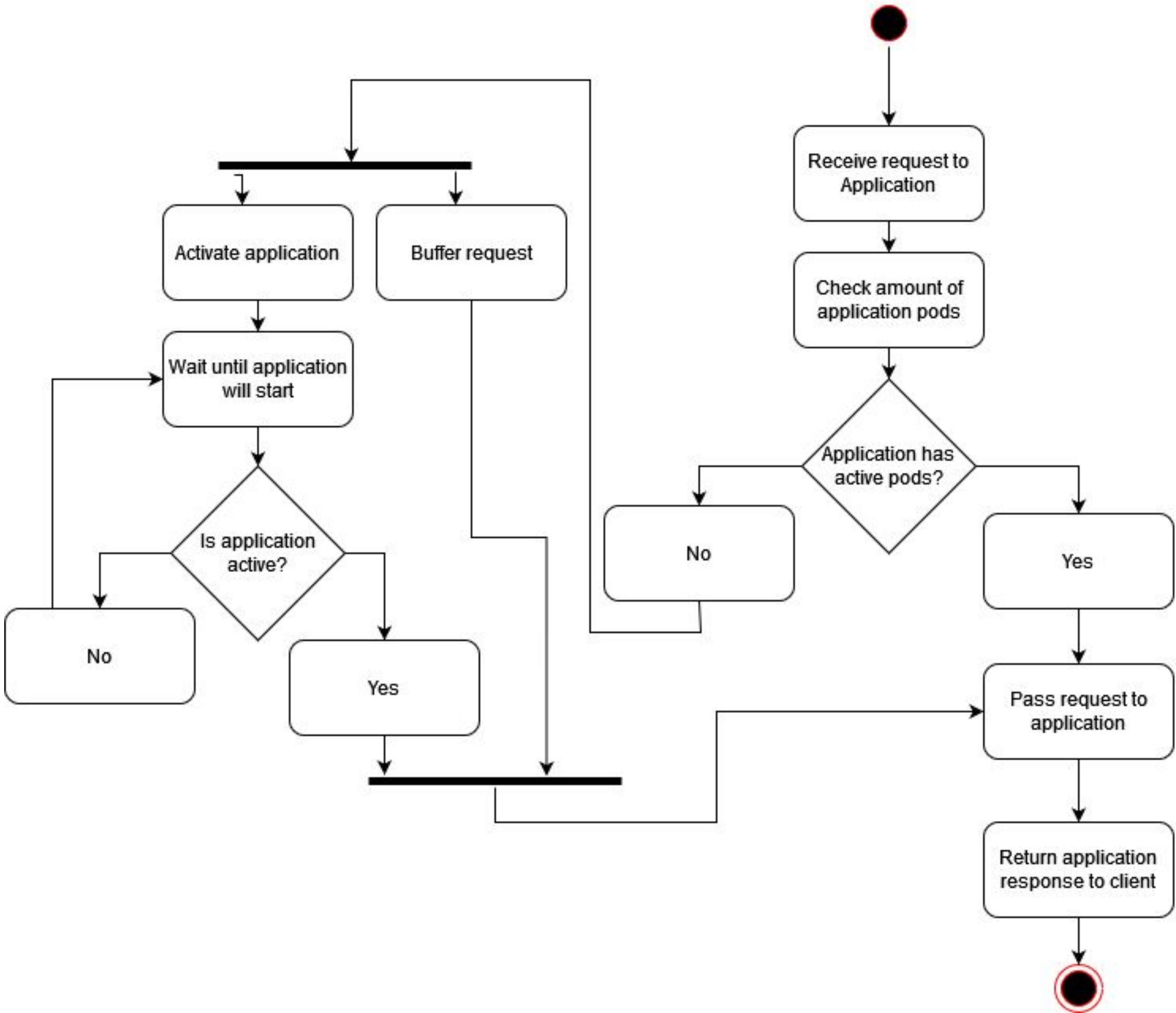
Use Case	Cooperation Type	Used Roles	Candidate Classes
Set scaling rule	Edit	User, ScalingEngine	ScalingEngine, ScalingRule, AutoScalerController, Treshhold
Set load balancing policy	Edit	User, LoadBalancer	LoadBalancer, AutoScalerController
Set application disabling rule	Edit	User, ScalingEngine	ScalingEngine, ScalingRule, AutoScalerController, Treshhold
Proxy request	Proxying	Application client, LoadBalancer, Application pod	LoadBalancer
Buffer request	Proxying	Application client, LoadBalancer	LoadBalancer
Activate cluster	Manage cluster	Application client, LoadBalancer, ScalingEngine	LoadBalancer
Get k8s cluster metrics	Validate	Kubernetes server, AutoScalerController	PodMetrics, MetricsCollector, AutoScalerController
Get current pods liveliness state	Validate	Kubernetes server, AutoScalerController, Application pod	PodMetrics, MetricsCollector, AutoScalerController
Decrease cluster pods amount	Manage cluster	Kubernetes server, AutoScalerController, Application pod	ScalingEngine, AutoScalerResource, AutoScalerController
Increase cluster pods amount	Manage cluster	Kubernetes server, AutoScalerController, Application pod	ScalingEngine, AutoScalerResource, AutoScalerController
Disable cluster	Manage cluster	Kubernetes server, AutoScalerController, Application pod	ScalingEngine, AutoScalerResource, AutoScalerController

