

NEMO

Next Generation Meta Operating System

D4.2 Advanced NEMO platform & laboratory testing results. Initial version

| Document Identification | | | |
|-------------------------|-------|-----------------|------------|
| Status | Final | Due Date | 30/11/2024 |
| Version | 1.0 | Submission Date | 18/12/2024 |

| | | | |
|------------------------|--|-------------------------|---------------------------|
| Related WP | WP4 | Document Reference | D4.2 |
| Related Deliverable(s) | D1.1, D1.2, D1.3, D2.2, D3.2, D4.1 | Dissemination Level (*) | PU |
| Lead Participant | INTRA | Lead Author | Dimitrios Skias (INTRA) |
| Contributors | SYN, INTRA, AEGIS, SPACE, ATOS, MAG, ENG, ESOFTE, SU | Reviewers | Panagiotis Karkazis (MAG) |
| | | | Ignacio Prusiel (ATOS) |

| |
|---|
| Keywords: |
| Integration, Validation, API, SDK, Lifecycle Management, Migration Controller, Automation |

Disclaimer for Deliverables with dissemination level PUBLIC

This document is issued within the frame and for the purpose of the NEMO project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070118. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author's view, and the European Commission is not responsible for any use that may be made of the information it contains. This deliverable is subject to final acceptance by the European Commission.

This document and its content are the property of the NEMO Consortium. The content of all or parts of this document can be used and distributed provided that the NEMO project and the document are properly referenced.

Each NEMO Partner may use this document in conformity with the NEMO Consortium Grant Agreement provisions.

(*) Dissemination level: **(PU)** Public, fully open, e.g., web (Deliverables flagged as public will be automatically published in CORDIS project's page). **(SEN)** Sensitive, limited under the conditions of the Grant Agreement. **(Classified EU-R)** EU RESTRICTED under the Commission Decision No2015/444. **(Classified EU-C)** EU CONFIDENTIAL under the Commission Decision No2015/444. **(Classified EU-S)** EU SECRET under the Commission Decision No2015/444.

Document Information

| List of Contributors | |
|----------------------------|---------|
| Name | Partner |
| Enric Pere Pages Montanera | ATOS |
| Rubén Ramiro | ATOS |
| Ignacio Prusiel | ATOS |
| Matija Cankar | COM |
| Dimitrios Skias | INTRA |
| Panagiotis Karkazis | MAG |
| Astik Samal | MAG |
| Nikos Drosos | SPACE |
| Emmanouil Bakiris | SPACE |
| Antonis Gonos | ESOFT |
| Theodore Zahariadis | SYN |
| Terpsi Velivassaki | SYN |
| Spyros Vantolas | AEGIS |
| Hassane Rahich | SU |

| Document History | | | |
|------------------|------------|------------------------------|---|
| Version | Date | Change editors | Changes |
| 0.1 | 12/09/2024 | INTRA | ToC |
| 0.2 | 03/10/2024 | INTRA | Updates in ToC and initial input |
| 0.3 | 18/10/2024 | INTRA | Updates in section 1,2 and 4 |
| 0.4 | 25/10/2024 | SYN, ATOS, AEGIS, ESOFT | Updates in section 3 and 4 |
| 0.5 | 08/11/2024 | INTRA, MAG | Updates in section 1 and 2 |
| 0.6 | 22/11/2024 | SYN, ATOS, AEGIS, ESOFT | Updates in section 3 |
| 0.7 | 29/11/2024 | INTRA, ATOS, AEGIS, SYN, COM | Updates in section 4, conclusions and introduction of Annex A & B |
| 0.8 | 6/12/2024 | INTRA | Document consolidation; Peer-review ready version |
| 0.9 | 13/12/2024 | INTRA, MAG, ATOS | Document consolidation: Peer-review comments addressed |
| 0.91 | 17/12/2024 | INTRA | Final version ready |
| 1.0 | 18/12/2024 | ATOS | Format review and submission to EC |

| Quality Control | | |
|---------------------|--------------------------|---------------|
| Role | Who (Partner short name) | Approval Date |
| Deliverable leader | D. Skias (INTRA) | 17/12/2024 |
| Quality manager | R. Valle Soriano (ATOS) | 18/12/2024 |
| Project Coordinator | E. Pages (ATOS) | 17/12/2024 |

| | | | |
|-----------------------|---|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 2 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: FINAL |

Table of Contents

| | |
|---|----|
| Document Information | 2 |
| Table of Contents | 3 |
| List of Tables..... | 6 |
| List of Figures | 9 |
| List of Acronyms..... | 12 |
| Executive Summary | 14 |
| 1 Introduction | 15 |
| 1.1 Purpose of the document..... | 15 |
| 1.2 Relation to other project work..... | 15 |
| 1.3 Structure of the document | 15 |
| 2 NEMO Integration, Validation & Verification approach and tools..... | 17 |
| 2.1 NEMO CI/CD Environment & Tools | 17 |
| 2.1.1 Open Source repository | 17 |
| 2.1.2 NEMO Automated Deployment and Configuration..... | 18 |
| 2.2 Cloud/Edge/IoT Integration and Validation Infrastructure..... | 23 |
| 2.2.1 OneLab Clusters for NEMO..... | 23 |
| 2.3 Integration & V&V Methodology & Plan | 31 |
| 2.4 NEMO OneLab infrastructure deployments | 33 |
| 2.5 NEMO Integrated Platform (Ver. 1)..... | 34 |
| 2.5.1 Meta-OS functionality in NEMO v1 | 36 |
| 3 NEMO Service Management Layer updates | 38 |
| 3.1 Intent-based Migration Controller..... | 38 |
| 3.1.1 Overview | 38 |
| 3.1.2 Architecture | 38 |
| 3.1.3 Interaction with other NEMO components..... | 40 |
| 3.1.4 Initial results | 41 |
| 3.1.5 Conclusion and roadmap | 42 |
| 3.2 Plugin & Applications Lifecycle Manager | 42 |
| 3.2.1 Architecture | 42 |
| 3.2.2 Lifecycle Manager..... | 43 |
| 3.2.3 Interaction with other NEMO components..... | 45 |
| 3.2.4 Initial results | 47 |
| 3.2.5 Conclusion and roadmap | 48 |
| 3.3 Monetization and Consensus-based Accountability | 48 |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 3 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

| | | |
|-------|--|-----|
| 3.3.1 | Overview | 48 |
| 3.3.2 | Architecture | 48 |
| 3.3.3 | Interaction with other NEMO components..... | 49 |
| 3.3.4 | Initial result..... | 51 |
| 3.3.5 | Conclusion and roadmap | 71 |
| 3.4 | Intent-based SDK/API | 72 |
| 3.4.1 | Overview | 72 |
| 3.4.2 | Architecture | 72 |
| 3.4.3 | Initial results | 77 |
| 3.4.4 | Conclusion and Roadmap | 77 |
| 4 | NEMO scenario-driven verification & results..... | 79 |
| 4.1 | NEMO Cluster registration | 79 |
| 4.1.1 | Verification scenario..... | 80 |
| 4.1.2 | Results | 81 |
| 4.1.3 | Verification summary checklist..... | 85 |
| 4.2 | NEMO workload registration, deployment & provisioning..... | 85 |
| 4.2.1 | Verification scenario..... | 88 |
| 4.2.2 | Results | 89 |
| 4.2.3 | Verification summary checklist..... | 103 |
| 4.3 | NEMO workload migration | 104 |
| 4.3.1 | Verification scenario..... | 104 |
| 4.3.2 | Results | 105 |
| 4.3.3 | Verification summary checklist..... | 109 |
| 4.4 | NEMO workload lifecycle management..... | 109 |
| 4.4.1 | Process diagram..... | 109 |
| 4.4.2 | Verification scenario..... | 110 |
| 4.4.3 | Results | 111 |
| 4.4.4 | Verification summary checklist..... | 118 |
| 5 | Conclusions | 119 |
| 6 | References | 120 |
| 7 | Annex A – MOCA API & data models..... | 121 |
| 7.1 | MOCA Data models..... | 121 |
| 7.2 | MOCA API endpoints..... | 123 |
| 7.2.1 | GET /api/v1/accounting_events..... | 123 |
| 7.2.2 | DELETE /cluster/delete/{id}..... | 124 |
| 7.2.3 | POST /cluster/register..... | 124 |

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 4 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | FINAL |

| | | |
|--------|--|-----|
| 7.2.4 | GET /cluster/retrieve | 125 |
| 7.2.5 | GET /cluster/retrieve/{id} | 126 |
| 7.2.6 | PUT, PATCH /cluster/update/{id} | 126 |
| 7.2.7 | POST /nemo_token_estimation_setup..... | 127 |
| 7.2.8 | GET /nemo_token_setup_retrieve/{region} | 127 |
| 7.2.9 | GET /nemo_user_info | 128 |
| 7.2.10 | GET /workload/retrieve | 128 |
| 7.2.11 | GET /workload/retrieve/{id} | 128 |
| 7.2.12 | GET /workload_computations/{id} | 129 |
| 8 | Annex B – Intent-based API & data models | 130 |
| 8.1 | NEMO Intent-based API..... | 130 |
| 8.2 | Intent-API data models | 130 |
| 8.3 | Intent-based API endpoints | 138 |
| 8.3.1 | POST /api/v1/auth/login/ | 138 |
| 8.3.2 | POST /api/v1/auth/logout/ | 138 |
| 8.3.3 | POST /api/v1/cluster/register/ | 138 |
| 8.3.4 | GET /api/v1/cluster/retrieve/ | 139 |
| 8.3.5 | GET /api/v1/cluster/retrieve/{id}/ | 139 |
| 8.3.6 | GET, POST /api/v1/intent/ | 139 |
| 8.3.7 | POST /api/v1/intent/template/ | 140 |
| 8.3.8 | GET /api/v1/intent/types/ | 140 |
| 8.3.9 | PUT /api/v1/intent/{id}/action/ | 141 |
| 8.3.10 | PUT /api/v1/intent/{id}/target/ | 141 |
| 8.3.11 | GET, POST /api/v1/workload/ | 141 |
| 8.3.12 | List or Create a new workload document(s) (POST) | 142 |
| 8.3.13 | GET /api/v1/workload/instance/ | 142 |
| 8.3.14 | PUT /api/v1/workload/instance/{instance_id}/delete/ | 143 |
| 8.3.15 | GET /api/v1/workload/instance/{instance_id}/manifests/ | 143 |
| 8.3.16 | POST /api/v1/workload/upload/ | 143 |

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 5 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

List of Tables

| | |
|---|-----|
| Table 1: NEMO dev cluster (K8S) | 25 |
| Table 2: Staging 1 cluster (K8S) | 27 |
| Table 3: Staging 2 Cluster (K3S) | 29 |
| Table 4: Production Cluster (k8S) | 30 |
| Table 5: The NemoTokenEstimation smart contract details | 54 |
| Table 6: The NemoFunds smart contract details | 59 |
| Table 7: The InfrastructureOwnerModel smart contract | 61 |
| Table 8: The ServiceProviderModel smart contract | 66 |
| Table 9: Checklist for cluster registration scenario | 85 |
| Table 10: Checklist for workload registration, deployment and provisioning scenario | 103 |
| Table 11: Test 3 - NEMO workload migration | 105 |
| Table 12: Checklist for Test3 - NEMO workload migration | 109 |
| Table 13: Checklist for Test4 | 118 |
| Table 14: MOCA AccountingEvents Data Model | 121 |
| Table 15: MOCA ClusterResources Data Model | 122 |
| Table 16: MOCA ClusterState Data Model | 122 |
| Table 17: MOCA Workload Data Model | 122 |
| Table 18: MOCA IPFS Handler Data Model | 122 |
| Table 19: MOCA WorkloadComputeTokensEvents Data Model | 123 |
| Table 20: MOCA UserSmartContracts Data Model | 123 |
| Table 21: MOCA NemoTokenSetup Data Model | 123 |
| Table 22: GET Accounting Events responses | 124 |
| Table 23: DELETE Cluster responses | 124 |
| Table 24: DELETE Cluster parameters | 124 |
| Table 25: REGISTER cluster parameters | 125 |
| Table 26: POST Cluster responses | 125 |
| Table 27: GET cluster parameters | 125 |
| Table 28: GET Clusters responses | 125 |
| Table 29: GET Cluster with ID parameters | 126 |
| Table 30: GET Cluster with ID responses | 126 |
| Table 31: PUT, PATCH Cluster parameters | 126 |
| Table 32: PUT, PATCH Cluster responses | 126 |
| Table 33: POST Region Costs responses | 127 |
| Table 34: GET Region Costs parameters | 127 |
| Table 35: GET Region Costs responses | 127 |
| Table 36: GET user information responses | 128 |
| Table 37: GET workloads' details responses | 128 |
| Table 38: GET workload's details parameters | 128 |
| Table 39: GET workload's details responses | 128 |
| Table 40: GET workload computation details parameters | 129 |
| Table 41: GET workload computation details responses | 129 |
| Table 42: Data model description: AuthToken | 130 |
| Table 43: Data model description: ClusterRegister | 130 |
| Table 44: Data model description: ClusterIpfs | 130 |
| Table 45: Data model description: Cluster | 130 |
| Table 46: Data model description: Context | 131 |
| Table 47: Data model description: ExpectationObject | 131 |
| Table 48: Data model description: ExpectationTarget | 131 |
| Table 49: Data model description: IntentExpectation | 131 |
| Table 50: Data model description: Intent | 132 |
| Table 51: Data model description: ContextInput | 132 |

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 6 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: FINAL |

| | |
|---|-----|
| Table 52: Data model description: ExpectationObjectInput | 132 |
| Table 53: Data model description: ExpectationTargetInput | 132 |
| Table 54: Data model description: IntentExpectationInput | 133 |
| Table 55: Data model description: IntentInput | 133 |
| Table 56: Data model description: IntentInputAttribute | 133 |
| Table 57: Data model description: IntentOutput | 133 |
| Table 58: Data model description: TargetTemplate | 133 |
| Table 59: Data model description: IntentTemplate | 133 |
| Table 60: Data model description: IntentActionInput | 134 |
| Table 61: Data model description: IntentTargetUpdate | 134 |
| Table 62: Data model description: User | 134 |
| Table 63: Data model description: WorkloadDocumentChartMaintainer | 134 |
| Table 64: Data model description: WorkloadDocumentChartDependency | 135 |
| Table 65: Data model description: WorkloadDocumentChartMetadata | 135 |
| Table 66: Data model description: WorkloadDocumentChart | 136 |
| Table 67: Data model description: WorkloadDocumentList | 136 |
| Table 68: Data model description: WorkloadDocumentCreate | 137 |
| Table 69: Data model description: WorkloadDocumentLifecycleEvent | 137 |
| Table 70: Data model description: WorkloadDocumentInstance | 137 |
| Table 71: Data model description: WorkloadDocumentUpdate | 138 |
| Table 72: Data model description: WorkloadDocumentTemplateInput | 138 |
| Table 73: POST authorization token parameters | 138 |
| Table 74: POST authorization logout responses | 138 |
| Table 75: POST cluster registration responses | 139 |
| Table 76: POST cluster registration parameters | 139 |
| Table 77: GET cluster retrieve responses | 139 |
| Table 78: GET cluster retrieve (id) responses | 139 |
| Table 79: GET cluster retrieve (id) parameter | 139 |
| Table 80: GET/POST intent responses | 140 |
| Table 81: GET/POST intent parameters | 140 |
| Table 82: GET/POST intent responses | 140 |
| Table 83: POST create intent responses | 140 |
| Table 84: POST create intent parameters | 140 |
| Table 85: GET intent types responses | 141 |
| Table 86: PUT intent action responses | 141 |
| Table 87: PUT intent action request | 141 |
| Table 88: PUT intent's target (id) action responses | 141 |
| Table 89: PUT intent's target (id) action parameters | 141 |
| Table 90: GET workload documents list responses | 142 |
| Table 91: GET workload documents list parameters | 142 |
| Table 92: POST workload document responses | 142 |
| Table 93: POST workload document parameters | 142 |
| Table 94: GET workload instances responses | 142 |
| Table 95: GET workload instances parameters | 142 |
| Table 96: PUT workload instance delete responses | 143 |
| Table 97: PUT workload instance delete parameters | 143 |
| Table 98: GET workload instance manifests responses | 143 |
| Table 99: GET workload instance manifests parameters | 143 |
| Table 100: POST workload upload request responses | 144 |
| Table 101: POST workload upload request parameters | 144 |
| Table 102: GET workload responses | 144 |
| Table 103: GET workload parameters | 144 |
| Table 104: PUT workload responses | 144 |
| Table 105: PUT workload parameters | 145 |
| Table 106: PATCH workload responses | 145 |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 7 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

| | |
|---|-----|
| Table 107: PATCH workload parameters | 145 |
| Table 108: DELETE workload responses | 145 |
| Table 109: DELETE workload parameters | 145 |
| Table 110: POST workload document instance parameters | 146 |
| Table 111: POST workload document instance responses | 146 |

Pending EC Approval

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 8 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

List of Figures

| | |
|--|----|
| Figure 1: NEMO code repository in Eclipse Research labs | 17 |
| Figure 2: NEMO Structure and Ownership | 18 |
| Figure 3: NEMO meta-OS docker registry | 19 |
| Figure 4: NEMO gitlab-ci.yml configuration | 19 |
| Figure 5: Kubernetes deployment manifest example (intent-based API) | 20 |
| Figure 6: Kubernetes ingress manifest example (intent-based API) | 21 |
| Figure 7: Image repository and policy configuration files | 22 |
| Figure 8: Example of the Deployment manifest | 23 |
| Figure 9: Example of MetalLB configuration | 24 |
| Figure 10: Dev cluster nodes | 24 |
| Figure 11: Dev cluster namespaces | 26 |
| Figure 12: Staging 1 cluster nodes | 27 |
| Figure 13: Staging 1 cluster namespaces | 28 |
| Figure 14: Staging 2 cluster nodes | 28 |
| Figure 15: Staging 2 cluster namespaces | 29 |
| Figure 16: Production cluster namespaces | 31 |
| Figure 17: NEMO project phases and main meta-OS version releases | 31 |
| Figure 18: Integration testing – Scenario template | 32 |
| Figure 19: NEMO Integration testing - Checklist template (Example) | 33 |
| Figure 20: Default namespace | 33 |
| Figure 21: Kubernetes-dashboard namespace | 33 |
| Figure 22: l2SM namespace | 33 |
| Figure 23: LinkerD namespace | 33 |
| Figure 24: NEMO Kernel namespace | 33 |
| Figure 25: NEMO-net namespace | 33 |
| Figure 26: NEMO-PPEF namespace | 34 |
| Figure 27: NEMO-sec namespace | 34 |
| Figure 28: NEMO-svc namespace | 34 |
| Figure 29: NEMO-workloads namespace | 34 |
| Figure 30: The NEMO high-level architecture | 35 |
| Figure 31: The 1st integrated version of NEMO meta-OS (high-level architecture view) | 35 |
| Figure 32: IBMC Simplified Architecture | 39 |
| Figure 33: IBMC Complete Architecture | 39 |
| Figure 34: Migration Sequence Diagram | 40 |
| Figure 35: LCM Architecture | 43 |
| Figure 36: Plugin Deployment | 43 |
| Figure 37: Onelab deployment LCM and Security Controller | 45 |
| Figure 38: Intent-based API workload management | 46 |
| Figure 39: Intent-based API intents management | 46 |
| Figure 40: MOCA Resource provisioning | 46 |
| Figure 41: Searching LCM Repository | 47 |
| Figure 42: LCM Dashboard Homepage | 47 |
| Figure 43: MOCA diagram | 49 |
| Figure 44: The MOCA deployment in the Onelab cluster | 49 |
| Figure 45: MOCA integration diagram | 50 |
| Figure 46: MOCA API authorization example | 50 |
| Figure 47: MOCA API | 51 |
| Figure 48: Setup region information through Event Server | 52 |
| Figure 49: Logs of inserting cluster information | 52 |
| Figure 50: Example of the transaction logs of the cluster registration | 59 |
| Figure 51: Example of the transaction logs of the workload registration | 61 |

