Load Balancer with Service Turn-Off

Requirements and analysis model

Product description

The product aims to enhance the functionality of a vanilla Kubernetes by enabling it to scale applications down to zero instances when they are not in use. This is relevant for ML applications that can be slow to start. The service will use event-driven automata to manage application scaling in response to real-time monitoring data and optimizing resource usage.

Team: Dmitry Kara, Daniil Mikulik, Ekaterina Karavayeva, Nikita Dumkin

Repo: https://github.com/dmitriykara/ads-tech-tornados-project

Report: https://docs.google.com/presentation/d/10YO03fKJG_qsp0rlpQzeyVC-tEgMQDFBrPsX9gV41J8/edit?usp=sharing

Personas

Name: Ivan
Age: 30
Gender: Male
Location: Russia, Moscow
Education: Bachelor’s Degree
Job title: ML Engineer
Income: 5000$

ML Engineers and Data Scientists

Background: Highly technical professional focused on developing and deploying machine learning models. They have expertise in Python, TensorFlow, PyTorch, and similar ML frameworks.

Needs: Efficient use of computational resources, especially during training and inference phases. Wants to avoid paying for idle resources while still having access to ML models when needed.

Challenges: Long startup times for ML models due to the size and complexity of models, making it difficult to experiment quickly or scale efficiently.

Goal: Minimize downtime and resource usage without compromising performance. Prefers a solution that scales ML workloads up or down automatically based on actual usage.

Personas

Name: Elena
Age: 35
Gender: Female
Location: Russia, Rostov
Education: Bachelor’s Degree
Job title: DevOps Engineer
Income: 1000$

DevOps Engineers

Background: Experienced in cloud infrastructure and Kubernetes. Responsible for managing the scaling, deployment, and optimization of applications within the infrastructure.

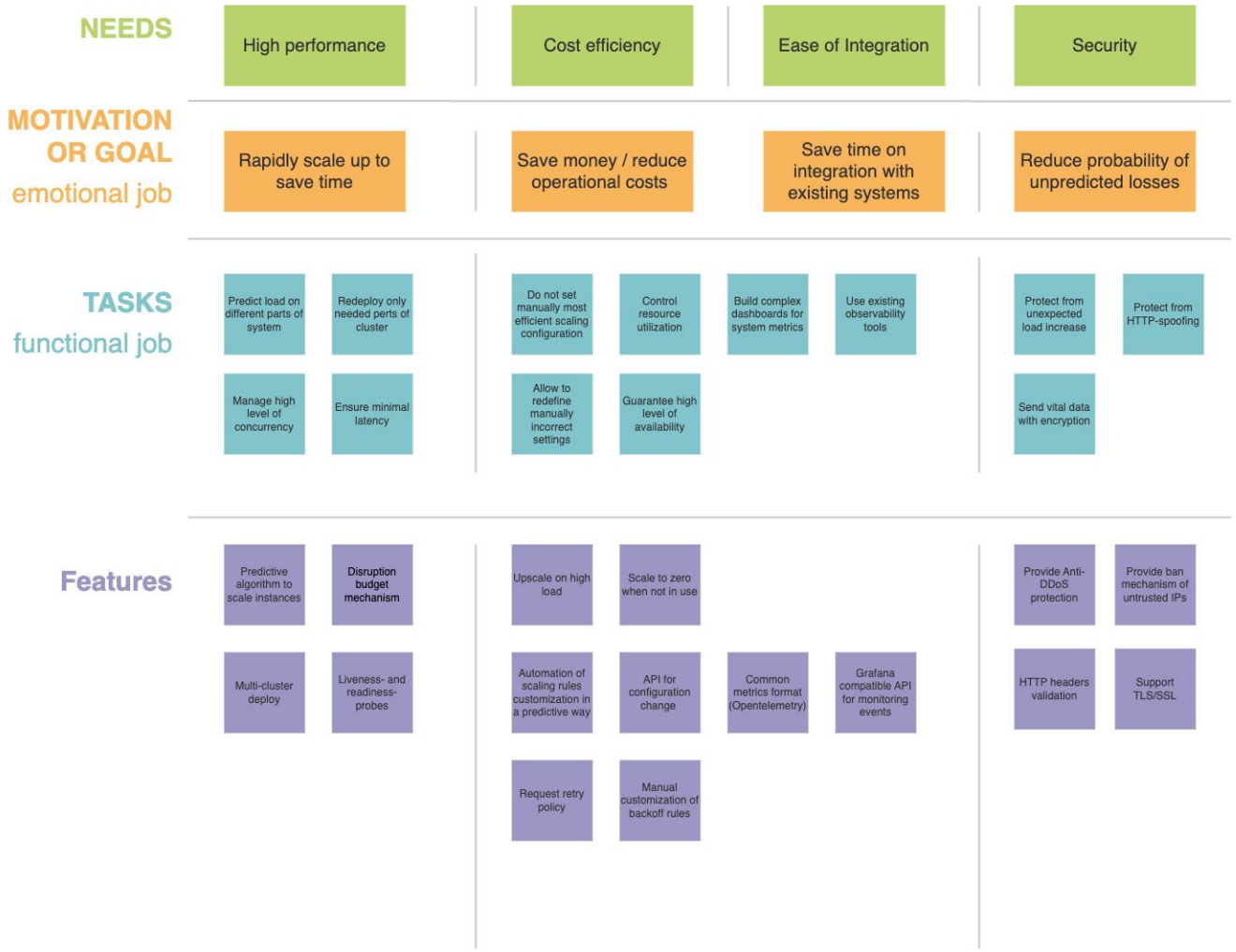
Needs: Tools to automate scaling, optimize resource allocation, and prevent unnecessary costs in cloud environments. Must maintain uptime while reducing waste in resource utilization.