Final class diagram

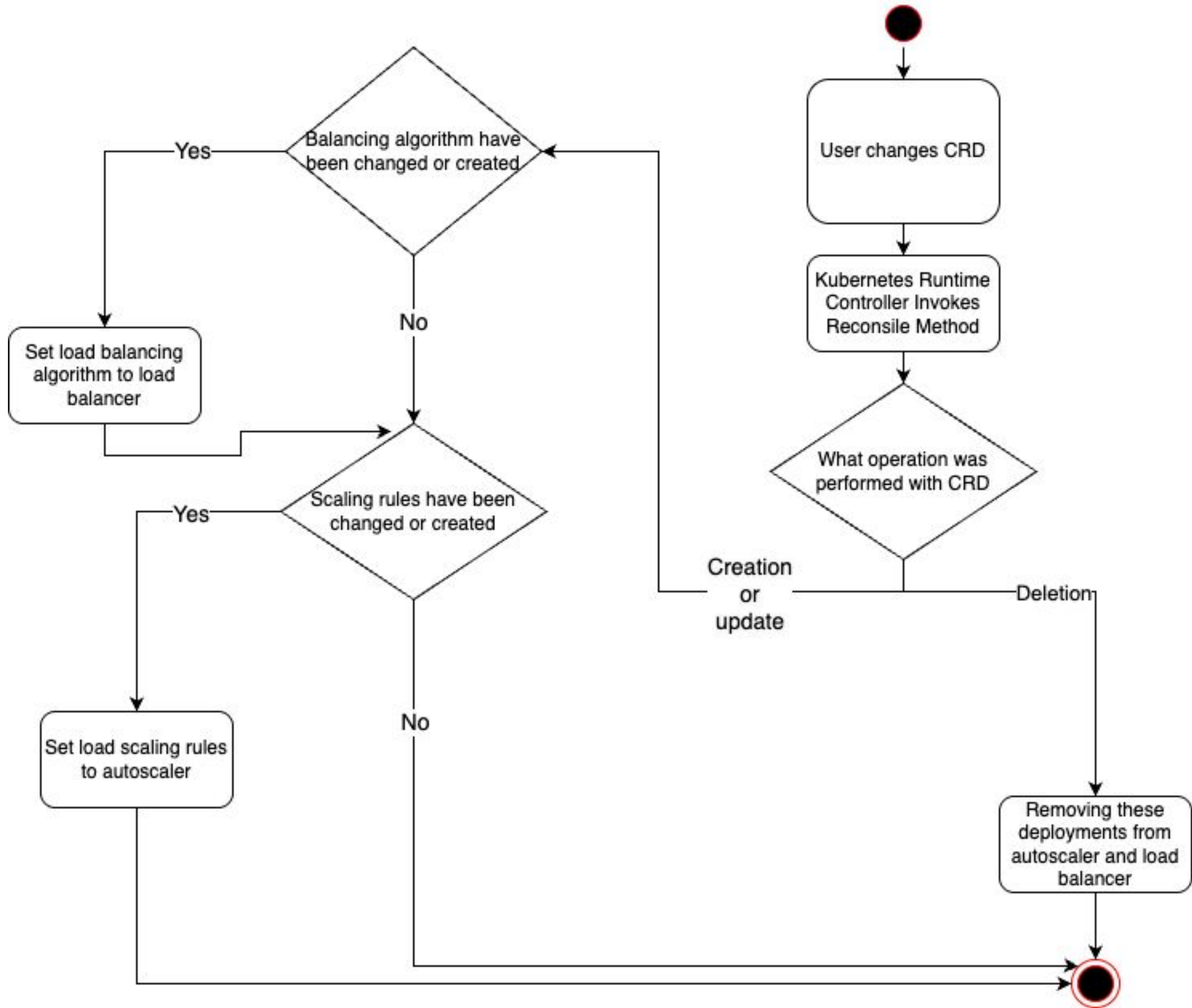


<<Interface>> KubernetesController
+ Reconcile(): List<CustomResource>
CustomResource {abstract}
+ id: String
+ spec: ConfigMap
+ <<signal>> apply()
Extends
<<entity>> AutoScalerResource
+ id: String
+ spec: ConfigMap
+ <<signal>>apply()
<<service>> AutoScalerController
- metricsColector: MetricCollector
- scalingEngine: ScalingEngine
+ getMetrics(): List<PodMetrics>
+ Reconcile(): List<CustomResource>
+ setLoadBalancingAlgorithm(algorithm: LoadBalancingAlgorithmType)
+ setScalingRule(rule: ScalingRule)
<<entity>> PodMetrics
- cpuUsage: float
- memoryUsage: float
- diskUsage: float
- averageResponseTime: float
+ getCpuUsage: float
+ getMemoryUsage: float
+ getDiskUsage: float
+ getAverageResponseTime: float
<<repository>> MetricsCollector
+ collectMetrics(): List<PodMetrics>
1
1
1
1
Manage load balancer settings
<<service>> ScalingEngine
- scalingRules: List<ScalingRule>
+ evaluateMetrics(metrics: PodMetrics)
+ setScalingRule(rule: ScalingRule)
- scaleUp()
- scaleDown()
- scaleToZero()
<<signal>> apply()
<<service>> LoadBalancer
- algorithm: LoadBalancingAlgorithmType
+ balanceRequest(request)
+ setLoadBalancingAlgorithm(algorithm: LoadBalancingAlgorithmType)
<<enumeration>> Action
INCREASE_INSTANCES
DECREASE_INSTANCES
DECREASE_INSTANCES_TO_ZERO
1
1
<<value>> ScalingRule
- action: Action
- thresholds: List<Threshold>
1
1..*
<<enumeration>> MetricType
CPU_USAGE
MEMORY_USAGE
DISK_USAGE
AVERAGE_RESPONSE_TIME
1
1
<<value>> Threshold
- metricType: MetricType
- value: float
- operator: Operator
1
1
<<enumeration>> Operator
GREATER_THAN
LESS_THAN
EQUAL_TO
Manage scaling
Proxy request
<<entity>> Application
+ PodMetrics
1
1
<<enumeration>> LoadBalancingAlgorithmType
ROUND_ROBIN
RANDOM