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 9 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: FINAL |

| | |
|---|----|
| Figure 52: Example of the transaction logs of the workload usage fee | 62 |
| Figure 53: Example of the transaction logs of the cluster reward | 62 |
| Figure 54: Example of the transaction logs of the NEMO fee paid by the cluster owner | 62 |
| Figure 55: Example of the transaction logs of the calculation of the workload usage fee | 63 |
| Figure 56: OneLab workload details | 66 |
| Figure 57: RabbitMQ logs of workload deployment | 66 |
| Figure 58: Workload registration to blockchain | 66 |
| Figure 59: User info for workload owner after registration | 66 |
| Figure 60: Accounting event for workload registration | 67 |
| Figure 61: Smart Contracts deployment through Helm chart | 67 |
| Figure 62: Logs of deployment of NemoFunds contracts | 67 |
| Figure 63: Registering NEMO OneLab Cluster regional info | 68 |
| Figure 64: Response for successful registration | 68 |
| Figure 65: MOCA logs of the DApps component calculating the resource usage of a NEMO workload | 68 |
| Figure 66: The accounting events of the workload user | 69 |
| Figure 67: Workload user information | 69 |
| Figure 68: Details of the workload computation events | 70 |
| Figure 69: Cluster owner accounting events | 70 |
| Figure 70: Cluster owner user information | 71 |
| Figure 71: Scaled up deployment | 71 |
| Figure 72: MOCA logs for scaled workload usage | 71 |
| Figure 73: Comparison of workload usage results | 71 |
| Figure 74: The final Intent-based API architecture | 73 |
| Figure 75: State transitions and reporting events for Intents delivered for fulfilment. [9], supported also in NEMO | 74 |
| Figure 76: The DeliverComputingWorkload intent definition in NEMO Intent-based API | 75 |
| Figure 77: Kubernetes Service Annotations | 77 |
| Figure 78: Process diagram for cluster registration | 80 |
| Figure 79: Cluster summary view on LCM GUI | 81 |
| Figure 80: Cluster registration page on LCM GUI | 81 |
| Figure 81: MOCA Cluster registration demonstration | 82 |
| Figure 82: MOCA Cluster registration response | 82 |
| Figure 83: MOCA sends cluster details to Meta Orchestrator | 83 |
| Figure 84: Meta Orchestrator receives the cluster registration request | 83 |
| Figure 85: MOCA receives the Meta Orchestrator response | 83 |
| Figure 86: Register cluster to blockchain | 83 |
| Figure 87: Updated cluster details | 84 |
| Figure 88: MOCA accounting event for cluster registration | 84 |
| Figure 89: Process diagram for workload registration | 86 |
| Figure 90: Process diagram for workload deployment (provisioning) | 87 |
| Figure 91: Access control sequence diagram - detailed view | 88 |
| Figure 92: NEMO workload registration through LCM UI | 90 |
| Figure 93: NEMO registered workloads | 90 |
| Figure 94: NEMO workload instance creation (workload deployment process) through LCM UI | 91 |
| Figure 95: NEMO workload instances and their respective status in LCM UI | 91 |
| Figure 96: NEMO workload validation | 91 |
| Figure 97: Workload deployment confirmation through RabbitMQ for the newly created workload instance | 92 |
| Figure 98: Intent-based API endpoint where the scenario starts. | 93 |
| Figure 99: JSON published in RabbitMQ to be consumed by MO. | 93 |
| Figure 100: Target cluster without the workload. | 94 |
| Figure 101: Deployment Controller log, receiving the workload petition and deploying it. | 94 |
| Figure 102: Workload already deployed in the cluster selected. | 94 |
| Figure 103: Deployment Controller (MO) final response. | 94 |
| Figure 104: Create workload through Intent-based API with Ingress support | 95 |
| Figure 105: Workload Ingress annotations for integrating with Access Control | 95 |

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 10 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: FINAL |

| | |
|---|-----|
| Figure 106: The Onelab KongPlugin resources | 95 |
| Figure 107: Workload Access Control service | 96 |
| Figure 108: Workload Access Control service details | 96 |
| Figure 109: Workload Access Control route | 96 |
| Figure 110: Workload Access Control route details | 97 |
| Figure 111: Workload OAuth2.0 plugin | 97 |
| Figure 112: Workload OAuth2.0 plugin details | 98 |
| Figure 113: Workload OAuth2.0 plugin (cont'd) | 98 |
| Figure 114: NEMO OAuth2.0 plugin test | 99 |
| Figure 115: OAuth2.0plugin test - expired or false token | 99 |
| Figure 116: OAuth2.0 plugin test - success | 100 |
| Figure 117: Locust experiments' setup | 101 |
| Figure 118: Locust request and response statistics for standard OAuth2.0 plugin | 102 |
| Figure 119: Locust request and response statistics for simplified OAuth2.0 plugin | 102 |
| Figure 120: Locust charts for the OAuth2.0 plugin implementations | 103 |
| Figure 121: NEMO workload migration sequence diagram | 104 |
| Figure 122: Intent message reaches MO | 105 |
| Figure 123: Pods currently running in onelab | 106 |
| Figure 124: Workload ID inspection | 106 |
| Figure 125: Availability check | 106 |
| Figure 126: Migration message reaches source cluster's IBMC instance | 107 |
| Figure 127: Backup status | 107 |
| Figure 128: Restore message reaches target cluster's IBMC instance | 107 |
| Figure 129: k3s cluster status before migration | 107 |
| Figure 130: k3s cluster status after migration completion | 107 |
| Figure 131: Description of workload in k3s cluster | 108 |
| Figure 132: OneLab cluster after migration | 108 |
| Figure 133: Process diagram for workload monitoring and enforcement | 110 |
| Figure 134: Three queries to obtain network traffic stats collected by CMDT through Linkerd | 112 |
| Figure 135: Expected RabbitMQ message data model | 113 |
| Figure 136: Workload – CPU usage | 114 |
| Figure 137: Workload - RAM usage | 114 |
| Figure 138: Workload - Energy consumption rate | 114 |
| Figure 139: Workload - Energy efficiency | 115 |
| Figure 140: Workload - Energy consumption | 115 |
| Figure 141: Intent-API EnergyEfficiency metrics update | 116 |
| Figure 142: Cluster RAM usage | 117 |
| Figure 143: Cluster CPU usage | 117 |
| Figure 144: Cluster Disk usage | 117 |
| Figure 145: cluster metrics published to RabbitMQ | 117 |

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 11 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

List of Acronyms

| Abbreviation / acronym | Description |
|------------------------|---|
| AAA | Authentication, Authorization, and Accounting |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CD | Continuous Delivery |
| CFDRL | Cybersecure Federated Deep Reinforcement Learning |
| CLI | Command Line Interface |
| CMDT | Cybersecure Microservices' Digital Twin |
| CI | Continuous Integration |
| CLI | Command-line Interface |
| CMDT | Cybersecure Microservices' Digital Twin |
| CNCF | Cloud Native Computing Foundation |
| CPU | Central Processing Unit |
| CRD | Custom Resource Definition |
| DApps | Distributed Applications |
| DLT | Distributed Ledger Technology |
| Dx.y | Deliverable number y belonging to WP x |
| E2E | End-to-End |
| EC | European Commission |
| FL | Federated Learning |
| GDPR | General Data Protection Regulation |
| GPU | Graphics Processing Unit |
| IBMC | Intent-based Migration Controller |
| IdM | Identity Management |
| IDS | Intrusion Detection System |
| IPFS | Interplanetary File System |
| IoT | Internet-of-Things |
| IT | Information Technology |
| K8s | Kubernetes |
| LAN | Local Area Network |
| LCM | Life-Cycle Manager |
| meta-OS | Meta-Operating System |
| ML | Machine Learning |
| mNCC | Meta Network Cluster Controller |
| MO | Meta-Orchestrator |
| MOCA | Monetization and Consensus-based Accountability |
| MQTT | Message Queuing Telemetry Transport |
| NAC | NEMO Access Control |
| OS | Operating System |
| PPEF | PRESS & Policy Enforcement Framework |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 12 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

| | |
|------|------------------------------|
| RAM | Random Access Memory |
| RBAC | Role-Based Access Control |
| RL | Reinforcement Learning |
| SDK | Software Development Kit |
| SEE | Secure Execution Environment |
| TRL | Technology Readiness Level |
| V&V | Validation & Verification |
| WAL | Write-Ahead Logging |
| WP | Work Package |
| YAML | Yet Another Markup Language |

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 13 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

Executive Summary

The document presents insights into the first integrated version of the NEMO meta-OS consisting of the core components, the interfaces and the integration activities to this point. The work also involves the creation of integration scenarios that guided the integration tests conducted in a laboratory setting, utilizing the supporting CI/CD environment and tools and verified the system-level technical capacity of the platform. Moreover, the deliverable provides a detailed presentation of the technical developments conducted within WP4, detailing the NEMO meta-OS Service Management Layer's technical advancements and updates, including their associated interactions within NEMO meta-OS. The final version of the NEMO meta-OS integrated platform is expected to be presented in D4.3 [1], "Advanced NEMO Platform & Laboratory Testing Results. Final Version," which will be produced in the second quarter of 2025.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 14 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

1 Introduction

The NEMO meta-OS framework aims to support optimal operation of hyper-distributed applications implemented as microservices in a highly distributed and diverse environment of cloud-edge and IoT technologies. Therefore, the associated integration and subsequently the verification and validation of such activities are not trivial and require a well-designed methodology and execution plan.

In D4.1 [2], a detailed description of the zero-ops CI/CD environment and the associated integration guidelines were discussed. This document aims to shed light on the scenario-driven integration and validation activities that aim to realise the first integrated version of the NEMO meta-OS as dictated by the Validation & Verification methodology that is presented. The main objective of this approach is on one hand to assure that NEMO specifications on interfaces and data models are coherent and followed by the stemming technical developments and on the other to facilitate and accelerated the necessary integration activities.

1.1 Purpose of the document

The purpose of this document is twofold. First, it presents the technical advancements that fall into the NEMO Service Management Layer and second to present in detail the integration activities that are driven by the NEMO Validation & Verification (V&V) methodology and document the associated results which led to the production of the first integrated NEMO meta-OS framework.

1.2 Relation to other project work

The integration and testing strategy act as the driver of the development process. Thus, this document is strongly connected with all the technical WPs (WP2, WP3 and WP4). Furthermore, the work presented in this document strongly relates to WP1 activities, as it considers the technical specifications arising from the requirements' elicitation process and the architectural specifications. In addition, the platform integrated view and current prototype implementations will be applied and tailored to each of the NEMO trials within WP5. Last, but not least, the document reports technical options and prototype functionalities, which are meant to be used and extended by third parties joining the project through the Open Calls.

1.3 Structure of the document

The remainder of this report is organized as follows.

Section 2 provides information on the CI/CD environment of the NEMO meta-OS and on the OneLab facilities that provide for the integration activities of the first integrated version of the NEMO meta-OS. In addition, it presents the high-level architecture view of the first integrated version of the NEMO meta-OS highlighting the key integration activities for each functional layer and describes the components that are fully or partially integrated.

Section 3 describes the overview, the architecture, the initial results and the interactions with other components for the modules that are comprising the Service Management Layer of the NEMO meta-OS platform, namely the intent-based Migration Controller (IBMC), the Plugin & Application Lifecycle Manager (LCM), the Monetization and Consensus based Accountability (MOCA) and the Intent-based SDK/API.

Section 4 sheds light into the integration activities that are conducted and materialized the first integrated version of the NEMO meta-OS, following the scenario-driven V&V methodology.

Section 5 provides conclusions and insights in view of the final version of the NEMO meta-OS.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 15 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | FINAL |

Finally, Annex A & B provide a detailed description of the Intent-based API and MOCA interfaces and data models.

Pending EC Approval

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 16 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

2 NEMO Integration, Validation & Verification approach and tools

The first integrated NEMO meta-OS version that is described in this document capitalizes on the NEMO CI/CD environment and tools. The associated CI/CD pipeline facilitates the agile integration and validation approach that the NEMO adopts and is applied throughout the cloud and edge infrastructure that is available and orchestrated by NEMO meta-OS. The NEMO meta-OS underlying infrastructure resides in OneLab facilities. The following sections shed light both on the source code repository configuration in Eclipse Gitlab and on the OneLab cloud and edge infrastructure that is provided to NEMO meta-OS. This environment is essential for conducting and subsequently demonstrating the integration and validation activities that resulted in the first integrated version of NEMO meta-OS.

2.1 NEMO CI/CD Environment & Tools

2.1.1 Open Source repository

For the NEMO project, the GitLab CI/CD framework has been set up and organized in an Eclipse Research Labs hosted instance of GitLab. The official GitLab group of NEMO is titled “NEMO Project” and is accessible publicly at <https://gitlab.eclipse.org/eclipse-research-labs/nemo-project>. The group hosts the source code that is related to each thematic entity-specific development as dictated by the NEMO meta-OS architecture. Each thematic entity is organized as a subgroup of the NEMO GitLab group, Figure 1.

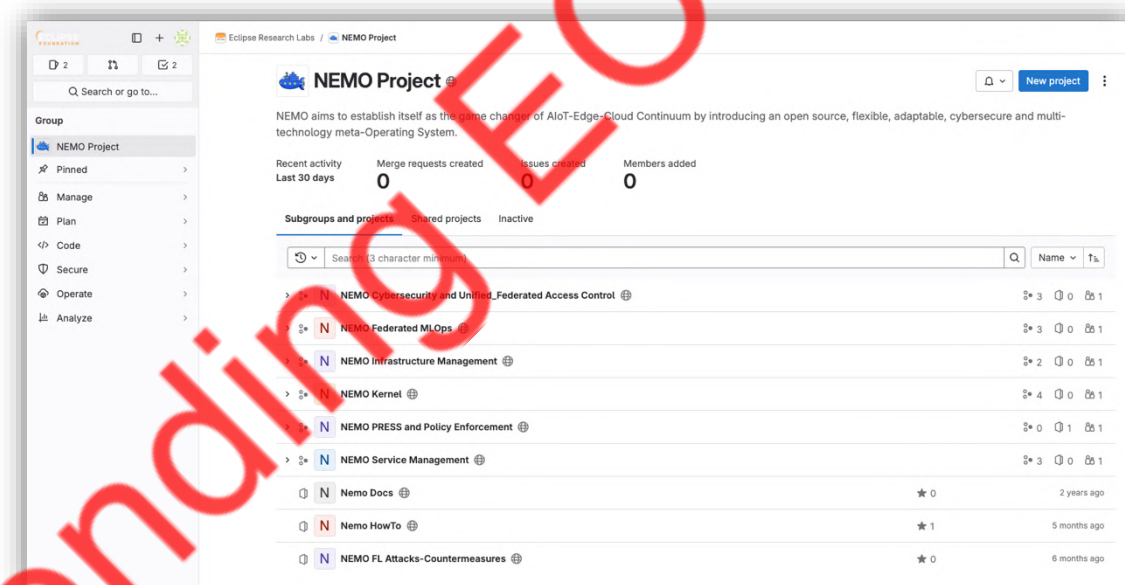


Figure 1: NEMO code repository in Eclipse Research labs

Within each subgroup, the development activities are organized based on the implemented outcomes of the relevant tasks. Moreover, for each subgroup an owner is assigned as illustrated in the Figure 2 below.

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 17 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

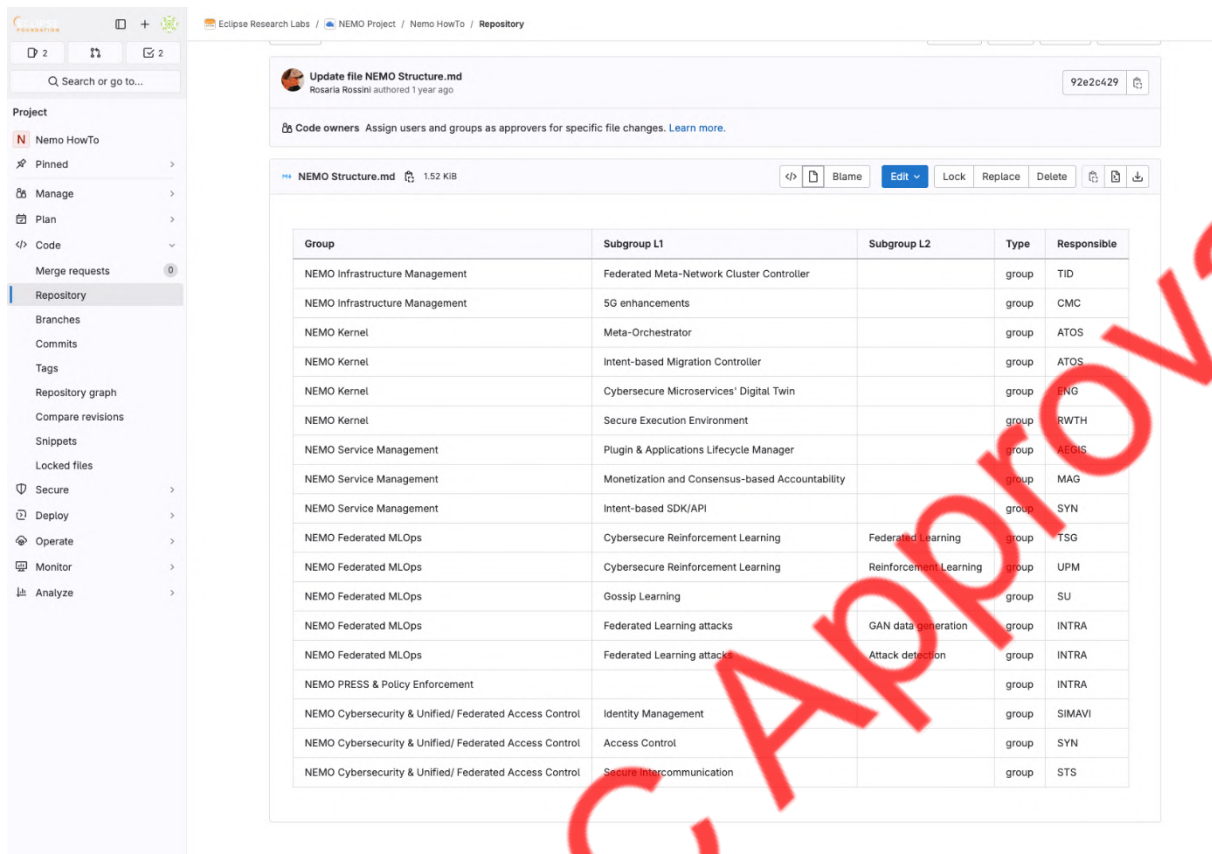


Figure 2: NEMO Structure and Ownership

2.1.2 NEMO Automated Deployment and Configuration

The NEMO CI/CD environment capitalizes on Kubernetes¹ manifests for the deployment of the NEMO components in the NEMO meta-OS infrastructure. The documentation that concerns the NEMO CI/CD integration steps that are necessary for the component deployment process is described in a readme file within the NEMO repository.