Challenges: Balancing the need for cost savings with ensuring reliable, responsive service for end-users. Often faces complex configuration challenges when managing event-driven scaling.

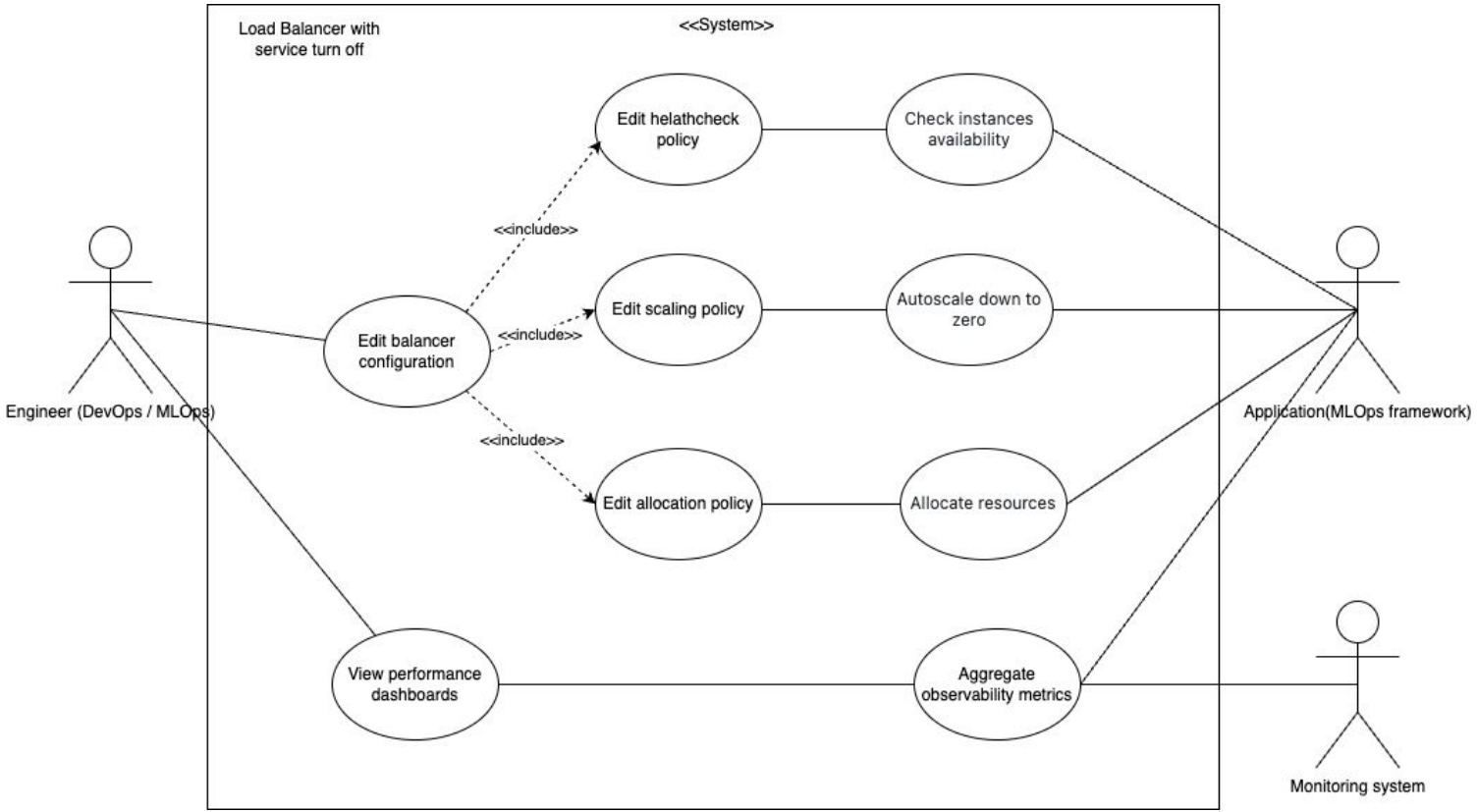
Goal: Deploy a reliable auto-scaling solution that integrates well with existing Kubernetes clusters, automating downscaling to zero when applications are idle and restarting them quickly when needed.