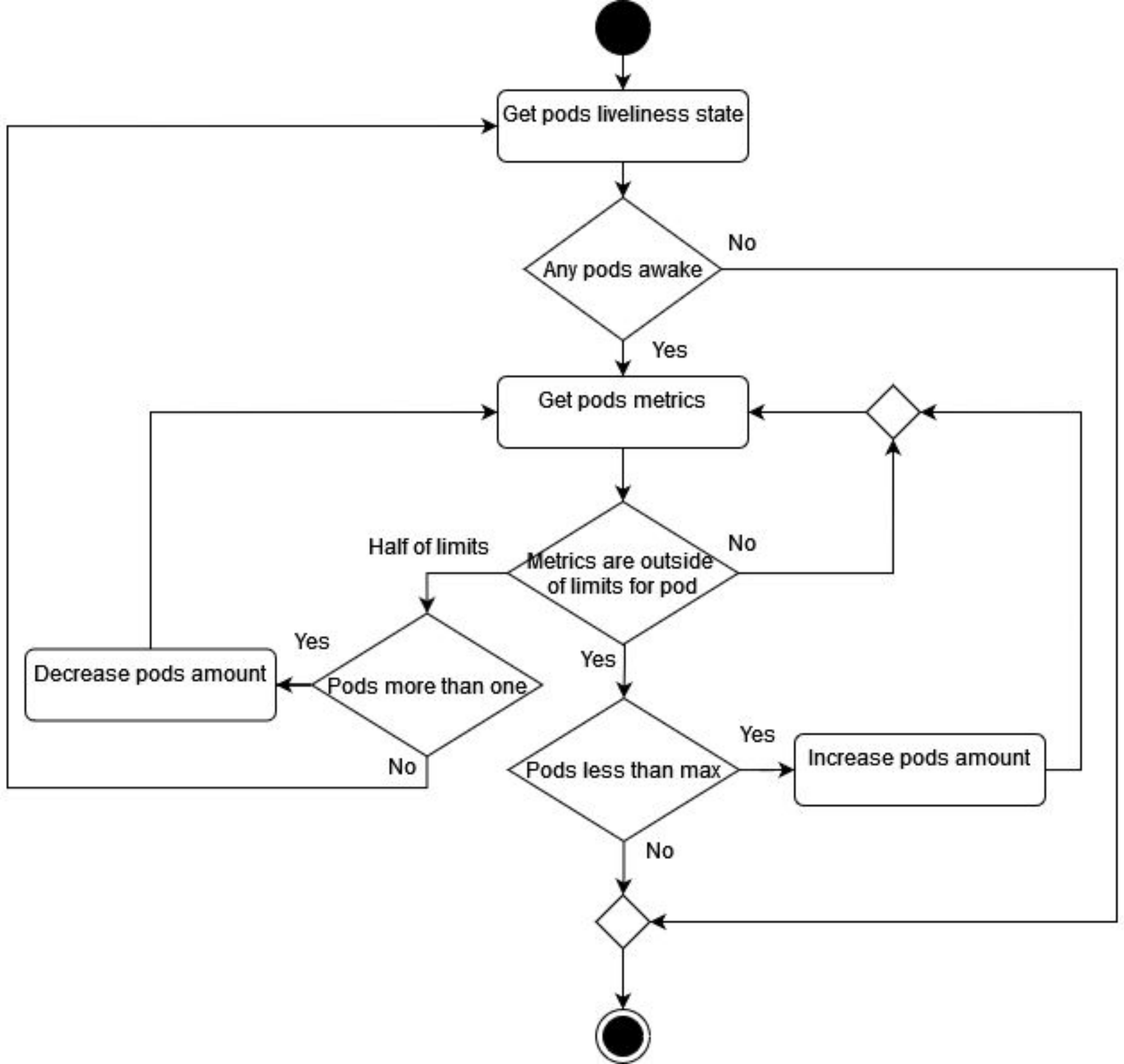
Activity diagram


Receive request to Application
Check amount of application pods
Application has active pods?
No
Yes
Activate application
Buffer request
Wait until application will start
Is application active?
No
Yes
Pass request to application
Return application response to client