The NEMO components' container images are uploaded to the docker.io instance under the account of NEMO, that is following the naming convention "*nemometaos/Xcomponent_nameX*". The docker² image repository (<https://hub.docker.com/u/nemometaos>) is presented in Figure 3.

¹ <https://kubernetes.io/>

² <https://www.docker.com/>

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 18 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

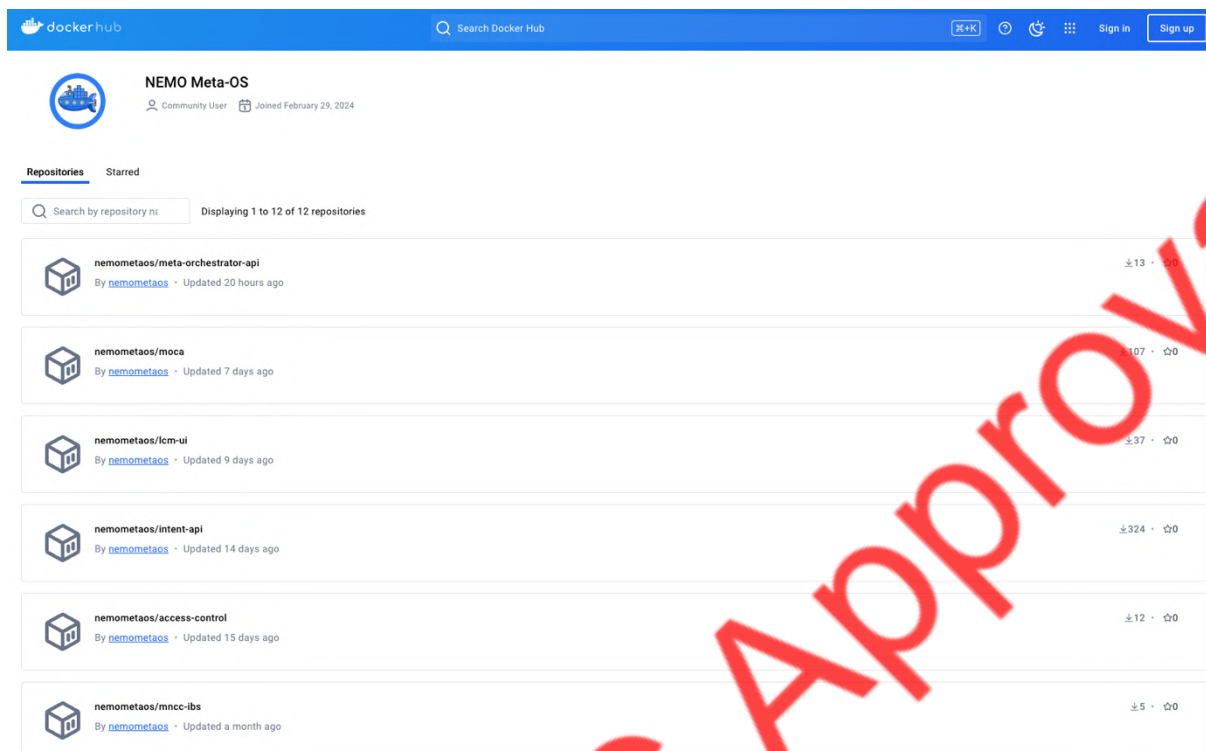


Figure 3: NEMO meta-OS docker registry

Once a valid *dockerfile*³ exists, a “.gitlab-ci.yml” file must be created in the project root directory. Then once the *dockerfile* exists for the NEMO developed component the “.gitlab-ci.yml” file must be created in the project root directory. Figure 4 below presents the relevant configuration file that the NEMO partners must adapt to their technical solution according to the instructions that are provided.

```
include:
  - project: 'eclipsefdn/it/releng/gitlab-runner-service/gitlab-ci-templates'
    file: 'jobs/buildkit.gitlab-ci.yml'
  - project: 'eclipsefdn/it/releng/gitlab-runner-service/gitlab-ci-templates'
    file: 'pipeline-autodevops.gitlab-ci.yml'

stages:
  - build
  - test

variables:
  CI_REGISTRY_IMAGE: nemometaos:<component_name>

buildkit:
  extends: buildkit

unit-test:
  stage: test
  script:
    - echo "Running unit tests... This will take about 10 seconds."
    # - docker run $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA /script/ta/run/tests
    - sleep 10
    - echo "Tests passed successfully!"
```

Figure 4: NEMO gitlab-ci.yml configuration

³ <https://docs.docker.com/reference/dockerfile/>

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 19 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

In the above example the NEMO partners have to substitute the `<component_name>` with the name of their component. This configuration will do the following:

Upload the docker image `nemometaos/<component_name>:latest` tag every time a commit happens to the main branch

Upload the docker image `nemometaos/<component_name>:<tag_name>` tag every time a new tag is created.

For example, the creation of tag `v0.1.1` on the `nemometaos/intent-api` uploaded to the docker registry is `nemometaos/intent-api:v0.1.1` image. The version of the component (tag) must be always ascending integers.

In order to deploy the NEMO component to Kubernetes orchestrated environment the component owner must provide all the necessary kubernetes configuration files (manifests) and test that they can be deployed and work in the OneLab cluster by using the provided credentials to access the cluster.

In order to pull images from the `nemometaos` account, every namespace in Kubernetes has the `nemo-regcred` secret that must be used as `imagePullSecrets` in the component's `Deployment` manifest as indicated in the following example.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: intent-api
  namespace: nemo-svc
  labels:
    app: nemo
    type: backend
    deployment: intent-api
spec:
  replicas: 3
  strategy:
    rollingUpdate:
      maxUnavailable: 1 # for pod anti-affinity to work
    type: RollingUpdate
  selector:
    matchLabels:
      pod: intent-api
  template:
    metadata:
      labels:
        app: nemo
        type: backend
        pod: intent-api
    spec:
      imagePullSecrets:
        - name: nemo-regcred
      containers:
        - name: django
          image: nemometaos/intent-api:v0.0.10
          imagePullPolicy: Always
...

```

Figure 5: Kubernetes deployment manifest example (intent-based API)

Moreover, the NEMO meta-OS provided integration guide described the *Ingress* configuration file that concerns the communication of the outside world with the deployed component.

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 20 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |


```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: intent-api-ingress
  namespace: nemo-svc
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-production"
spec:
  ingressClassName: "nginx"
  tls:
  - hosts:
    - intent-api.nemo.onelab.eu
    secretName: intent-api-tls
  rules:
  - host: intent-api.nemo.onelab.eu
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: intent-api
            port:
              number: 80

```

Service:

```

service:
  name: intent-api
  port:
    number: 80

```

must match the service attached to your Deployment manifest

```

annotations:
  cert-manager.io/cluster-issuer: "letsencrypt-production"

```

and

```

tls:
  - hosts:
    - intent-api.nemo.onelab.eu
    secretName: intent-api-tls

```

automatically created certificates via `cert-manager` / Lets Encrypt

```

ingressClassName: "nginx"

```

connects to the onelab nginx ingress controller

```

- host: intent-api.nemo.onelab.eu

```

Your component must be `intent-api.nemo.onelab.eu`.

Figure 6: Kubernetes ingress manifest example (intent-based API)

Once the steps mentioned above has been completed then the developed component is deployed successfully in the Kubernetes orchestrated environment in OneLab. In NEMO, the Continuous Deployment part of the pipeline is configured through the FLUX CD⁴ which is a CNCF⁵ adopted open-source tool that enables GitOps for managing the configuration of a Kubernetes cluster. In a GitOps pipeline, the desired state of the cluster is stored in a Git repository, and FLUX CD ensures that the actual cluster state matches the desired state defined in the repository.

⁴ <https://fluxcd.io/>

⁵ <https://cncf.io/>

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 21 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

Once the manifests are created and the component is verified that is working, the manifests must be transferred to the FLUX CD repository. The repository is structured in folders as follows:

<cluster_name> / <kubernetes_namespace> / <component_name> / <component_subcomponents>

Each component must be deployed into the respective folder that matched their Kubernetes assigned namespace. Sub dependencies (e.g. postgres⁶, redis⁷ etc) can be also setup as Helm charts⁸.

To automate the process an *ImageRepository* & *ImagePolicy* Custom Resource Definition (CRD) must be committed alongside the component manifests as indicated by the following examples.

ImageRepository

```
---
apiVersion: image.toolkit.fluxcd.io/v1beta1
kind: ImageRepository
metadata:
  name: intent-api
  namespace: flux-system
spec:
  image: nemometaos/intent-api
  interval: 1m0s
  secretRef:
    name: nemo-regcred
```

ImagePolicy

```
---
apiVersion: image.toolkit.fluxcd.io/v1beta1
kind: ImagePolicy
metadata:
  name: intent-api
  namespace: flux-system
spec:
  imageRepositoryRef:
    name: intent-api
  filterTags:
    extract: $version
    pattern: ^(?P<version>v?\d+\.\d+\.\d+[a-zA-Z]*)$
  policy:
    semver:
      range: '*'
```

Figure 7: Image repository and policy configuration files

The proper names must be set and must reside inside the *flux-system* namespace, and the appropriate image repository must be set too. After that, the *Deployment manifest* of the component must upsert the following annotation to the line that defines the newly created docker image as indicated in the rectangular box in Figure 8 below.

⁶ <https://www.postgresql.org/>

⁷ <https://redis.io/>

⁸ <https://helm.sh/>

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 22 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: intent-api
  namespace: nemo-svc
  labels:
    app: nemo
    type: backend
    deployment: intent-api
spec:
  replicas: 3
  strategy:
    rollingUpdate:
      maxUnavailable: 1 # for pod anti-affinity to work
    type: RollingUpdate
  selector:
    matchLabels:
      pod: intent-api
  template:
    metadata:
      labels:
        app: nemo
        type: backend
        pod: intent-api
    spec:
      imagePullSecrets:
        - name: nemo-regcred
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: pod
                    operator: In
                    values:
                      - intent-api
              topologyKey: "kubernetes.io/hostname"
      containers:
        - name: django
          image: nemometaos/intent-api:v0.1.1 # {"$imagepolicy": "flux-system:intent-api"}
...

```

Figure 8: Example of the Deployment manifest

After that, any **ascending, new tag version**, that is created from the component repository, will be pushed to the docker hub repository and set inside the deployment manifest (version bump) as commit to the repository by FLUX CD.

2.2 Cloud/Edge/IoT Integration and Validation Infrastructure

2.2.1 OneLab Clusters for NEMO

Four distinct Kubernetes clusters have been established to fulfil specific operational requirements. These include one primary cluster designated for development workloads, two supporting clusters including a lightweight cluster deployed on Raspberry Pis⁹ and finally the production cluster optimized for handling tasks requiring Graphics Processing Unit (GPU) resources.

Each cluster consists of a series of nodes structured to ensure efficient operation. The master node is responsible for core functionalities such as application scheduling, scaling, and overarching cluster

⁹ <https://www.raspberrypi.com/>

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 23 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

management. Worker nodes are dedicated to executing tasks assigned by the master node, which include deploying containers and hosting applications. In a production environment, multiple worker nodes are typically utilized to provide redundancy and enhance service availability, thereby ensuring robust and uninterrupted operations. Additionally, each cluster is configured with a load balancer, which distributes incoming network traffic across the nodes to ensure optimal resource utilization, fault tolerance, and high availability of the services.

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: default
  namespace: metallb-system
spec:
  addresses:
  - 132.227.122.2-132.227.122.4
  - 132.227.122.83-132.227.122.88