Story map

Full diagram is available here.



Use case diagram



Interaction analysis

Class	Responsibilities (Know, Do, Interact)	Collaborators
Pod	- **Know:** Its configuration, state, and resource usage. - **Do:** Start, stop, restart, and update itself. - **Interact:** Communicate with Kubelet and KubeProxy.	Kubelet, KubeProxy
Kubelet	- **Know:** The state of Pods on the node and their lifecycle. - **Do:** Manage Pods' lifecycle (create, delete, monitor). - **Interact:** Communicate with Pods and KubeProxy.	Pod, KubeProxy
KubeProxy	- **Know:** Network rules and routing tables. - **Do:** Apply network rules and manage routing. - **Interact:** Communicate with Kubelet and Balancer.	Kubelet, Balancer
Balancer	- **Know:** Available Pods and their health status. - **Do:** Distribute traffic among Pods. - **Interact:** Communicate with KubeProxy and ObservabilitySystem.	KubeProxy, ObservabilitySystem

Interaction analysis

Class	Responsibilities (Know, Do, Interact)	Collaborators
HealthCheckPolicy	- **Know:** Configuration for health checks. - **Do:** Execute health checks on Pods and report status. - **Interact:** Communicate with Pod and ObservabilitySystem.	Pod, ObservabilitySystem
ScalingEvent	- **Know:** Current scaling parameters and Pods involved. - **Do:** Scale Pods up or down based on demand. - **Interact:** Communicate with Pod and AllocationPolicy.	Pod, AllocationPolicy
AllocationPolicy	- **Know:** Resource allocation strategies and limits. - **Do:** Allocate resources to Pods. - **Interact:** Communicate with KubernetesNode.	KubernetesNode
KubernetesNode	- **Know:** Groups of services and their limits. - **Do:** Allocate and reallocate services based on policies. - **Interact:** Communicate with AllocationPolicy.	AllocationPolicy
ObservabilitySystem	- **Know:** Metrics and logs from various components. - **Do:** Emit metrics, store logs, and send alerts. - **Interact:** Communicate with Balancer and HealthCheckPolicy.	Balancer, HealthCheckPolicy