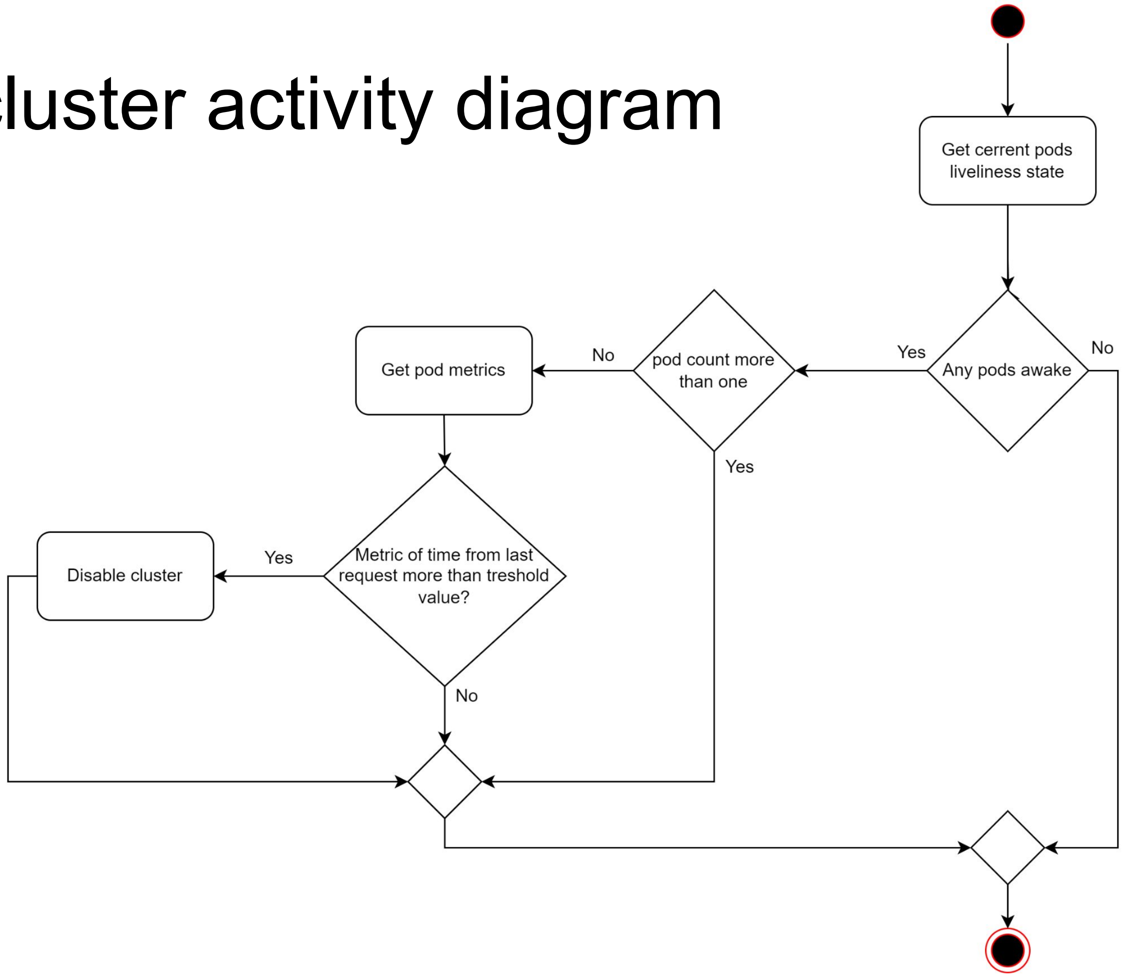
Configuration


User changes CRD
Kubernetes Runtime Controller Invokes Reconsile Method
What operation was performed with CRD
Creation or update
Deletion
Removing these deployments from autoscaler and load balancer