---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: default
  namespace: metallb-system
spec:
  ipAddressPools:
  - default
```

Figure 9: Example of MetalLB¹⁰ configuration

2.2.1.1 NEMO Dev cluster

The development cluster consists of one control-plane node (k8smaster.onelab.eu) and five worker nodes (k8sworker1-5.onelab.eu), all in a *Ready* status. The Kubernetes specified versions ranging from v1.28.7 to v1.28.15.

| NAME | STATUS | ROLES | AGE | VERSION |
|----------------------|--------|---------------|------|----------|
| k8smaster.onelab.eu | Ready | control-plane | 277d | v1.28.7 |
| k8sworker1.onelab.eu | Ready | worker | 277d | v1.28.7 |
| k8sworker2.onelab.eu | Ready | worker | 277d | v1.28.7 |
| k8sworker3.onelab.eu | Ready | worker | 277d | v1.28.7 |
| k8sworker4.onelab.eu | Ready | worker | 40d | v1.28.15 |
| k8sworker5.onelab.eu | Ready | worker | 40d | v1.28.15 |

Figure 10: Dev cluster nodes

| Node name | Node Type | Specifications | Public IP |
|---------------------|-----------|---|----------------|
| k8smaster.onelab.eu | Master | CPU: 8 CPU Cores RAM: 16GB Storage: 140GB Ephemeral OS-Image: Ubuntu 22.04.4 LTS | 132.227.122.23 |

¹⁰ <https://metallb.universe.tf/>

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 24 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

| Node name | Node Type | Specifications | Public IP |
|----------------------|--------------------|---|----------------|
| | | Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | |
| k8sworker1.onelab.eu | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS-Image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | 132.227.122.66 |
| k8sworker2.onelab.eu | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS-Image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | 132.227.122.24 |
| k8sworker3.onelab.eu | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS-Image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | 132.227.122.59 |
| k8sworker4.onelab.eu | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS-image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | 132.227.122.41 |
| k8sworker5.onelab.eu | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB RAM Storage: 120GB Ephemeral + 150GB Ceph OS-image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-116-generic Container-runtime: containerd://1.6.28 | 132.227.122.47 |

Table 1: NEMO dev cluster (K8S)

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 25 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | FINAL |

| NAME | STATUS | AGE |
|-----------------------------|-------------|-------|
| argo | Active | 134d |
| cert-manager | Active | 277d |
| default | Active | 277d |
| flux-system | Active | 264d |
| ingress-nginx | Active | 267d |
| k3s-cluster | Active | 6d20h |
| k3s-onelab | Active | 34d |
| kube-flannel | Active | 271d |
| kube-node-lease | Active | 277d |
| kube-public | Active | 277d |
| kube-system | Active | 277d |
| kubernetes-dashboard | Active | 275d |
| l2sm-system | Active | 23d |
| linkerd | Active | 25d |
| linkerd-viz | Active | 25d |
| metallb-system | Active | 267d |
| nemo-ai | Active | 275d |
| nemo-demo | Active | 247d |
| nemo-kernel | Active | 275d |
| nemo-net | Active | 275d |
| nemo-ppef | Active | 275d |
| nemo-sec | Active | 275d |
| nemo-svc | Active | 275d |
| nemo-workloads | Active | 140d |
| open-cluster-management | Active | 182d |
| open-cluster-management-hub | Active | 182d |
| raspberrypi | Terminating | 168d |
| rasptest | Active | 86d |
| reflector | Active | 270d |
| rook-ceph | Active | 276d |
| test-cluster | Terminating | 69d |

Figure 11: Dev cluster namespaces

2.2.1.2 Staging 1 cluster

The staging cluster (Staging 1) comprises one control-plane node (nemo-s1-master) and three worker nodes (nemo-s1-worker1, nemo-s1-worker2, and nemo-s1-worker3), all reporting a *Ready* status, running Kubernetes versions v1.31.3 (control-plane) and v1.30.7 (workers).

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 26 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

| NAME | STATUS | ROLES | AGE | VERSION |
|-----------------|--------|---------------|-----|---------|
| nemo-s1-master | Ready | control-plane | 8d | v1.31.3 |
| nemo-s1-worker1 | Ready | worker | 8d | v1.30.7 |
| nemo-s1-worker2 | Ready | worker | 8d | v1.30.7 |
| nemo-s1-worker3 | Ready | worker | 8d | v1.30.7 |

Figure 12: Staging 1 cluster nodes

| Node Name | Node Type | Specifications | Public IP |
|-----------------|--------------------|--|-----------------|
| nemo-s1-master | Master | CPU: 8 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.104 |
| nemo-s1-worker1 | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.105 |
| nemo-s1-worker2 | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.106 |
| nemo-s1-worker3 | Worker and Storage | CPU: 16 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.107 |

Table 2: Staging 1 cluster (K8S)

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 27 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

| NAME | STATUS | AGE |
|----------------------|--------|-------|
| cert-manager | Active | 8d |
| default | Active | 8d |
| ingress-nginx | Active | 8d |
| kube-flannel | Active | 8d |
| kube-node-lease | Active | 8d |
| kube-public | Active | 8d |
| kube-system | Active | 8d |
| kubernetes-dashboard | Active | 7d22h |
| metallb-system | Active | 8d |
| rook-ceph | Active | 8d |

Figure 13: Staging 1 cluster namespaces

2.2.1.3 Staging 2 cluster (K3S)

The Staging 2), deployed on Raspberry Pis 4, consists of one control-plane node (nemo-k3s-master) and two worker nodes (nemo-k3s-node-1 and nemo-k3s-node-2), all in a *Ready* state and running Kubernetes version v1.30.6+k3s1.

| NAME | STATUS | ROLES | AGE | VERSION |
|-----------------|--------|----------------------|-----|--------------|
| nemo-k3s-master | Ready | control-plane,master | 36d | v1.30.6+k3s1 |
| nemo-k3s-node-1 | Ready | worker | 35d | v1.30.6+k3s1 |
| nemo-k3s-node-2 | Ready | worker | 35d | v1.30.6+k3s1 |

Figure 14: Staging 2 cluster nodes

| Node name | Node Type | Specifications | Public IP |
|-----------------|--------------------|--|----------------|
| nemo-k3s-master | Master | CPU: 4 CPU Cores RAM: 8GB Storage: 64GB External SSD OS Image: Ubuntu 24.10 Kernel Version: 6.11.0-1004-raspi Container-runtime: containerd://1.7.22-k3s1 | 132.227.122.99 |
| nemo-k3s-node-2 | Worker and Storage | CPU: 4 CPU Cores RAM: 8GB Storage: 1TB External SSD OS Image: Ubuntu 24.10 Kernel Version: 6.11.0-1004-raspi Container-runtime: containerd://1.7.22-k3s1 | 132.227.122.88 |

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 28 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

| Node name | Node Type | Specifications | Public IP |
|-----------------|--------------------|--|----------------|
| nemo-k3s-node-3 | Worker and Storage | CPU: 4 CPU Cores RAM: 8GB Storage: 64GB External SSD OS Image: Ubuntu 24.10 Kernel Version: 6.11.0-1004-raspi Container-runtime: containerd://1.7.22-k3s1 | 132.227.122.91 |

Table 3: Staging 2 Cluster (K3S)

| NAME | STATUS | AGE |
|--------------------------------------|--------|-----|
| 1875c8d8-d599-4869-ba2c-f7c4d1421833 | Active | 28d |
| cert-manager | Active | 36d |
| default | Active | 36d |
| falco | Active | 35d |
| flux-system | Active | 33d |
| kube-node-lease | Active | 36d |
| kube-public | Active | 36d |
| kube-system | Active | 36d |
| metallb-system | Active | 36d |
| nemo-kernel | Active | 34d |
| nemo-ppef | Active | 33d |
| nemo-sec | Active | 29d |
| nemo-svc | Active | 32d |
| nemo-workloads | Active | 33d |
| open-cluster-management | Active | 34d |
| open-cluster-management-agent | Active | 34d |
| open-cluster-management-agent-addon | Active | 34d |
| reflector | Active | 28d |
| rook-ceph | Active | 36d |

Figure 15: Staging 2 cluster namespaces

2.2.1.4 Production cluster (K8S)

The production cluster includes one control-plane node (nemo-prod-master), three worker nodes (nemo-prod-worker1, nemo-prod-worker2, and nemo-prod-worker3), and one GPU-enabled worker node (nemo-prod-gpu-worker), all in a Ready state and running Kubernetes version v1.30.7.

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 29 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | FINAL |

| Node Name | Node Type | Specifications | Public IP |
|----------------------|--------------------|---|-----------------|
| nemo-prod-master | Master | CPU: 4 CPU Cores RAM: 8GB Storage: 80GB Ephemeral OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.42 |
| nemo-prod-worker1 | Worker and Storage | CPU: 8 CPU Cores RAM: 16GB Storage: 250GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.3 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.43 |
| nemo-prod-worker2 | Worker and Storage | CPU: 8 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.113 |
| nemo-prod-worker3 | Worker and Storage | CPU: 8 CPU Cores RAM: 16GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.2 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.114 |
| nemo-prod-gpu-worker | Worker and Storage | CPU: 4 CPU Cores RAM: 8GB Storage: 120GB Ephemeral + 150GB Ceph OS Image: Ubuntu 22.04.4 LTS Kernel Version: 5.15.0-78-generic Container-runtime: containerd://1.7.12 | 132.227.122.115 |

Table 4: Production Cluster (k8S)

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 30 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

| NAME | STATUS | AGE |
|----------------------|--------|-----|
| cert-manager | Active | 9d |
| default | Active | 9d |
| ingress-nginx | Active | 9d |
| kube-flannel | Active | 9d |
| kube-node-lease | Active | 9d |
| kube-public | Active | 9d |
| kube-system | Active | 9d |
| kubernetes-dashboard | Active | 9d |
| metallb-system | Active | 9d |
| rook-ceph | Active | 9d |

Figure 16: Production cluster namespaces

2.3 Integration & V&V Methodology & Plan

NEMO will follow an agile and incremental approach of iteration cycles, grouped in 3 Phases, as depicted in Figure 17.

Phase 1: Baseline (M1-M18). Provides the initial NEMO Proof of Concept. Phase 1 starts with system, specification of the meta-OS Architecture and decomposition (WP1), design analysis, prototyping (WP2-WP4), integration, testing and validation of all key meta-OS components (WP4). The outcome will be *NEMO Ver. A* and initial Living Labs validation and the selection of the new consortium members and new components from Open Call #1 to be implemented with Phase 2.

Phase 2: Advance (M19-M30). All NEMO components are further developed (WP2-WP4), while NEMO is expanded with new functionality added from the new consortium members accepted via Open Call #1. Stronger integration with 5G networks and MANO systems will be realized and validated in Living Labs). The outcome will be *NEMO Ver. B* and Living Labs validation, along with new AIoT applications and services from Open Call #2.

Phase 3: Mature (M30-M36). Focus on validation and optimization, and more realistic field conditions testing and verification, not only from NEMO consortium but also from 3rd parties selected via Open Call #2, increasing system TRL and preparing NEMO Ver. 1.0, validated in Living Labs. This phase also strengthens activities related to engagement of open-source communities and relevant initiatives, ensuring accessibility, sustainability and availability in open-source platforms.

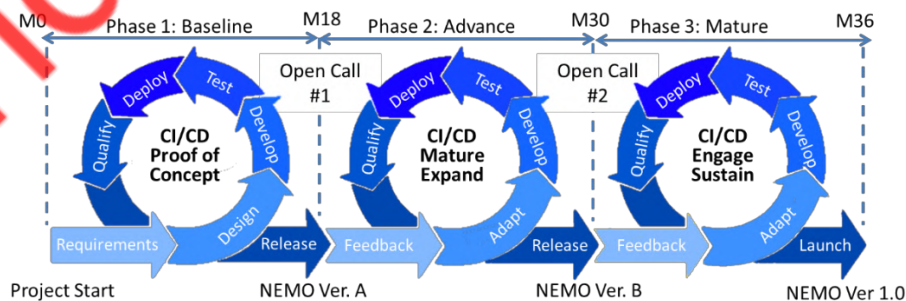


Figure 17: NEMO project phases and main meta-OS version releases

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 31 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

It should be underlined that each phase will follow an agile and incremental ***Continuous Integration/ Continuous Deployment/ Continuous Piloting (CI/CD/CP)*** approach, as explained in the previous subsections. The proposed approach allows responding to developments in the state of the art and emerging technology trends, as well as to continuously improve the results based on experimentation in the field.

2.3.1.1 NEMO Scenario-driven integration and verification

This section elaborates on the NEMO scenario-driven approach that is adopted by the project as the foundation of the integration activities that resulted into the first integrated NEMO platform.

The integration tests conducted are scenario-driven and each scenario covers a part of the integration workflow that are defined and described in section 4. The specified tests might be distinguished in bilateral, that is between component A and B, and/or system level cross-cutting ones.

The integration tests that are conducted follow the below presented structure. First, the scenario is defined. For this, the Scenario template presented below is used to specify the particular test. More specifically, the Scenario template incorporates details that pertain to the *objective* of the tests, the participating *components*, the *requirements* that are addressed, the *features* that are tested, the *steps* that are needed to be verified and finally the *test setup* which provides details on the integration setting that facilitates the test.

Then, the results are presented in detail as dictated by the steps that are defined in the scenario. Finally, the Checklist template is applied describing the successful or unsuccessful are reported and fall into the specific integration scenario. The stemming results for all the defined workflows are presented in section 5, following the abovementioned structure Scenario, *Outcome/Results*, *Checkpoints*.

| Test 1: | |
|------------------------|--|
| Objective | |
| Components | |
| Requirements alignment | |
| Features to be tested | |
| Test setup | |
| Steps | <ol style="list-style-type: none"> 1. 2. 3. |

Figure 18: Integration testing – Scenario template

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 32 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

Checklist for Test1

| | | Yes | No | Comments |
|---|--|-----|----|----------|
| 1 | Is a service created? | ✓ | | |
| 2 | Is the device registration completed successfully? | ✓ | | |
| 3 | Is the device sending its data successfully? | ✓ | | |
| 4 | Is the data stored in Database / Registry? | ✓ | | |

Figure 19: NEMO Integration testing - Checklist template (Example)

2.4 NEMO OneLab infrastructure deployments

The latest stable releases of the developed NEMO meta-OS components are deployed in the OneLab cluster as indicated in the following figures that depict the NEMO deployments in each namespace.

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|--|----|-------|----------|---------|-----|------|--------|--------|--------|--------|--------------|----------------------|------|
| external-dns-b65f9779-szgsr | ● | 1/1 | 0 | Running | 23 | 54 | 23 | 0 | 42 | n/a | 10.244.3.55 | k8sworker3.onelab.eu | 54d |
| falco-4jvjd | ● | 2/2 | 0 | Running | 94 | 119 | 94 | n/a | 33 | n/a | 10.244.3.227 | k8sworker3.onelab.eu | 42d |
| falco-5gcx9 | ● | 2/2 | 1 | Running | 87 | 96 | 87 | n/a | 18 | n/a | 10.244.2.178 | k8sworker2.onelab.eu | 42d |
| falco-5t8p | ● | 2/2 | 0 | Running | 77 | 117 | 77 | n/a | 13 | n/a | 10.244.1.197 | k8sworker1.onelab.eu | 42d |
| falco-15f8j | ● | 2/2 | 0 | Running | 23 | 58 | 23 | n/a | 13 | n/a | 10.244.3.8 | k8sworker3.onelab.eu | 42d |
| falco-n3jxp | ● | 2/2 | 48 | Running | 92 | 151 | 92 | n/a | 29 | n/a | 10.244.4.162 | k8sworker4.onelab.eu | 42d |
| falco-6krnz | ● | 2/2 | 2 | Running | 79 | 131 | 79 | n/a | 25 | n/a | 10.244.5.72 | k8sworker5.onelab.eu | 42d |
| keycloak-483d8757-hprmg | ● | 1/1 | 0 | Running | 3 | 2664 | n/a | n/a | n/a | n/a | 10.244.3.149 | k8sworker3.onelab.eu | 49d |
| nyrlease2-echo-server-5f7fd7b9f8-nbbdz | ● | 2/2 | 11 | Running | 1 | 44 | 1 | 0 | 34 | 32 | 10.244.1.143 | k8sworker1.onelab.eu | 246d |
| network-probe-daemonset-44dgc | ● | 1/1 | 0 | Running | 1 | 72 | n/a | n/a | n/a | n/a | 10.244.2.185 | k8sworker2.onelab.eu | 17d |
| network-probe-daemonset-44vcq | ● | 1/1 | 0 | Running | 1 | 72 | n/a | n/a | n/a | n/a | 10.244.5.89 | k8sworker5.onelab.eu | 17d |
| network-probe-daemonset-mrcj | ● | 1/1 | 0 | Running | 1 | 71 | n/a | n/a | n/a | n/a | 10.244.1.61 | k8sworker1.onelab.eu | 17d |
| network-probe-daemonset-tznjn | ● | 1/1 | 0 | Running | 1 | 70 | n/a | n/a | n/a | n/a | 10.244.3.232 | k8sworker3.onelab.eu | 17d |
| network-probe-daemonset-zzavc | ● | 1/1 | 0 | Running | 1 | 70 | n/a | n/a | n/a | n/a | 10.244.4.242 | k8sworker4.onelab.eu | 17d |
| network-worker-d8e7f536c-19pvc | ● | 1/1 | 2 | Running | 1 | 0 | n/a | n/a | n/a | n/a | 10.244.2.77 | k8sworker2.onelab.eu | 274d |
| nginx-deployment-7c79c4bf97-njq7p | ● | 1/1 | 2 | Running | 0 | 7 | n/a | n/a | n/a | n/a | 10.244.2.76 | k8sworker2.onelab.eu | 274d |
| nginx-deployment-7c79c4bf97-xtq9b | ● | 1/1 | 2 | Running | 0 | 7 | n/a | n/a | n/a | n/a | 10.244.1.219 | k8sworker1.onelab.eu | 246d |
| postgres-977c1cdd6-hnlmp | ● | 1/1 | 0 | Running | 1 | 68 | n/a | n/a | n/a | n/a | 10.244.3.148 | k8sworker3.onelab.eu | 49d |
| ubuntu-worker-6bb6bd74-st5h2 | ● | 1/1 | 2 | Running | 1 | 8 | n/a | n/a | n/a | n/a | 10.244.2.82 | k8sworker2.onelab.eu | 274d |

Figure 20: Default namespace

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|--|----|-------|----------|---------|-----|-----|--------|--------|--------|--------|--------------|----------------------|------|
| dash-metrics-solver-lstlw | ● | 1/1 | 2 | Running | 1 | 23 | 10 | 1 | 14 | 14 | 10.244.1.142 | k8sworker1.onelab.eu | 245d |
| dashboard-metrics-scraper-5657497c4c-mqz82 | ● | 1/1 | 2 | Running | 1 | 23 | n/a | n/a | n/a | n/a | 10.244.1.111 | k8sworker1.onelab.eu | 246d |
| kubernetes-dashboard-78f8dffc-98swg | ● | 1/1 | 7 | Running | 1 | 17 | n/a | n/a | n/a | n/a | 10.244.2.59 | k8sworker2.onelab.eu | 278d |

Figure 21: Kubernetes-dashboard namespace

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|--|----|-------|----------|---------|-----|------|--------|--------|--------|--------|-------------|----------------------|-----|
| i2sm-controller-5b7ed58678-4ck6z | ● | 1/1 | 0 | Running | 45 | 1703 | n/a | n/a | n/a | n/a | 10.244.4.96 | k8sworker4.onelab.eu | 25d |
| i2sm-controller-manager-5fc4798b0f-ckph5 | ● | 2/2 | 7 | Running | 4 | 38 | 26 | 0 | 29 | 14 | 10.244.5.15 | k8sworker5.onelab.eu | 25d |

Figure 22: I2SM namespace

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|---------------------------------------|----|-------|----------|---------|-----|-----|--------|--------|--------|--------|--------------|----------------------|-----|
| linkerd-destination-cc5f77f7-9d619 | ● | 1/4 | 1 | Running | 7 | 62 | n/a | n/a | n/a | n/a | 10.244.4.241 | k8sworker4.onelab.eu | 16d |
| linkerd-identity-7cfd4d4fb-v47qh | ● | 2/2 | 0 | Running | 2 | 16 | n/a | n/a | n/a | n/a | 10.244.4.187 | k8sworker4.onelab.eu | 16d |
| linkerd-proxy-injector-749955c4-bddx8 | ● | 2/2 | 0 | Running | 2 | 32 | n/a | n/a | n/a | n/a | 10.244.4.243 | k8sworker4.onelab.eu | 16d |
| metrics-apt-5b57cbbf7-spvv6 | ● | 2/2 | 0 | Running | 4 | 43 | n/a | n/a | n/a | n/a | 10.244.5.211 | k8sworker5.onelab.eu | 16d |
| tap-8sc77747f5-srzdw | ● | 2/2 | 0 | Running | 3 | 45 | n/a | n/a | n/a | n/a | 10.244.4.189 | k8sworker4.onelab.eu | 16d |
| tap-injector-575bc8558-k724q | ● | 2/2 | 0 | Running | 1 | 21 | n/a | n/a | n/a | n/a | 10.244.5.210 | k8sworker5.onelab.eu | 16d |
| web-766745ffd-v79kc | ● | 1/2 | 0 | Running | 1 | 13 | n/a | n/a | n/a | n/a | 10.244.4.188 | k8sworker4.onelab.eu | 16d |

Figure 23: Linkerd namespace

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|--|----|-------|----------|---------|-----|-----|--------|--------|--------|--------|--------------|----------------------|-------|
| deployment-controller-5d6d6c8fb-z56jc | ● | 1/1 | 0 | Running | 1 | 5 | n/a | n/a | n/a | n/a | 10.244.5.216 | k8sworker5.onelab.eu | 7d20h |
| lbc-599b99fcd-1mncf | ● | 1/1 | 2 | Running | 1 | 5 | n/a | n/a | n/a | n/a | 10.244.3.31 | k8sworker3.onelab.eu | 7d20h |
| migration-webhook | ● | 1/1 | 188 | Running | 0 | 0 | n/a | n/a | n/a | n/a | 10.244.4.69 | k8sworker4.onelab.eu | 7d20h |
| no-apt-56f555c35-z7254 | ● | 3/30 | 1 | Running | 0 | 0 | 0 | 0 | 0 | 0 | 10.244.4.30 | k8sworker4.onelab.eu | 86s |
| no-cluster-nginx-75459f84d7-sj2ns | ● | 1/1 | 0 | Running | 1 | 3 | 0 | 0 | 5 | 2 | 10.244.1.133 | k8sworker5.onelab.eu | 16d |
| no-mem-5d6455c35-5pnm2 | ● | 1/1 | 0 | Running | 1 | 4 | n/a | n/a | 4 | 4 | 10.244.3.215 | k8sworker3.onelab.eu | 43d |
| undeployed-controller-5c8c475ccc-72kvd | ● | 1/1 | 0 | Running | 1 | 5 | n/a | n/a | n/a | n/a | 10.244.4.176 | k8sworker4.onelab.eu | 8d |
| server-d8f1cddc4-fwv1c | ● | 1/1 | 1 | Running | 10 | 98 | 2 | 1 | 77 | 19 | 10.244.3.69 | k8sworker3.onelab.eu | 52d |

Figure 24: NEMO Kernel namespace

| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | IP | NODE | AGE |
|------------------------------|----|-------|----------|---------|-----|-----|--------|--------|--------|--------|--------------|----------------------|-----|
| i2sm-server-6554566af6-ckpvt | ● | 1/1 | 0 | Running | 0 | 2 | n/a | n/a | n/a | n/a | 10.244.3.3 | k8sworker3.onelab.eu | 57d |
| nncc-lbs-749f97566c-nghzq | ● | 1/1 | 0 | Running | 1 | 55 | n/a | n/a | n/a | n/a | 10.244.3.178 | k8sworker3.onelab.eu | 52d |
| nnex-54dd7b848-qcpz8 | ● | 1/1 | 0 | Running | 0 | 0 | 0 | 0 | 0 | 0 | 10.244.5.253 | k8sworker5.onelab.eu | 20s |

Figure 25: NEMO-net namespace

| | | | | | | | |
|----------------|---|----------------|----|----------|-------|-----------|-------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 33 of 146 | |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

| Pods (nemo-ppcf) [27] | | | | | | | | | | | | | |
|--|----|-------|----------|---------|-----|------|--------|--------|--------|-----------|-----------------|----------------------|------|
| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L IP | NODE | AGE | |
| kepler-b7nvw | ● | 1/1 | 33 | Running | 37 | 89 | n/a | n/a | n/a | n/a | 192.168.111.139 | k8sworker4.onelab.eu | 42d |
| kepler-cgss | ● | 1/1 | 2844 | Running | 39 | 152 | n/a | n/a | n/a | n/a | 192.168.111.148 | k8sworker3.onelab.eu | 169d |
| kepler-hg1lr | ● | 1/1 | 1 | Running | 23 | 157 | n/a | n/a | n/a | n/a | 192.168.111.144 | k8sworker5.onelab.eu | 42d |
| kepler-lsgfr | ● | 1/1 | 3 | Running | 40 | 44 | n/a | n/a | n/a | n/a | 132.227.122.66 | k8sworker1.onelab.eu | 169d |
| kepler-qmww | ● | 1/1 | 1 | Running | 12 | 33 | n/a | n/a | n/a | n/a | 192.168.111.146 | k8sworker2.onelab.eu | 169d |
| kepler-tqpw | ● | 1/1 | 1 | Running | 49 | 224 | n/a | n/a | n/a | n/a | 192.168.111.147 | k8sworker2.onelab.eu | 169d |
| netrics-server-699ffdc79-cx259 | ● | 1/1 | 26 | Running | 8 | 36 | 8 | n/a | 18 | n/a | 10.244.2.73 | k8sworker1.onelab.eu | 253d |
| proneheus-alertmanager-0 | ● | 1/1 | 9 | Running | 3 | 18 | n/a | n/a | n/a | n/a | 10.244.3.171 | k8sworker3.onelab.eu | 94d |
| proneheus-kube-state-metrics-7bd9b77c7-hshhf | ● | 1/1 | 4 | Running | 5 | 36 | n/a | n/a | n/a | n/a | 10.244.1.207 | k8sworker1.onelab.eu | 169d |
| proneheus-proneheus-node-exporter-2x5pw | ● | 1/1 | 5296 | Running | 25 | 10 | n/a | n/a | n/a | n/a | 192.168.111.148 | k8sworker3.onelab.eu | 170d |
| proneheus-proneheus-node-exporter-af9gn | ● | 1/1 | 14 | Running | 15 | 18 | n/a | n/a | n/a | n/a | 132.227.122.66 | k8sworker1.onelab.eu | 170d |
| proneheus-proneheus-node-exporter-5n7w2 | ● | 1/1 | 1 | Running | 1 | 13 | n/a | n/a | n/a | n/a | 192.168.111.146 | k8sworker2.onelab.eu | 170d |
| proneheus-proneheus-node-exporter-5vt7b | ● | 1/1 | 3 | Running | 1 | 16 | n/a | n/a | n/a | n/a | 192.168.111.144 | k8sworker5.onelab.eu | 42d |
| proneheus-proneheus-node-exporter-1j3gn | ● | 1/1 | 10 | Running | 19 | 15 | n/a | n/a | n/a | n/a | 192.168.111.147 | k8sworker2.onelab.eu | 170d |
| proneheus-proneheus-node-exporter-zw3k5 | ● | 1/1 | 43 | Running | 1 | 17 | n/a | n/a | n/a | n/a | 192.168.111.139 | k8sworker4.onelab.eu | 42d |
| proneheus-proneheus-pushgateway-7697dcd6-kz5pf | ● | 1/1 | 3 | Running | 1 | 17 | n/a | n/a | n/a | n/a | 10.244.1.224 | k8sworker1.onelab.eu | 104d |
| proneheus-server-685f467dbb-sfv8j | ● | 3/3 | 1 | Running | 125 | 2309 | n/a | n/a | n/a | n/a | 10.244.3.195 | k8sworker3.onelab.eu | 91d |
| sla-registry-f964d584-xjzvw | ● | 3/3 | 9110 | Running | 158 | 487 | n/a | n/a | n/a | n/a | 10.244.2.68 | k8sworker2.onelab.eu | 170d |
| sla-registry-postgresql-0 | ● | 2/2 | 6 | Running | 8 | 45 | 3 | n/a | 17 | n/a | 10.244.2.110 | k8sworker2.onelab.eu | 252d |
| sla-registry-redls-master-0 | ● | 2/2 | 53 | Running | 18 | 16 | n/a | n/a | n/a | n/a | 10.244.1.204 | k8sworker1.onelab.eu | 245d |
| sla-registry-worker-768cf5bb0-g4hwf | ● | 1/1 | 1 | Running | 3 | 341 | n/a | n/a | n/a | n/a | 10.244.1.203 | k8sworker1.onelab.eu | 105d |
| thanos-bucketweb-f9cc9f3cc-thc79 | ● | 1/1 | 0 | Running | 1 | 45 | n/a | n/a | n/a | n/a | 10.244.3.161 | k8sworker1.onelab.eu | 184d |
| thanos-compactor-5fca4db6b-pzkfq | ● | 1/1 | 1 | Running | 1 | 32 | n/a | n/a | n/a | n/a | 10.244.1.220 | k8sworker1.onelab.eu | 94d |
| thanos-query-7bc79897d-2mqx5 | ● | 1/1 | 0 | Running | 2 | 38 | n/a | n/a | n/a | n/a | 10.244.5.82 | k8sworker5.onelab.eu | 16d |
| thanos-query-frontent-7645dcf74-fxzdj | ● | 1/1 | 0 | Running | 1 | 14 | 1 | n/a | 10 | 7 | 10.244.4.43 | k8sworker4.onelab.eu | 9d |
| thanos-ruler-0 | ● | 1/1 | 0 | Running | 4 | 32 | n/a | n/a | n/a | n/a | 10.244.2.200 | k8sworker5.onelab.eu | 3d |
| thanos-storegateway-0 | ● | 1/1 | 0 | Running | 1 | 30 | 1 | 0 | 23 | 15 | 10.244.4.44 | k8sworker4.onelab.eu | 35d |

Figure 26: NEMO-PPEF namespace

| Pods(nemo-sec)[7] | | | | | | | | | | | IP | Node | Age |
|--|----|-------|----------|-----------|-----|-----|--------|--------|--------|------------------|----------------------|-------|-----|
| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L IP | NODE | AGE | |
| cha-aclm-http-solver-bddpl | ● | 1/1 | 0 | Running | 1 | 10 | 10 | 1 | 5 | 5 10.244.3.164 | k8sworker3.onelab.eu | 52d | |
| nemo-kong-kong-85f4f498d4-n6xpp | ● | 2/2 | 22 | Running | 18 | 497 | n/a | n/a | n/a | n/a 10.244.3.129 | k8sworker3.onelab.eu | 56d | |
| nemo-kong-kong-post-upgrade-nigrations-cp1qt | ● | 0/1 | 0 | Completed | 0 | 0 | n/a | n/a | n/a | n/a n/a | k8sworker5.onelab.eu | 6d23h | |
| nemo-kong-kong-pre-upgrade-nigrations-t0rwq | ● | 0/1 | 0 | Completed | 0 | 0 | n/a | n/a | n/a | n/a n/a | k8sworker2.onelab.eu | 6d23h | |
| nemo-kong-postgresql-0 | ● | 1/1 | 0 | Running | 10 | 31 | 4 | n/a | 12 | n/a 10.244.3.29 | k8sworker3.onelab.eu | 52d | |
| nemo-rabbitmq-0 | ● | 1/1 | 1 | Running | 18 | 231 | n/a | n/a | n/a | n/a 10.244.3.173 | k8sworker3.onelab.eu | 94d | |
| nginx-6076dcbbd-q5fpz | ● | 1/1 | 0 | Running | 0 | 7 | n/a | n/a | n/a | n/a 10.244.3.165 | k8sworker3.onelab.eu | 52d | |

Figure 27: NEMO-sec namespace

| Pods(nemo-svc) [36] | | | | | | | | | | | | | |
|--|----|-------|----------|-----------|-----|------|--------|--------|--------|--------|--------------|----------------------|------|
| NAME | PF | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L | NODE | AGE | |
| Intant-apl-58f64dff46-b2dsk | ● | 1/1 | 0 | Running | 5 | 1789 | n/a | n/a | n/a | n/a | 10.244.3.174 | k8sworker3.onelab.eu | 29d |
| Intant-apl-58f64dff46-w1mz | ● | 1/1 | 0 | Running | 5 | 1016 | n/a | n/a | n/a | n/a | 10.244.3.123 | k8sworker1.onelab.eu | 29d |
| Intant-apl-58f64dff46-z4w7b | ● | 1/1 | 0 | Running | 61 | 1719 | n/a | n/a | n/a | n/a | 10.244.4.79 | k8sworker4.onelab.eu | 29d |