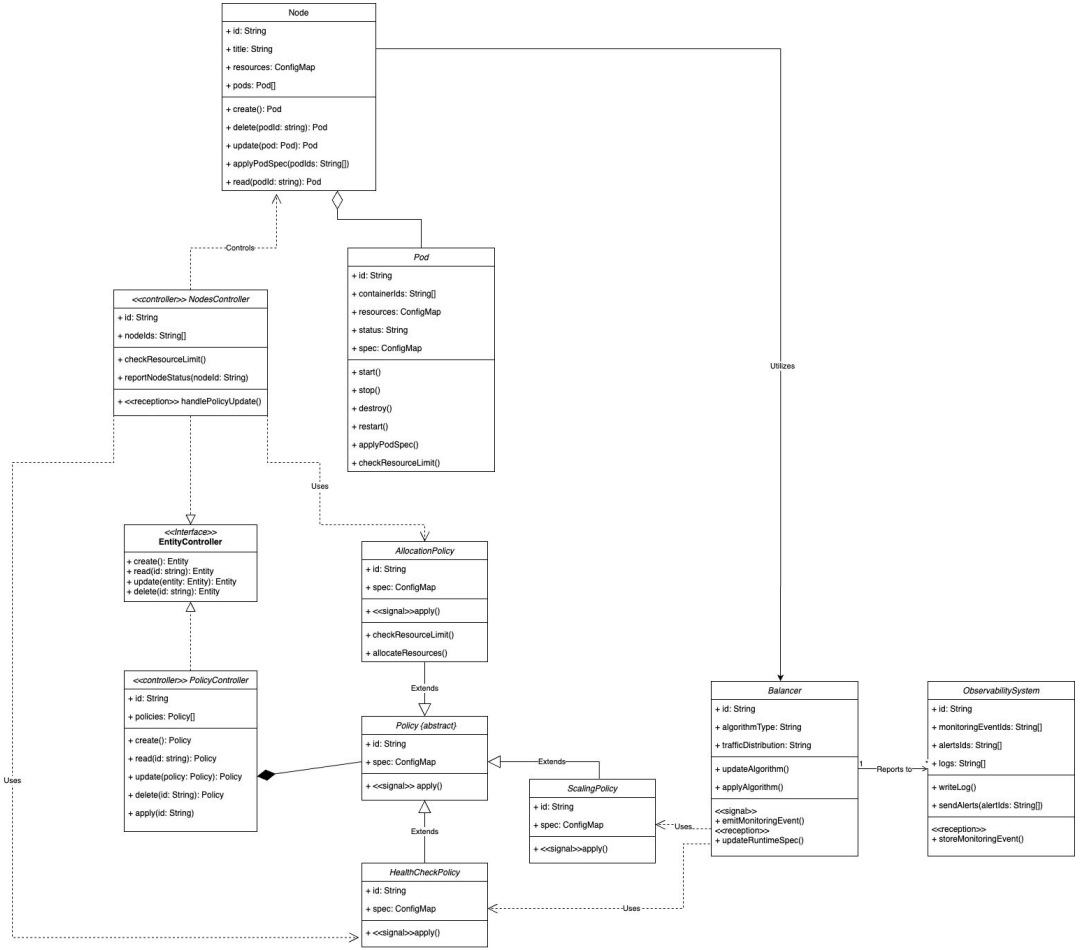
Interaction analysis

Candidate	Criteria	Stored information	Operations
HealthCheckPolicy	SOUT	Id, spec (liveness, readiness, announce probes)	CRUD, apply
AllocationPolicy	SIOU	Id, spec (resources, limits)	CRUD, apply, checkResourceLimit, allocateResources
ScalingPolicy	SOUT	Id, spec (scaling algorithm, scaling timings)	CRUD, apply
Balancer	SOUT	Id, algorithmType, trafficDistribution	applyAlgorithm, updateAlgorithm, updateSpec, emitMonitoringEvent
Pod	SOUT	Id, containerIds, resources, status, podSpec	start, stop, restart, destroy, applyPodSpec, checkResourceLimit
NodeController	SOUT	Id, nodeId, pods, resources	reportNodeStatus, handlePolicyUpdate, checkResourceLimit
ObservabilitySystem	SOUT	Id, metrics, logs, alerts	writeLog, sendAlerts
Node	SOUT	Id, title, resources, pods	applyPodSpec, CRUD for pods
PolicyController	SOUT	Id, policies	CRUD for policies, apply

Final class diagram

Full diagram is available here.

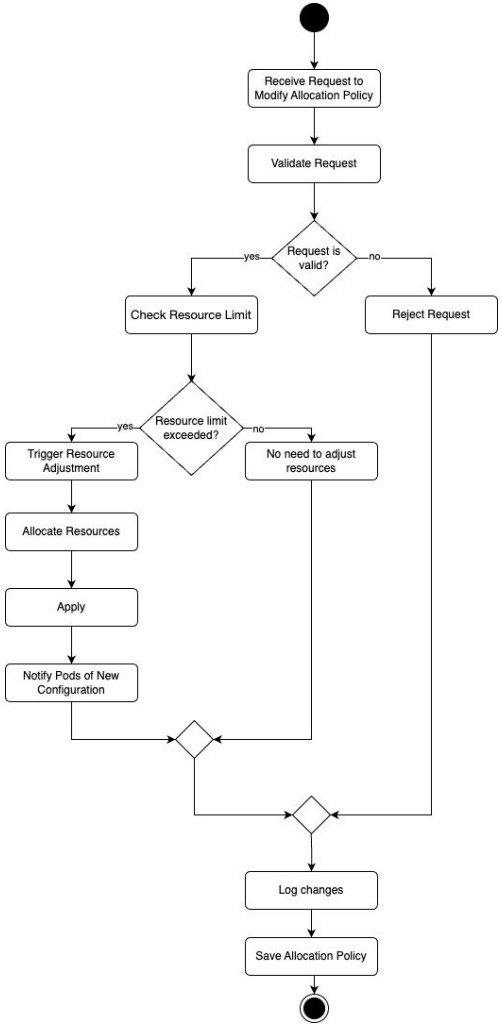
Feel free to investigate a description for the diagram.