Balancing algorithm have been changed or created
Yes
No
Set load balancing algorithm to load balancer
Scaling rules have been changed or created
Yes
No
Set load scaling rules to autoscaler

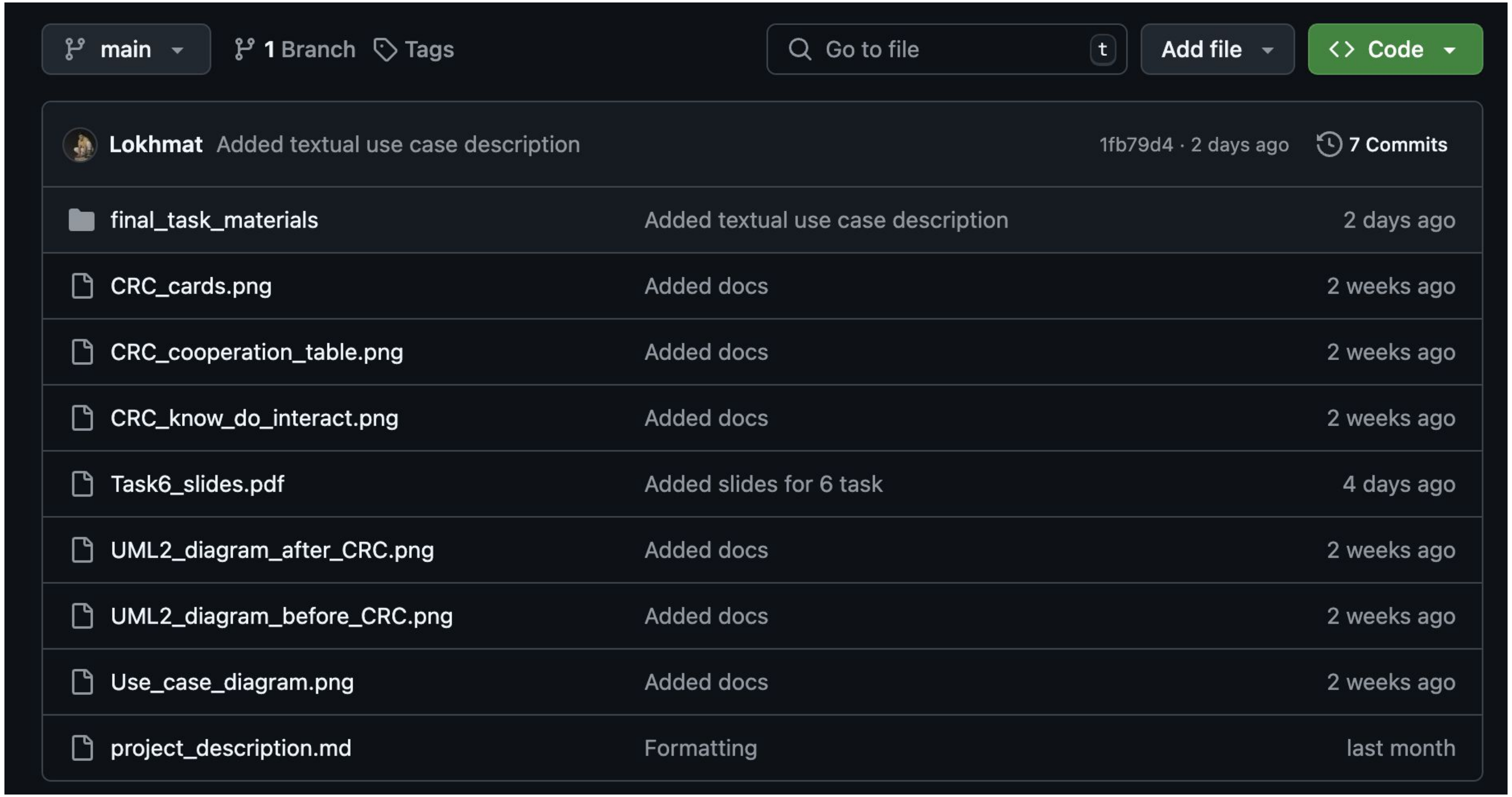
Decrease and Increase cluster pods amount Activity diagram