| Intant-apl-Intant-collector-85cbd445bf-r4f6v | ● | 1/1 | 0 | Running | 10 | 99 | n/a | n/a | n/a | n/a | 10.244.5.118 | k8sworker5.onelab.eu | 29d |
| Intant-apl-worker-c54c7fff-hqoif | ● | 1/1 | 0 | Running | 3 | 511 | n/a | n/a | n/a | n/a | 10.244.5.117 | k8sworker5.onelab.eu | 29d |
| Intant-apl-workload-lifecycle-listener-7599c954f-4qcb2 | ● | 1/1 | 239 | Running | 1 | 110 | n/a | n/a | n/a | n/a | 10.244.4.80 | k8sworker4.onelab.eu | 29d |
| Intant-postgresql-0 | ● | 2/2 | 4 | Running | 14 | 53 | 5 | n/a | 20 | n/a | 10.244.2.108 | k8sworker2.onelab.eu | 272d |
| Intant-redls-master-0 | ● | 2/2 | 2 | Running | 21 | 19 | n/a | n/a | n/a | n/a | 10.244.1.158 | k8sworker1.onelab.eu | 43d |
| lpfs-0 | ● | 1/1 | 2188 | Running | 6 | 22 | n/a | n/a | n/a | n/a | 10.244.2.112 | k8sworker2.onelab.eu | 253d |
| nemo-celery-beat-c57cd5f70-hk4f7 | ● | 1/1 | 0 | Running | 1 | 165 | n/a | n/a | n/a | n/a | 10.244.5.136 | k8sworker5.onelab.eu | 9d |
| nemo-celery-flower-6555f7dc8d-dwz52 | ● | 1/1 | 0 | Running | 10 | 170 | n/a | n/a | n/a | n/a | 10.244.5.137 | k8sworker5.onelab.eu | 9d |
| nemo-celery-worker-86874cb3d6-rlb0h | ● | 1/1 | 0 | Running | 8 | 596 | n/a | n/a | n/a | n/a | 10.244.4.157 | k8sworker4.onelab.eu | 9d |
| nemo-cluster-parser-68674d4d0d-41sk8 | ● | 1/1 | 0 | Running | 1 | 145 | n/a | n/a | n/a | n/a | 10.244.5.144 | k8sworker5.onelab.eu | 9d |
| nemo-postgres-686c9cb4bf-454xc | ● | 1/1 | 0 | Running | 2 | 33 | n/a | n/a | n/a | n/a | 10.244.4.11 | k8sworker4.onelab.eu | 31d |
| nemo-redls-77fbd757cc-hf8ns | ● | 1/1 | 0 | Running | 7 | 4 | n/a | n/a | n/a | n/a | 10.244.5.143 | k8sworker5.onelab.eu | 15d |
| nemo-server-6d9bdcfd-n8n1 | ● | 1/1 | 0 | Running | 6 | 12 | n/a | n/a | n/a | n/a | 10.244.4.156 | k8sworker4.onelab.eu | 9d |
| nemo-workload-parser-798d44986c-dqgm7 | ● | 1/1 | 244 | Running | 1 | 141 | n/a | n/a | n/a | n/a | 10.244.5.139 | k8sworker5.onelab.eu | 9d |
| nemo-cluster-metrics-collector-28901588-kjnpd | ● | 0/1 | 0 | Completed | 0 | 0 | n/a | n/a | n/a | n/a | 10.244.5.254 | k8sworker5.onelab.eu | 40s |
| nemo-falco-7x5w | ● | 2/2 | 36 | Running | 66 | 113 | 66 | n/a | 22 | n/a | 10.244.3.169 | k8sworker3.onelab.eu | 248d |
| nemo-falco-89w1 | ● | 2/2 | 7 | Running | 51 | 69 | 11 | n/a | 31 | n/a | 10.244.2.87 | k8sworker2.onelab.eu | 248d |
| nemo-falco-gl659 | ● | 2/2 | 0 | Running | 74 | 91 | 4 | n/a | 17 | n/a | 10.244.5.73 | k8sworker5.onelab.eu | 42d |
| nemo-falco-k8s-metacollector-0df56fb649-45dsw | ● | 1/1 | 5 | Running | 8 | 41 | n/a | n/a | n/a | n/a | 10.244.2.75 | k8sworker2.onelab.eu | 253d |
| nemo-falco-n42z | ● | 2/2 | 6 | Running | 6 | 42 | 8 | n/a | 19 | n/a | 10.244.1.188 | k8sworker1.onelab.eu | 45d |
| nemo-falco-nx4rh | ● | 2/2 | 47 | Running | 62 | 118 | 62 | n/a | 21 | n/a | 10.244.4.160 | k8sworker4.onelab.eu | 42d |
| nemo-falco-sncnd | ● | 2/2 | 4 | Running | 11 | 53 | 13 | n/a | 10 | n/a | 10.244.0.7 | k8sworker1.onelab.eu | 248d |
| nemo-falco-decklet-sd7b5ddcd-dbnst | ● | 1/1 | 0 | Running | 1 | 22 | n/a | n/a | n/a | n/a | 10.244.1.167 | k8sworker1.onelab.eu | 43d |
| nemo-falco-decklet-sd7b5ddcd-sbp71 | ● | 1/1 | 0 | Running | 1 | 22 | n/a | n/a | n/a | n/a | 10.244.2.63 | k8sworker2.onelab.eu | 253d |
| nemo-falco-decklet-ul-ef777cf47c-zqzh6 | ● | 1/1 | 6 | Running | 1 | 11 | n/a | n/a | n/a | n/a | 10.244.3.12 | k8sworker3.onelab.eu | 76d |
| nemo-falco-decklet-ul-ef777cf47c-lspbx | ● | 1/1 | 0 | Running | 1 | 17 | n/a | n/a | n/a | n/a | 10.244.1.140 | k8sworker1.onelab.eu | 43d |
| nemo-falco-decklet-ul-redls-0 | ● | 1/1 | 4 | Running | 7 | 4 | 6624 | n/a | n/a | n/a | 10.244.2.111 | k8sworker2.onelab.eu | 253d |
| nemo-Intant-evaluator-28901588-4jrnj | ● | 0/1 | 0 | Completed | 0 | 0 | n/a | n/a | n/a | n/a | 10.244.5.2 | k8sworker5.onelab.eu | 40s |
| nemo-lcm-ul-7ff8f48b9-pxdns | ● | 1/1 | 0 | Running | 1 | 7 | n/a | n/a | n/a | n/a | 10.244.4.207 | k8sworker4.onelab.eu | 8d |
| nemo-quorum-node1-deployment-848c46dcfd-vcckk | ● | 2/2 | 0 | Running | 24 | 1361 | n/a | n/a | n/a | n/a | 10.244.4.183 | k8sworker4.onelab.eu | 14d |
| nemo-quorum-node2-deployment-57bcd769-nsq64 | ● | 2/2 | 1 | Running | 49 | 1265 | n/a | n/a | n/a | n/a | 10.244.5.249 | k8sworker5.onelab.eu | 14d |
| nemo-quorum-node3-deployment-0cf74bdcfd-fwx9w | ● | 2/2 | 1 | Running | 36 | 1273 | n/a | n/a | n/a | n/a | 10.244.5.250 | k8sworker5.onelab.eu | 14d |
| nemo-quorum-node4-deployment-769dcf959-udn7c | ● | 2/2 | 1 | Running | 68 | 1058 | n/a | n/a | n/a | n/a | 10.244.4.101 | k8sworker4.onelab.eu | 14d |

Figure 28: NEMO-svc namespace

The deployed workloads reside in a workloads' specific namespace as indicated in the figure below.

| Pods(nemo-workloads)[19] | | | | | | | | | | | Starting Memory Level | | |
|---|-----|-------|----------|--------|-----|-----|--------|--------|--------|--------------|-----------------------|----------------------|------|
| NAME | IP | READY | RESTARTS | STATUS | CPU | MEM | NCPU/R | NCPU/L | NMEM/R | NMEM/L IP | NODE | AGE | |
| demo-nginx-b87758879-e2bhz | 1/1 | 0 | Running | 1 | 6 | 2 | 2 | n/a | 4 | n/a | 10.244.3.86 | k8sworker3.onelab.eu | 56d |
| demo-nginx-b87758879-gshlc | 1/1 | 0 | Running | 1 | 2 | 2 | n/a | n/a | 1 | n/a | 10.244.1.206 | k8sworker1.onelab.eu | 56d |
| demo-nginx-b87758879-ws11l | 1/1 | 0 | Running | 1 | 2 | 2 | n/a | n/a | 1 | n/a | 10.244.2.167 | k8sworker2.onelab.eu | 56d |
| echo-server-computing-1-6ffdc6c55-6nxvt | 1/1 | 0 | Running | 1 | 24 | 3 | 233 | n/a | n/a | n/a | 10.244.3.233 | k8sworker3.onelab.eu | 17d |
| echo-server-computing-5ffdc6c55-6nxvt | 1/1 | 0 | Running | 1 | 29 | n/a | n/a | n/a | n/a | n/a | 10.244.2.184 | k8sworker2.onelab.eu | 18d |
| echo-server-demo-5fbd8f5cc-7a1 | 1/1 | 0 | Running | 1 | 25 | n/a | n/a | n/a | n/a | n/a | 10.244.4.202 | k8sworker4.onelab.eu | 37d |
| echo-server-emery-5cf6dd58f-6u18h | 1/1 | 3 | Running | 1 | 26 | n/a | n/a | n/a | n/a | n/a | 10.244.3.180 | k8sworker3.onelab.eu | 91d |
| echo-server-ingress-1-cf466c0b-571rh | 1/1 | 4 | Running | 1 | 27 | n/a | n/a | n/a | n/a | n/a | 10.244.3.222 | k8sworker3.onelab.eu | 71d |
| kva11s-echo-server-77f56d759-6m9w | 1/1 | 1 | Running | 1 | 1 | n/a | n/a | n/a | n/a | n/a | 10.244.1.222 | k8sworker1.onelab.eu | 142d |
| l2sn-switch-k8sworker1.onelab.eu-ffdc513c8-8z92 | 1/1 | 0 | Running | 7 | 11 | n/a | n/a | n/a | n/a | n/a | 10.244.1.33 | k8sworker1.onelab.eu | 25d |
| l2sn-switch-k8sworker2.onelab.eu-ffdc513c8-8z92 | 1/1 | 0 | Running | 8 | 12 | n/a | n/a | n/a | n/a | n/a | 10.244.2.161 | k8sworker2.onelab.eu | 25d |
| l2sn-switch-k8sworker3.onelab.eu-ffdc513c8-8z92 | 1/1 | 0 | Running | 7 | 12 | n/a | n/a | n/a | n/a | n/a | 10.244.3.211 | k8sworker3.onelab.eu | 25d |
| l2sn-switch-k8sworker4.onelab.eu-ffdc513c8-8z92 | 1/1 | 0 | Running | 8 | 10 | n/a | n/a | n/a | n/a | n/a | 10.244.4.97 | k8sworker4.onelab.eu | 25d |
| l2sn-switch-k8sworker5.onelab.eu-ffdc513c8-8z92 | 1/1 | 0 | Running | 7 | 10 | n/a | n/a | n/a | n/a | n/a | 10.244.5.16 | k8sworker5.onelab.eu | 25d |
| lakka-echo-server-5d5f7f75-9t0rh | 2/2 | 0 | Running | 1 | 23 | 1 | 0 | 18 | 17 | 10.244.1.93 | k8sworker1.onelab.eu | 11d | |
| lakka-echo-server-5d5f7f75-9t0rh | 2/2 | 0 | Running | 1 | 25 | 1 | 0 | 20 | 18 | 10.244.5.237 | k8sworker5.onelab.eu | 34d | |
| lakka-echo-server-dd80f7f7-8d9pt | 2/2 | 0 | Running | 1 | 25 | 1 | 0 | 19 | 18 | 10.244.4.123 | k8sworker4.onelab.eu | 11d | |
| pong | 0/0 | 0 | Running | 0 | 0 | n/a | n/a | n/a | n/a | n/a | 10.244.2.162 | k8sworker2.onelab.eu | 25d |
| | 1/1 | 0 | Running | 0 | 0 | n/a | n/a | n/a | n/a | n/a | 10.244.4.100 | k8sworker4.onelab.eu | 25d |

Infrastructure Management layer, the NEMO Kernel and the NEMO Service Management layer, including three NEMO cross-cutting functions (verticals) namely, the PPEF, the CFDRl and the Cybersecurity & Federated Access Control.

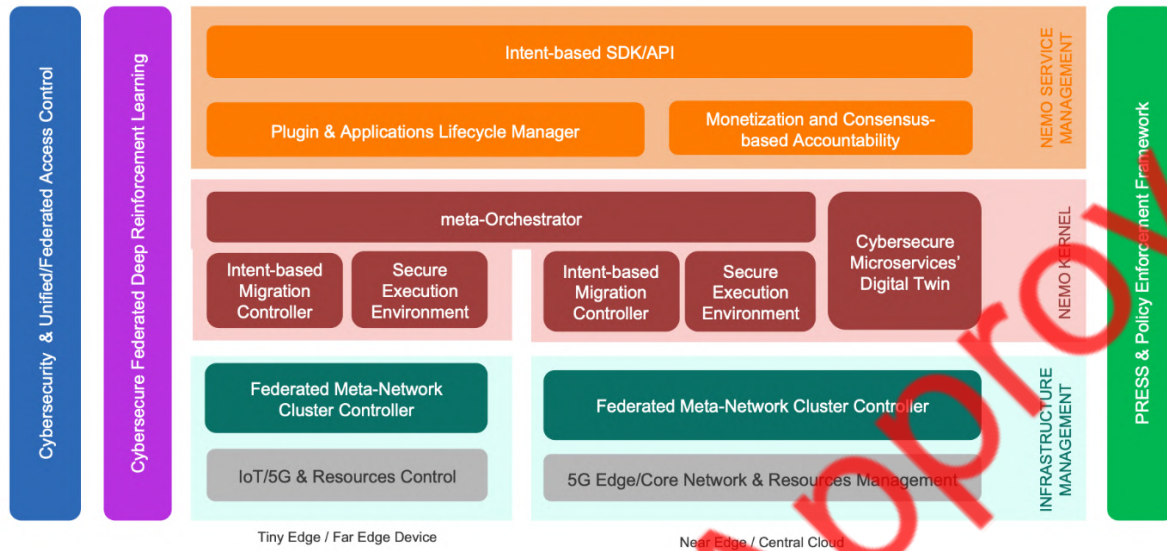


Figure 30: The NEMO high-level architecture

The first integrated NEMO meta-OS platform materialized through four scenario-driven integration activities that aimed glue together the first stable releases of the NEMO technical ecosystem. Figure 31 depicts the overview of the first integrated NEMO platform. The square rectangles denote the integrated components.

The continuous lines indicate that the respective components are fully integrated meaning that the exposed interfaces, the respective data models and the provided functionality that is included in their latest stable release has been tested in the NEMO OneLab infrastructure and the communication between the participating modules has been successfully verified.

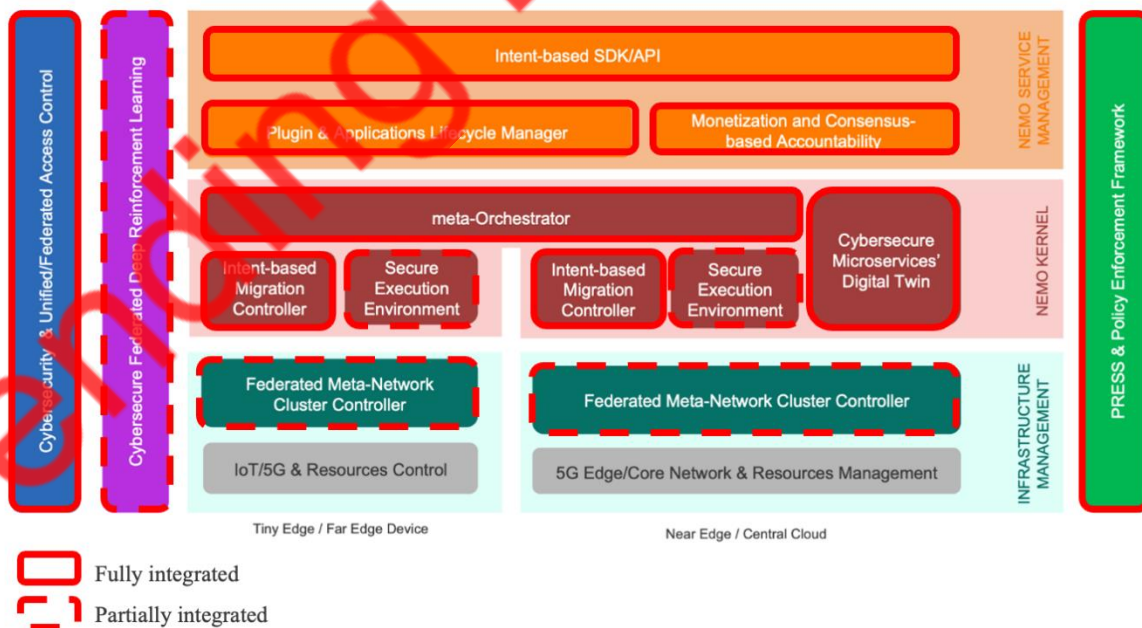


Figure 31: The 1st integrated version of NEMO meta-OS (high-level architecture view)

| | | | | | | | |
|----------------|---|----------------|----|----------|-------|-----------|-------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 35 of 146 | |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

On the other hand the dashed-line square rectangles indicate that the integration has been partially achieved. This means that for the particular components the integration within NEMO meta-OS has been achieved either through integration tests that were conducted in the context of partners premises or indicates that the integration was limited due to implementation activities that are currently ongoing. The results of the integration tests that were executed in view of the first release of the NEMO meta-OS are presented in detail in section 4 of the deliverable. The main goal of the current release is to demonstrate a good level of system-level coherence, verifying its readiness for the final integration activities that will be performed to produce the final version of the NEMO meta-OS.

2.5.1 Meta-OS functionality in NEMO v1

This section aims to provide an overview of the conducted integration activities and provided functionality of the current development state of the NEMO meta-OS components for each functional layer of the NEMO meta-OS architecture.

2.5.1.1 NEMO Infrastructure Management

The mNCC is the NEMO components that provides an abstraction layer of the underlying infrastructure network level. The mNCC delivers network orchestration overseeing network connectivity management guaranteeing multi-cluster and multi-domain connectivity. The incorporated connectivity adaptors that provide for data translation between different network protocols. More specifically, the *L2S-M* enables dynamic creation and management of isolated virtual networks within meta-OS operator clusters, the *5G adaptor*, supports deterministic communications through TSN bridges with 5G LAN solutions and the SDN-based connectivity adaptor, supports network management. Finally, mNCC component facilitates the monitoring of the communications between pods deployed in each cluster's nodes. The collected data are communicated through RabbitMQ.

Regarding integration activities, mNCC offered functionality is currently being finalized and initial integration activities with on premise deployments have been achieved and documented. The latest development updates of the component are presented in D2.3 [4] that will be submitted on M28.

2.5.1.2 NEMO Kernel

The NEMO components associated with the Kernel layer are the MO, the CMDT, the IBMC and the SEE as illustrated in the NEMO meta-OS architecture view figures above. The core functionalities offered from the NEMO Kernel components have been presented, demonstrated and documented in the present document through scenario driven integration activities (section 4). More specifically, the MO facilitates the workload deployment and migration processes. For the latter the IBMC supports the workload migration process ensuring efficient resource use, improved scalability, and continuous service availability during migrations. The CMDT provides enhanced workload monitoring related measurements to the platform, which are being consumed through the RabbitMQ, by the NEMO meta-OS components. Finally, the SEE (Kubernetes cluster) which is a solution for creating secure execution environments for critical and dynamic services, ensuring robust, secure, and efficient operations, is available and integrated within NEMO meta-OS ecosystem. The NEMO user (workload provider) can formulate a request through the Service Management layer asking for their services to be deployed in SEE. This is indicated by the proper configuration of workloads' intents.

2.5.1.3 NEMO Service Management

The NEMO Service Management layer components' functional updates are presented in detail in section 3 of the present document. More specifically, section 3 provides insights on components' architecture, provided functionality offered by the respective modules, their communication interfaces and associated data models, initial results and plans in view of the final version of the platform.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 36 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | FINAL |

The provided functionality from the NEMO Service Management layer components' namely, IBMC, MOCA, LCM and Intent-based API can be considered almost completed. The final releases of these components will be documented in D4.3 [1], due on M33, which will also describe the final integrated version of the NEMO meta-OS.

2.5.1.4 NEMO Cross-cutting Functions

With reference to the crosscutting functions, AI is vertically present in the metaOS architecture. The CFDRL component provides capacity of learning decision-making models capitalizing on the data collected by the NEMO meta-OS monitoring tool procedures offered by PPEF, CMDT and mNCC. The integration between the monitoring tools and CFDRL has been achieved allowing NEMO workload and cluster-level measurements to be digested by the CFDRL through RabbitMQ. The AI-assisted decision making that concern the workload optimal deployment and migration processes will be conducted in view of the final version of the NEMO meta-OS, as it requires the verified integration of the participating components for these workflows which has been achieved in the framework of this version (first) and is documented in section 4. The learning procedure in CFDRL combines two complementary learning paradigms: Federated Learning (FL) and Reinforcement Learning (RL). In addition, CFDRL to address privacy preservation challenges introduced FREDY (Federated Resilience Enhanced with Differential Privacy) [5] which integrates Flower with Private Aggregation of Teacher Ensembles (PATE) [6] to bolster privacy features. For the first release of the NEMO meta-OS the CFDRL component is considered as partially integrated.

Regarding Security in the NEMO meta-OS is built on the concept of ZeroTrust. NEMO Access Control (NAC) allows the implementation of a comprehensive approach to applying flexible, easily configurable, granular privileged access to NEMO resources by either internal components or, beyond the perimeter, to external entities. NAC provides a common secure API gateway for all the requests that are targeting NEMO meta-OS and offers access control based on a set of modular criteria, which may include identity management, catering for Authentication, Authorization, and Accounting (AAA). The NAC integration results in the context of the first release of the NEMO meta-OS are presented in section 4.

The NEMO meta-OS communication layer is based on RabbitMQ, a message broker enabling communication and synchronization among distributed systems and applications. It acts as an intermediary, facilitating secure message exchange while offering essential capabilities like message routing, queuing, and transformation.

Finally, the PPEF component facilitates service and resource monitoring for the NEMO meta-OS at both workload and cluster level. The PPEF concerns the metrics collection from the deployed monitoring tools, the evaluation between the collected measurements and the intents' expectation targets and the communication of this information within NEMO meta-OS (through RabbitMQ). The integration of the PPEF within NEMO meta-OS has been achieved and presented in section 4.4.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 37 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

3 NEMO Service Management Layer updates

This section reports the latest specifications and design options for the NEMO components of the Service Management layer in the NEMO architecture. These components provide a middleware between core NEMO functionality and workloads, but also end users. They support ZeroOps principles and expose interfaces to external entities (services or users). Moreover, the supported services include Lifecycle Management and DLT-based accountability of workload or infrastructure usage and collectively contribute to NEMO openness and adoption by third parties, referring to application or infrastructure owners, as well as developing entities.

3.1 Intent-based Migration Controller

3.1.1 Overview

The Intent-based Migration Controller (IBMC) serves as a key component within the NEMO ecosystem, it is designed to facilitate the migration of workloads across the IoT, Edge, and Cloud Continuum. By employing intent-based networking concepts, the IBMC ensures efficient resource use, improved scalability, and continuous service availability during migrations. This approach enables the IBMC to interpret and execute high-level migration intents, which supports the flexibility needed to manage the complexities of the meta-OS environment effectively.

3.1.2 Architecture

Figure 32 illustrates a simplified high-level architecture of the components underneath the Development View of the IBMC.

ibmc-controller: Is in charge of handling the communications with other components. This communication is performed by managing different RabbitMQ¹¹ queues and reading/delivering the correct messages needed for each migration step.

Velero¹²: Each Velero operation is defined as a custom resource using a Kubernetes Custom Resource Definition (CRD) and is stored in *etcd*. Velero also includes controllers responsible for processing these custom resources to handle backups, restores, and related tasks. This allows to backup or restore every resource in a cluster, with the capacity of selectively filter by resource type, namespace and/or label.

S3 Storage¹³: Dedicated to store the backups created by Velero, it's a key element for the migration process. The S3 bucket is located in the main NEMO cluster and every other cluster has access to it to allow the possibility of retrieving backups from one cluster to another.

¹¹ <http://www.rabbitmq.com/>

¹² <https://velero.io/>

¹³ <https://aws.amazon.com/s3/>

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 38 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: FINAL |

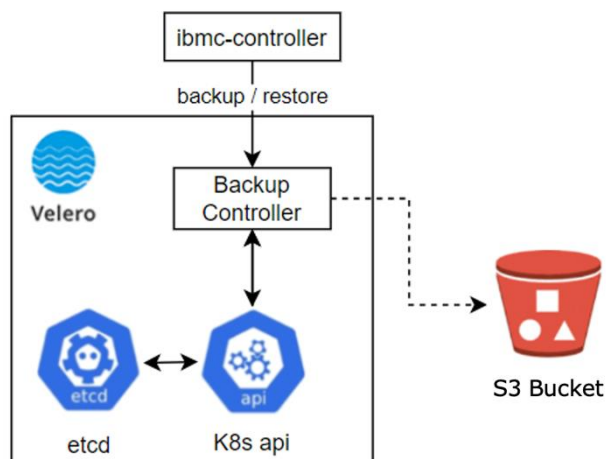


Figure 32: IBMC Simplified Architecture

Figure 33 shows a more complete architecture of the IBMC. In this figure, the migration process of a workload between two different clusters is represented, displaying the communication sequence since the migration is triggered.

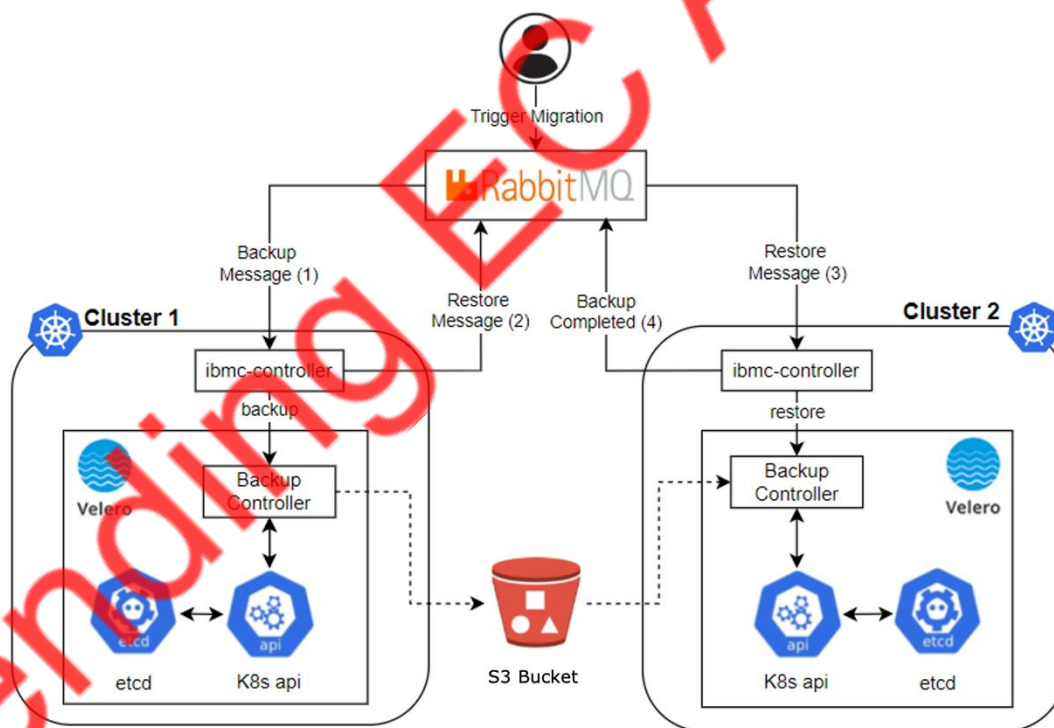


Figure 33: IBMC Complete Architecture

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 39 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | FINAL |

3.1.3 Interaction with other NEMO components

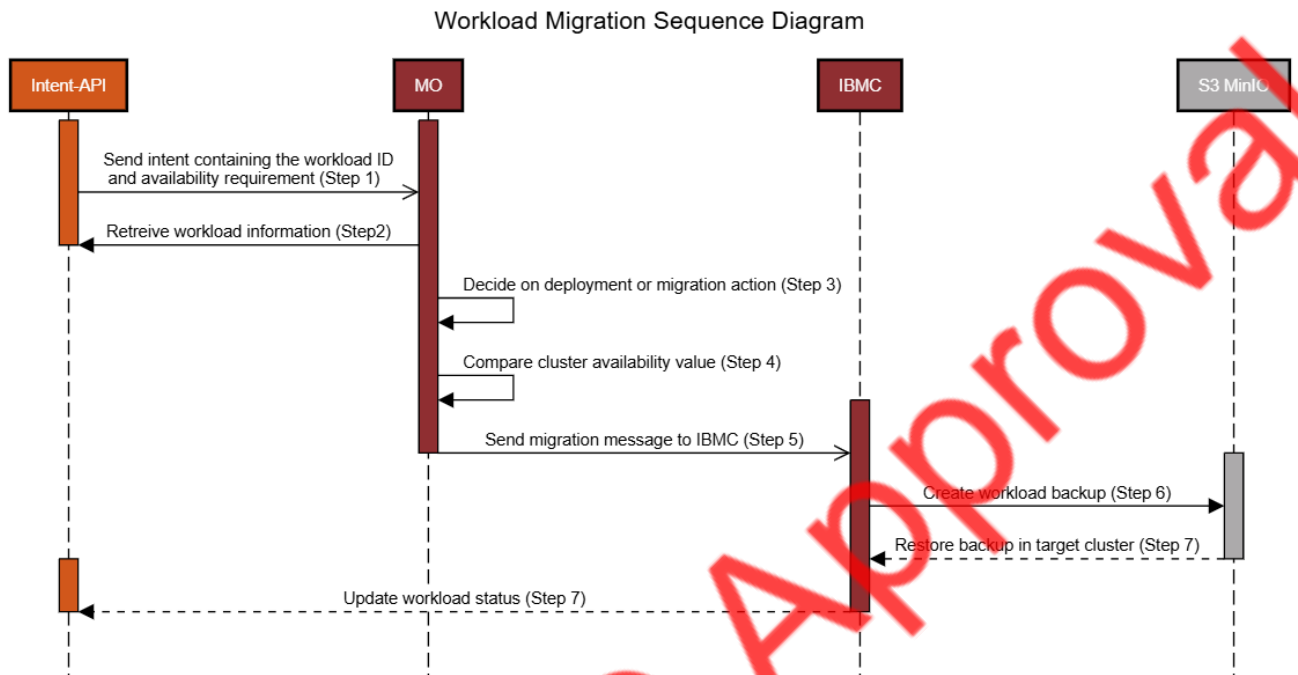


Figure 34: Migration Sequence Diagram

1. The Intent-API publishes a workload intent with an availability requirement.
2. The MO retrieves workload status from the Intent-API (deployed or not deployed).
3. If the workload is already deployed in any cluster, then a migration action is triggered.
4. The cluster availability where the workload is currently deployed is compared to the one from the intent. If $X < Y$, the workload migration is triggered.
5. The MO sends a message via RabbitMQ queue to the IBMC containing the workload ID and a target cluster that meets the availability requirement.
6. The IBMC creates a backup of the workload associated resources and uploads it to the S3 MinIO¹⁴ instance located in the OneLab main cluster.
7. The IBMC downloads the resources and restores them in the target cluster. After this, the workload is removed from the source cluster.
8. The IBMC sends a message to the Intent-API updating the workload status, specifying the cluster where it has been deployed.

¹⁴ <https://min.io/>

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 40 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: FINAL |

3.1.3.1 Meta-Orchestrator (MO)

The meta-Orchestrator (MO) plays a central role in the IBMC's workflow. When the MO receives a migration intent, it makes a placement decision based on cluster availability. This decision is sent to the IBMC and contains the target cluster for the migration.

3.1.3.2 Intent-Based API

The Intent-Based API stores all the information related to the workloads deployed. It is responsible of creating and sending the intents that trigger the migration of a workload.

When a migration is successfully completed, a message from the IBMC is sent to the Intent-Based API to update the workload status, including the new cluster where it has been deployed.

3.1.4 Initial results

The sections below provide a summary of the results generated through the utilization of the component.

3.1.4.1 Standalone results

An initial test of the standalone IBMC component was conducted in the OneLab environment. The experiment's setup included both OneLab clusters (the main cluster and the K3s cluster) with Velero pre-installed and configured. Both clusters had access to the MinIO¹⁵ instance deployed in the main cluster. Additionally, a workload was already deployed in the main cluster as part of the preconditions.

To simulate a migration scenario, a migration trigger was manually sent. This triggered the migration process, moving the workload from the main cluster to the K3s cluster. Upon completion, the migration was successful, with the workload fully deployed in the K3s cluster and removed from the main cluster.

3.1.4.2 Integration results

An end-to-end test was conducted involving the Intent-Based API, the Meta-Orchestrator and IBMC. The initial conditions were the same of the previous experiment, with the same objective of migrating a workload from the main cluster to the K3s cluster.

In this experiment, the process starts with the Intent-Based API posting an intent to RabbitMQ, which is read by the Meta-Orchestrator. The Meta-Orchestrator interprets the intent and verifies whether the workload ID specified in the intent is already deployed in any cluster. To obtain this information, the Meta-Orchestrator sends a request back to the Intent-Based API in order to retrieve the workload's deployment status.

If the workload is already deployed in a cluster, the Meta-Orchestrator compares the availability value of that cluster to the one specified in the intent. If the availability value of the current cluster is lower, the Meta-Orchestrator initiates a migration by posting a message to the RabbitMQ queue corresponding to the cluster where the workload is currently deployed. This message contains the information detailing the workload to be migrated (workload ID), the cluster where it is deployed and a new target cluster meeting the availability requirements.

When the IBMC controller receives this message, the migration process proceeds as in the initial experiment, resulting in the workload being successfully deployed in the K3s cluster and removed from the main cluster.

¹⁵ <https://min.io/>

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 41 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

3.1.5 Conclusion and roadmap

The Intent-Based Migration Controller (IBMC) within the meta-OS framework represents a major leap forward for the NEMO platform, enabling smooth workload migration across the IoT-to-Edge-to-Cloud continuum while sustaining a dynamic balance within the meta-OS environment.

Looking forward, the IBMC roadmap emphasizes ongoing refinement and adaptation to meet emerging needs and technological advancements within the meta-OS landscape. Planned enhancements and future directions are outlined to ensure the IBMC continues to lead in migration capabilities as the meta-Operating System ecosystem evolves.

3.2 Plugin & Applications Lifecycle Manager

Overview The Plugin & Applications Lifecycle Manager (LCM) is a versatile mechanism designed for unified, just-in-time management of plugins and applications throughout the NEMO ecosystem. Serving as the bridge between NEMO users and the ecosystem, the LCM enables seamless deployment of workloads, such as services, applications, and plugins, within the NEMO environment. It also supports over-the-air updates and bug fixes, ensuring the system remains up to date.

While workloads are running on the NEMO meta-OS, an event-driven mechanism monitors critical performance-related events. Additionally, a security controller oversees security events, notifies users of detected anomalies, and implements mitigation measures against identified cyber threats. The LCM's user interface will integrate with other NEMO components, including the Intent-based API, PPEF, MOCA and CMDT, to provide a comprehensive and cohesive user experience. The interfaces offered include user profiles, workload management and monitoring, security monitoring, and historical analysis tailored to the user's role.

3.2.1 Architecture

The LCM comprises of a set of subcomponents namely the LCM CD, LCM Controller, Security Controller, Event-based Response, LCM Repository and LCM Dashboard.

The LCM high-level architecture of NEMO meta-OS is depicted in the development view diagram in Figure 35.

LCM CD is based on ARGO CD framework to manage NEMO workloads provided by NEMO partners or NEMO Open Call participants and deploys workloads in S3 bucket container.

LCM Controller is a control mechanism that facilitates communication between LCM submodules and the NEMO ecosystem, offering endpoints for sending and receiving information.

Security Controller handles runtime security monitoring of NEMO workloads, notifying both users and relevant NEMO components of detected events.

Event-based Response module is designed to implement automated actions in response to events initiated by user input or detected by other NEMO components.

LCM Repository is used to store data related to workload lifecycle, security incidents, detected events and other workload related information to provide historical analysis and runtime statuses.

LCM Dashboard serves as the gateway between end-users and the NEMO meta-OS ecosystem, granting privileged users access to manage their workloads and monitor both performance and security.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 42 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

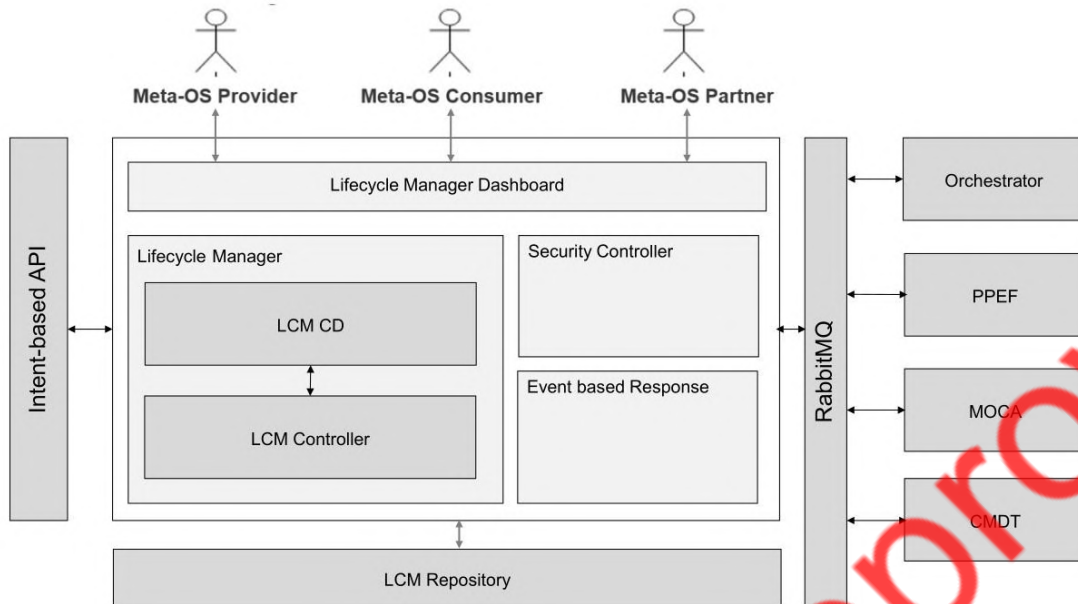


Figure 35: LCM Architecture

3.2.2 Lifecycle Manager

3.2.2.1 LCM CD

LCM CD, which corresponds to LCM Continuous Deployment, automates workloads deployment by ensuring that the state of applications in a Kubernetes cluster matches the configurations stored in Git repositories. Its key strength lies in the declarative approach to application definition, enabling users to define Kubernetes manifests in a version-controlled format.

Figure 36 provides a description of the payload transmitted while a plugin is being deployed.