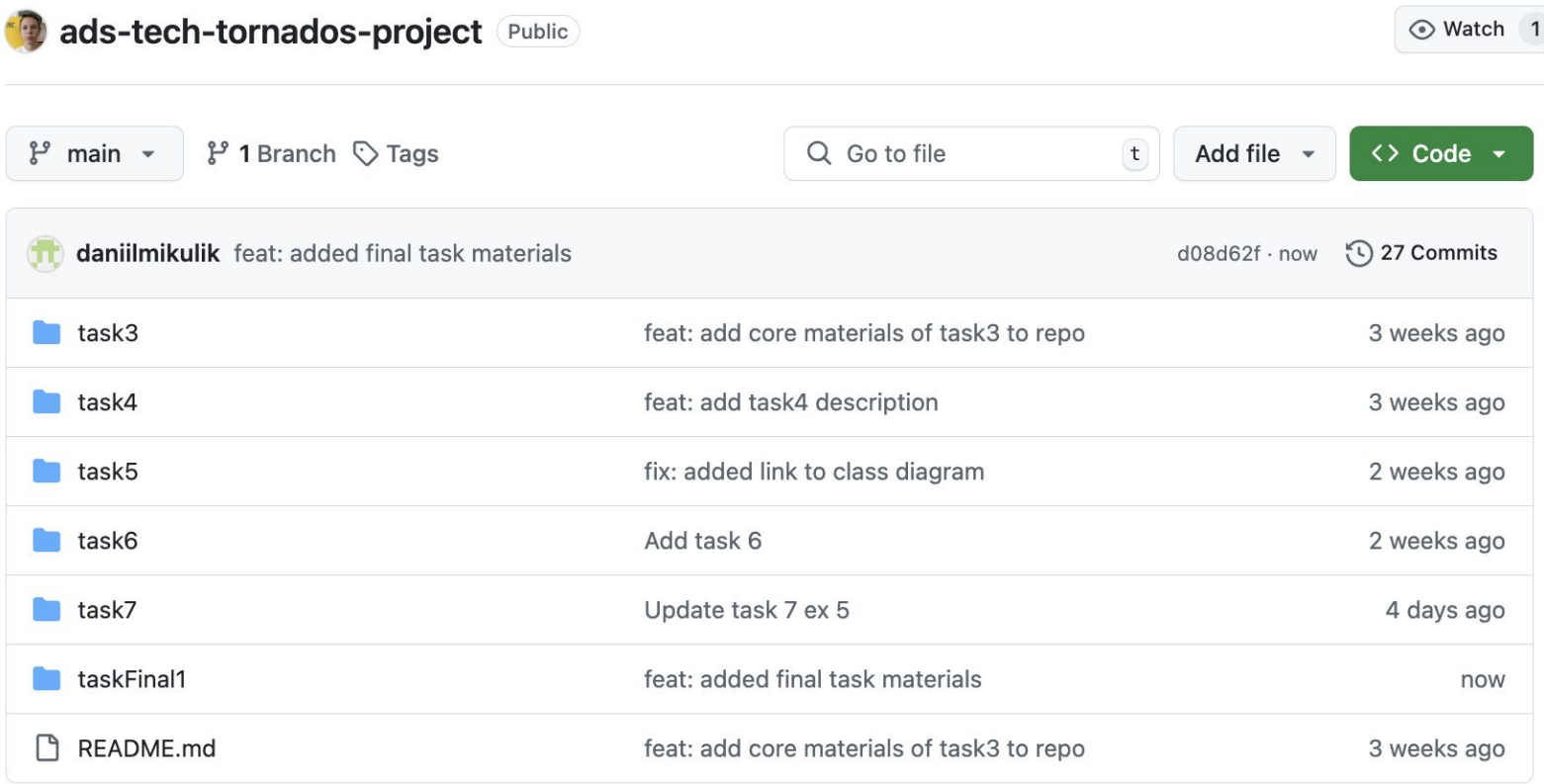
Detailed behavior

The system receives a request to modify the allocation policy. It validates the request, checks resource limits, and if limits are exceeded, adjusts resources, updates Pods, and logs changes before saving the updated policy.

Full diagram which describes edit allocation policy is available here.



Repository structure



Team and roles

 Dmitry Kara (TG: @dmitrykara) - GO developer, DFD and CRC modeller.

 Daniil Mikulik (TG: @CAEDITE_EOS) - React developer, USM, Use Case and Class Diagram modeller.

 Nikita Dumkin (TG: @nikita_dumkin) - Vue developer, requirements and VPC modeller.

 Ekaterina Karavayeva (TG: @KitKat01011) - C# developer, class candidates and personas modeller.