Disable cluster activity diagram


Get cerrent pods liveliness state
Any pods awake
Yes
No
pod count more than one
No
Yes
Get pod metrics
Metric of time from last request more than treshold value?
Yes
No
Disable cluster

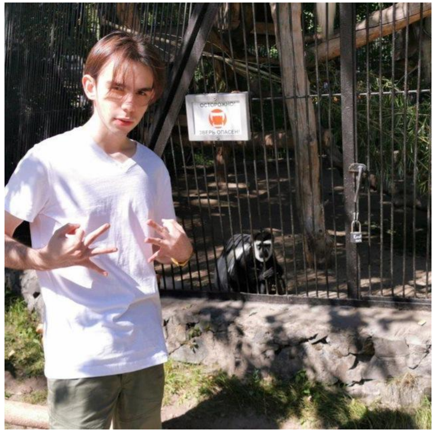
Repository structure



Lokhmat Added textual use case description 1fb79d4 · 2 days ago 7 Commits

final_task_materials	Added textual use case description	2 days ago
CRC_cards.png	Added docs	2 weeks ago
CRC_cooperation_table.png	Added docs	2 weeks ago
CRC_know_do_interact.png	Added docs	2 weeks ago
Task6_slides.pdf	Added slides for 6 task	4 days ago
UML2_diagram_after_CRC.png	Added docs	2 weeks ago
UML2_diagram_before_CRC.png	Added docs	2 weeks ago
Use_case_diagram.png	Added docs	2 weeks ago
project_description.md	Formatting	last month

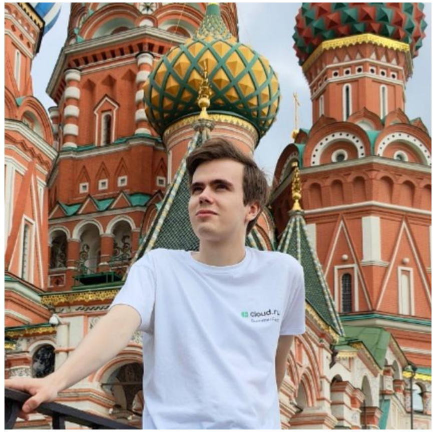
Team and roles



Vasiliy Lopatkin
@DFCZok
Final class diagram
Interaction analysis

Sergey
Lokhmatikov
@Lohmat_Sergey
Story map
Use case diagram

Roman Kuzmenko
@definitely_not_rk
Personas
Behavioral diagram



Andrey Tamplon
@andreytamplon
Final class diagram
Interaction analysis