```
payload = {
  "metadata": {
    "name": self.application_name,
    "namespace": "argocd"
  },
  "spec": {
    "project": "default",
    "source": {
      "repoURL": self.gitlab_repo_url,
      "chart": self.helm_chart_path,
      "targetRevision": "HEAD",
      "helm": {
        "valueFiles": ["values.yaml"]
      }
    },
    "destination": {
      "server": "https://kubernetes.default.svc",
      "namespace": self.target_namespace
    },
    "syncPolicy": {
      "automated": {
        "prune": True,
        "selfHeal": True
      }
    }
  }
}
url = f"{self.ARGOCD_URL}/api/v1/applications"
```

Figure 36: Plugin Deployment

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 43 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

LCM CD is based on ArgoCD tool and provides features like automated synchronization, rollback options, and support for multi-environment deployments. With its user-friendly web interface and seamless Kubernetes integration, LCM CD simplifies the deployment lifecycle, enhancing collaboration, traceability, and overall efficiency.

3.2.2.2 LCM Controller

The LCM Controller contains the logic of the LCM component and interacts with the internal subcomponents as well as the other NEMO components to retrieve information in real-time and feed the user interface while stores meaningful information in the LCM storage for keeping the historical status and performance for further analysis. The LCM controller includes functions to subscribe and consume data from relevant RabbitMQ channels, API endpoints to communicate with other components and functions to store and retrieve data from the storage repository. As the LCM Controller interacts with the other NEMO components more details on functions and APIs used will be reported in Section 4.2.3.

3.2.2.3 LCM Repository

LCM Repository uses Elasticsearch¹⁶ for storing, searching, and analysing data provided by various NEMO components like Intent-based API, PPEF, MOCA and CMDT. Elasticsearch provides fast search responses and comes with extensive REST APIs for storing and searching the data. Stored data include the status and lifecycle of the workloads, security events detected, workload performance and resource usage.

3.2.2.4 Security Controller

The Security Controller caters for security monitoring at runtime regarding NEMO workloads and alerts both the user and relevant NEMO components for detected events. This component aims to complement the security validation checks made before deployment of workloads into NEMO clusters, such as scanning processes in the Continuous Integration workflow or in the images registries, as these validation checks take place prior to the containers' deployment and even block some deployments as a result of failing the security assessment. The Security Controller aims to identify security incidents which take place at containers' runtime and may refer either to events at the system call level or to vulnerabilities arising from software dependencies, known vulnerabilities and insufficient security configurations.

Falco¹⁷ framework was selected as the foundation for the development of the plugin that is available through Security Controller. Falco is an open-source, CNCF adopted, runtime security platform that allows you to detect and respond to suspicious behavior within containers and applications. Falco is deployed in OneLab premises, as illustrated in section 2.4.

Falco continuously monitors the deployed containers and generates security auditing events that are digested by the Security Controller and are handled by the LCM. The Security Controller is responsible for the filtration of security events and subsequently their mapping with deployed workloads that correspond to a NEMO user.

3.2.2.5 LCM Visualization

LCM visualization is the main interface of NEMO project providing the necessary interfaces for each NEMO user role to manage workloads and resources in NEMO meta-OS. The LCM visualizations aim to provide interfaces for seamless user experience with NEMO ecosystem available to experts and less experienced users. The target is to provide the relevant information for workloads and resources lifecycle, usage, and security in a compact format at different levels of detail (workload, resources, user,

¹⁶ <https://elastic.co/>

¹⁷ <https://www.falcoframework.com/>

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 44 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

system). Indicative screenshots are presented in section 3.2.4 containing the initial results of the LCM component based on the implementation and integration progress at the current phase.

3.2.3 Interaction with other NEMO components

The LCM component is deployed in the NEMO Onelab infrastructure, as shown in Figure 37.

| | | | | |
|---|-----|-----------|--------------|------|
| nemo-falco-7xsnm | 2/2 | Running | 36 (86d ago) | 240d |
| nemo-falco-89wz1 | 2/2 | Running | 7 (22d ago) | 240d |
| nemo-falco-gl659 | 2/2 | Running | 0 | 34d |
| nemo-falco-k8s-metacollector-6df56fb649-45dxw | 1/1 | Running | 5 (106d ago) | 245d |
| nemo-falco-m48zx | 2/2 | Running | 0 | 36d |
| nemo-falco-nx4rh | 2/2 | Running | 47 (34d ago) | 34d |
| nemo-falco-smcnd | 2/2 | Running | 4 (142d ago) | 240d |
| nemo-falcosidekick-5d7b65dcd5-dbm41 | 1/1 | Running | 0 | 36d |
| nemo-falcosidekick-5d7b65dcd5-skp71 | 1/1 | Running | 27 (21d ago) | 245d |
| nemo-falcosidekick-ui-6f777cf47c-2qzh6 | 1/1 | Running | 6 (3d9h ago) | 69d |
| nemo-falcosidekick-ui-6f777cf47c-lspbx | 1/1 | Running | 0 | 36d |
| nemo-falcosidekick-ui-redis-0 | 1/1 | Running | 4 (106d ago) | 245d |
| nemo-intent-evaluator-28890144-bcsh6 | 0/1 | Completed | 0 | 41s |
| nemo-lcm-ui-7ff8f848b9-pxdns | 1/1 | Running | 0 | 18h |
| nemo-quorum-node1-deployment-848c400cf0-vscbk | 2/2 | Running | 1 (601h ago) | 601h |

Figure 37: Onelab deployment LCM and Security Controller

The LCM interacts with Intent-based API and MOCA components through relevant API endpoints and consumes data from PPEF, CMDT and Security Controller through RabbitMQ exchanges. This section describes in more detail the functions and data models used to interact with other NEMO components. Figure 38 depicts the data model of the Intent-based API endpoints which are used for the management of the workloads.

```

GET /workload/
{
  name: string,
  version: string,
}

POST /workload/
{
  name: string,
  version: string,
  schema:{},
  type: string (chart),
  intents: Array [],
}

POST /workload/upload/
{
  file: file (tgz helm chart),
  name: string,
  version: string
}

POST /workload/{id}/template/
{
  release_name: string,
  namespace: string,
  values_override:{},
  include_crds:boolean,
  is_upgrade:boolean,
  no_hooks:boolean,
  ingress_enabled:boolean,
  intents:[
    {
      intent_type,
      service_start_time,
      service_end_time,
      targets:[

```

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. | | | Page: | 45 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

```

        target_name,
        target_condition,
        target_value_range
    ]
}

GET /workload/instance/
{
    instance_id: string
}

```

Figure 38: Intent-based API workload management

Figure 39 presents the data model of the Intent-based API endpoints which are used for the management of the intents.

```

GET /intent/
{
    intent_id:string
}

POST /intent/template/
{
    instance_id: string,
    intent_type: string,
    service_start_time: string,
    service_end_time: string,
    targets:[
        {
            target_name: string,
            target_condition: string,
            target_value_range: string
        }
    ]
}

```

Figure 39: Intent-based API intents management

Figure 40 presents the data model of the MOCA API for the resources provisioning.

```

GET /moca/cluster/retrieve
{
}

POST /moca/cluster/register
{
    cluster_name: string,
    cpus: number,
    memory: number,
    storage: number,
    availability: string,
    green_energy: string,
    cost: string,
    cpu_base_rate: number,
    memory_base_rate: number
}

```

Figure 40: MOCA Resource provisioning

Additionally, the LCM subscribes to RabbitMQ exchanges to retrieve real-time data from PPEF, Security Controller and CMDT.

In summary, the retrieved information includes cluster usage metrics from PPEF (CPU, RAM, and storage, etc.), security events identified by the Security Controller (both system-wide and per workload), and data from CMDT, which currently encompasses the number of workload replicas and network

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 46 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

related information including workload's response times. Finally, data consumed from RabbitMQ or API endpoints are stored in LCM internal repository for historical overview and detailed analysis.

Figure 41 depicts functions to retrieve data to/from LCM Repository.

```
GET /cluster/{id}/data/
{
    timestamp_from: string,
    timestamp_to: string,
    cluster_id: string
}
GET /workload_instance/{id}/data
{
    timestamp_from: string,
    timestamp_to: string,
    workload_id: string
}
GET /workload_security_events/{id}/data
{
    timestamp_from: string,
    timestamp_to: string,
    workload_id: string
}
GET /workload_CMDT/{id}/data
{
    timestamp_from: string,
    timestamp_to: string,
    workload_id: string
}
```

Figure 41: Searching LCM Repository

3.2.4 Initial results

The LCM provides interfaces for different services such as *Plugin Monitoring*, *Workload Monitoring*, *Intent Management* and *Resource Provisioning*. Figure 42 illustrates the homepage of LCM dashboard.

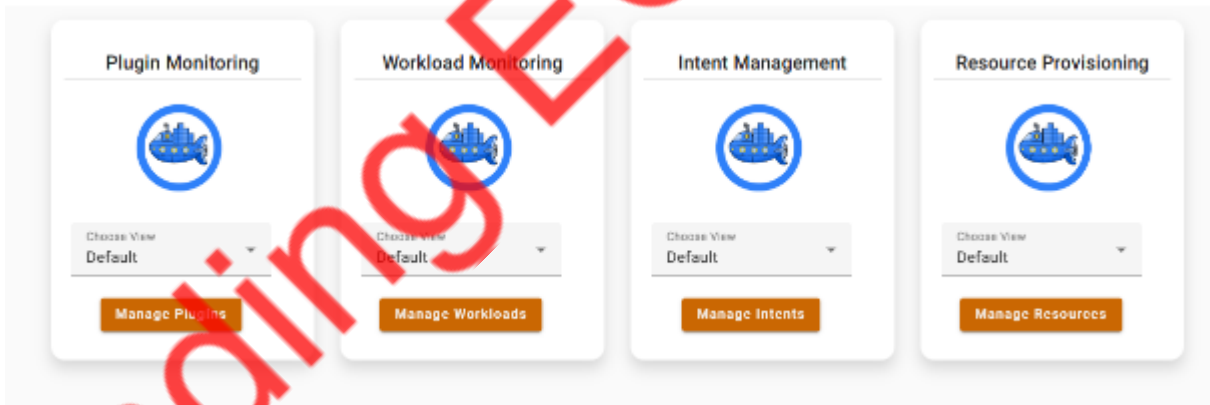


Figure 42: LCM Dashboard Homepage

Plugin Monitoring offers a CI/CD process and lifecycle monitoring for NEMO plugins and applications. Currently, the user is able to deploy a plugin and manage already deployed plugins while monitoring basic lifecycle parameters like versioning and activity are also available.

The *Workload Monitoring* section includes functionality to manage workloads in their whole lifecycle, from registering to Intent-based API to deployment and running several instances according to the user role and credentials. More details on the LCM UI available views are presented in Section 4 which concerns the NEMO scenario-driven integration and verification results.

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 47 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

3.2.5 Conclusion and roadmap

The achievements of LCM component, considering developments that are deployed in OneLab environment, fully integrated in NEMO meta-OS include:

- Workload management and monitoring
- Resources provisioning and monitoring
- Plugin deployment and lifecycle management
- Security/vulnerability scanning and monitoring

Towards the NEMO final version release, our goal is to deliver a comprehensive tool for managing and monitoring workload and resource lifecycle. This tool will consolidate information from various NEMO components into a streamlined format, ensuring a seamless experience for both expert and non-expert users within the NEMO ecosystem.

3.3 Monetization and Consensus-based Accountability

3.3.1 Overview

The Monetization and Consensus-based Accountability (MOCA) component enables the fair and secure monetization of the NEMO platform among the different users participating (consumers or providers). MOCA creates a distributed, tamper-resistant blockchain based ledger between different operators and verticals to track provenance and enforce secure negotiation and transaction of resources such as through smart contracts. The MOCA system provides the users with “credits” - the accountability unit of the platform. Their purpose is to reward the users who contribute to the platform by either providing infrastructure or deploying their services and allow them to accelerate precommercial exploitation of multi-tenant AIoT-Edge-Cloud continuum. The usage of DLT-based smart contracts for computing the accounting tasks allows for transparency and immutability in the transactions.

In retrospect, MOCA encompasses the following features:

- It uses blockchain technologies (more specifically Quorum) to perform the accounting actions and to transmit the results. This allows for immutability and traceability for all actions.
- It is integrated with NEMO’s Authentication platform; therefore, the users have role-based capabilities.
- It gives periodic but also real-time reports of the users’ information, like their bill details and the state of their resources (clusters, workloads).
- The accounting process takes into consideration the amount of the offered resources, the region demands and the infrastructure type to properly calculate the costs and rewards of each user.

3.3.2 Architecture

MOCA comprises of the following components:

- An **Event Server** that allows other components and users to retrieve information on the details of the registered resources (clusters and workloads) and the accounting events.
- The **Decentralized Applications (DApps)**, which contain the accounting logic and store information like the IPFS links to the cluster configuration files and the NEMO resources’ information.
- The **Smart Contracts component** (private Quorum blockchain), where DApps are deployed, and the transactions and calculations take place.
- An **IPFS** network, where the cluster configuration files are stored. Like Quorum, IPFS offers immutability to the data and detection of malicious attacks.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 48 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

Since the full description of the sub-components' functionality is provided in deliverable D4.1, the basic workflow will be presented, briefly. Figure 43 shows the sub-components that are part of the MOCA component. The *Event Server* is the main communication interface with the rest of the MOCA components which is responsible for handling (a) the requests for the cluster registration, (b) the communication with the *IPFS* for storing the cluster configuration files and (c) the exposure of the functionality of the *DApps* through REST API endpoints. The MOCA in periodic basis computes the resource usage of the deployed workloads and the usability status of the registered clusters. These periodic tasks communicate with the DApps and update appropriately the users' information held by the *Event Server*. A closer inspection on the calculation details is delivered in Section 3.3.4 and a full example of the workflow in Section 4.

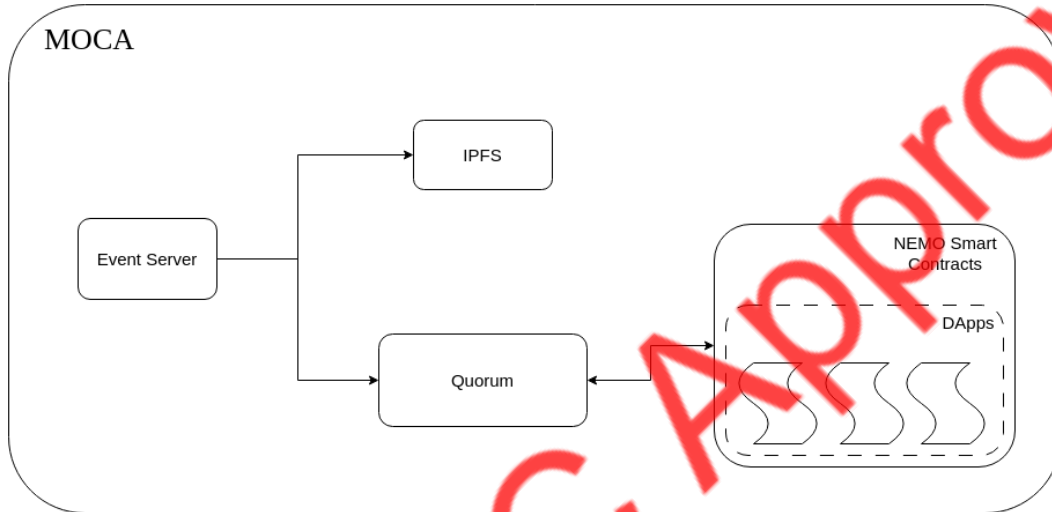


Figure 43: MOCA diagram

3.3.3 Interaction with other NEMO components

The MOCA component is deployed in the NEMO OneLab premises, as shown in Figure 44.

| | | | | |
|---------------------------------------|---|-----|------|---------|
| ipfs-0 | ● | 1/1 | 2072 | Running |
| moca-celery-beat-5984c5594c-c4tqv | ● | 1/1 | 0 | Running |
| moca-celery-flower-85f6858944-nqcbx | ● | 1/1 | 0 | Running |
| moca-celery-worker-fd654fcd6-bmvq2 | ● | 1/1 | 0 | Running |
| moca-cluster-parser-765db9b6fd-bkrl5 | ● | 1/1 | 0 | Running |
| moca-postgres-686c9cb4bf-4s4cx | ● | 1/1 | 0 | Running |
| moca-redis-77fdb757cc-jlms4 | ● | 1/1 | 0 | Running |
| moca-server-6548bfbfd9-schfr | ● | 1/1 | 0 | Running |
| moca-workload-parser-55ff9bd4f6-skcfj | ● | 1/1 | 0 | Running |

Figure 44: The MOCA deployment in the Onelab cluster

Figure 45 demonstrates the interactions of MOCA with the rest of the NEMO components. More specifically, MOCA integrates directly with the NEMO Intent API and NEMO RabbitMQ. Other NEMO components (LCM, PPEF, CMDT) can go through the Intent API, to access the MOCA Event Server endpoints. The RabbitMQ integration establishes the connection between MOCA and the NEMO Meta-Orchestrator. During the cluster registration, the two components exchange through the RabbitMQ the appropriate information to complete the action.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 49 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

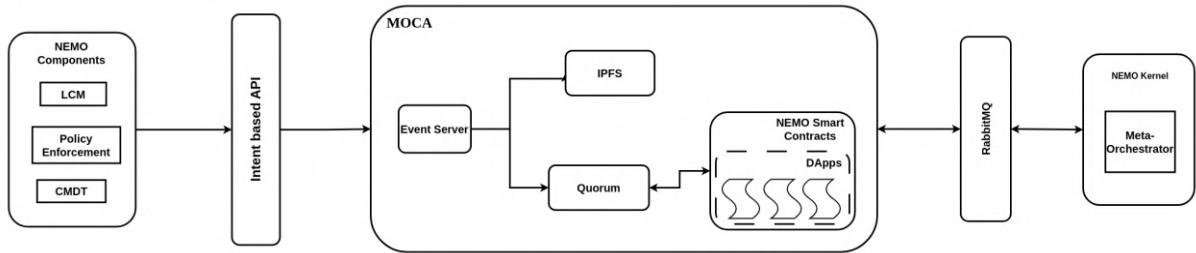


Figure 45: MOCA integration diagram

The following sections give a more thorough look on the MOCA API and the capabilities it offers.

Note: All the endpoints require to be authorized with the use of an authorization header, which sends a token provided from the *NEMO Access Control* plugin, as shown in the demonstration of Figure 46.

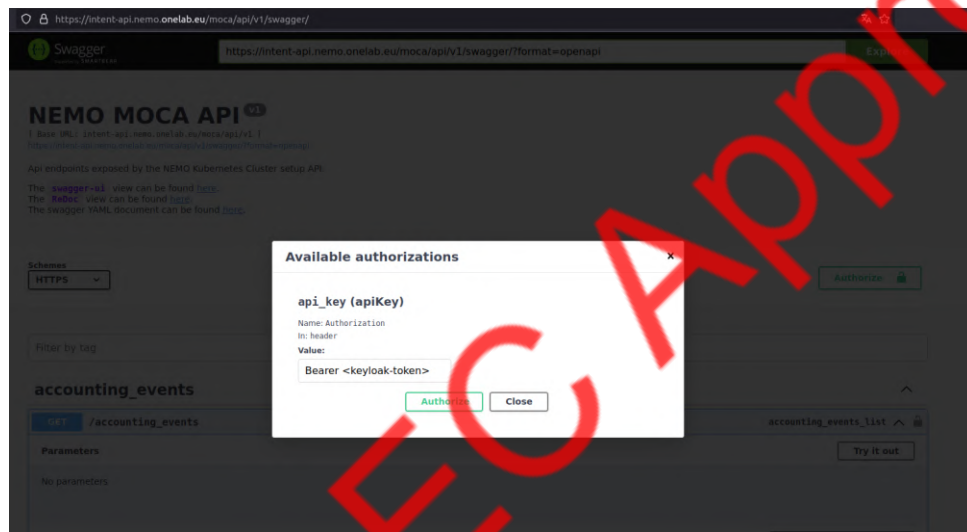


Figure 46: MOCA API authorization example

3.3.3.1 MOCA API

This section presents the MOCA API that is exposed and is available in openAPI format (<https://intent-api.nemo.onelab.eu/moca/api/v1/swagger/>). Figure 47, illustrates the relevant contents. The complete functionality offered by the MOCA API (endpoints and data models) is described in detail in ANNEX A.

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 50 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

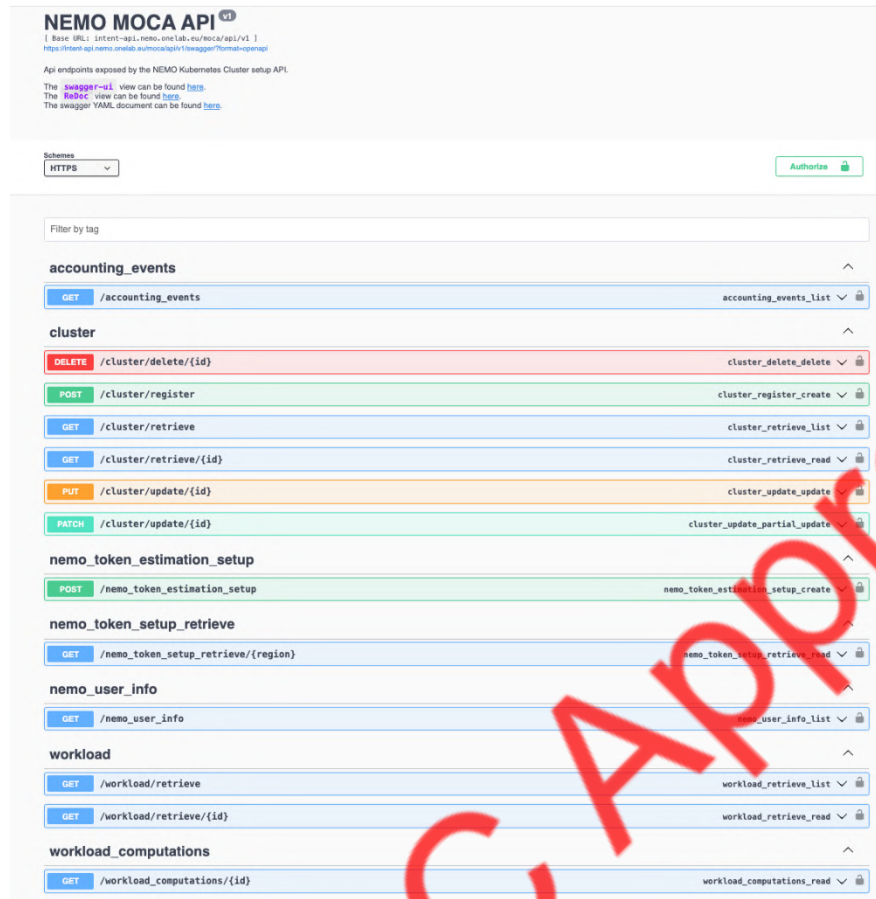


Figure 47: MOCA API

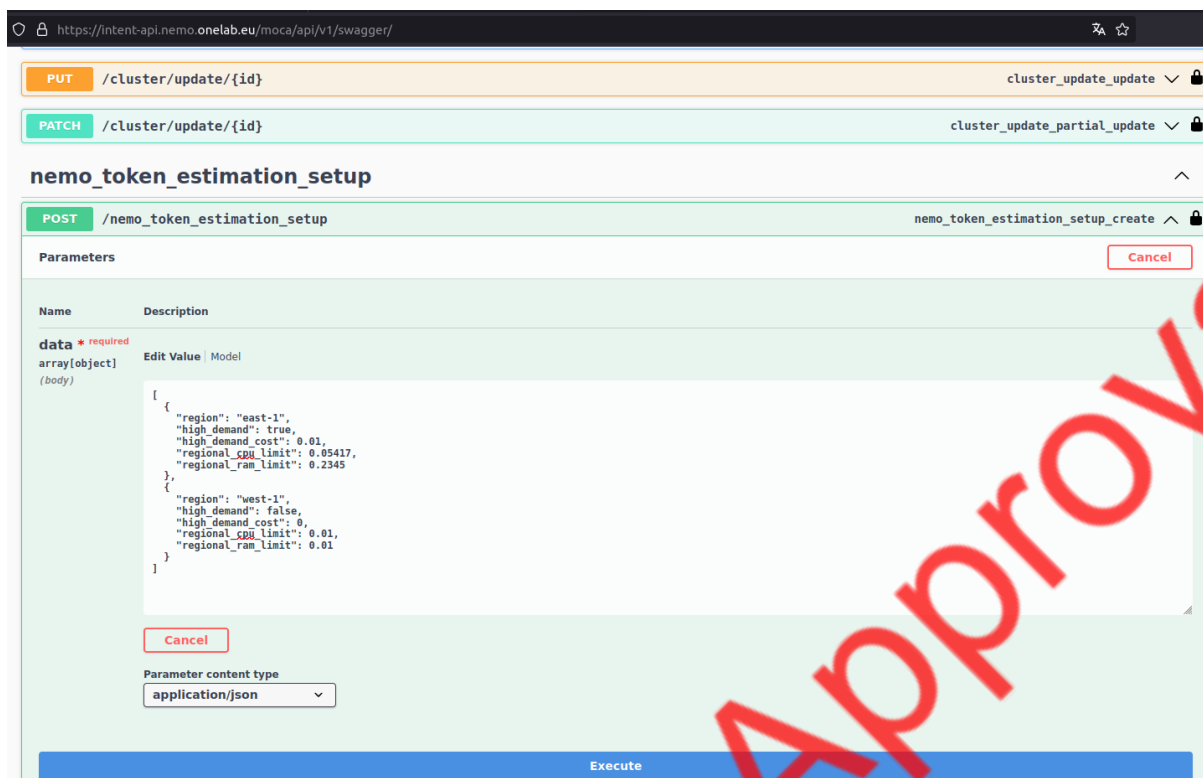
3.3.4 Initial result

MOCA uses various smart contracts to perform the accounting of the NEMO platform. In first version four contracts are supported namely the (a) *NemoTokenEstimation*, (b) *NemoFunds*, (c) *InfrastructureOwnerModel* and (d) *ServiceProviderModel*. In the following sections, a more thorough analysis of their functionality will be presented to better understand the calculation mechanism.

3.3.4.1 Regional Costs

The contract *NemoTokenEstimation* is responsible for supplying the costs for all the registered clusters to NEMO. A cluster can be categorized by whether it is in high/low demand, and its usability status in terms of available resources (CPU, RAM, Network bandwidth, etc.). The contract stores that information and allows for the retrieval of the details via the MOCA Event Server with the use of the */nemo_token_estimation_setup* endpoint (Figure 48).

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 51 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |



PUT /cluster/update/{id} cluster_update_update

PATCH /cluster/update/{id} cluster_update_partial_update

nemo_token_estimation_setup

POST /nemo_token_estimation_setup nemo_token_estimation_setup_create

Parameters

Name Description

data * required
array[object]
(body)

Edit Value | Model

```
{
  "region": "east-1",
  "high_demand": true,
  "high_demand_cost": 0.01,
  "regional_cpu_limit": 0.05417,
  "regional_ram_limit": 0.2345
},
{
  "region": "west-1",
  "high_demand": false,
  "high_demand_cost": 0,
  "regional_cpu_limit": 0.01,
  "regional_ram_limit": 0.01
}
}
```

Cancel

Parameter content type
application/json

Execute

Figure 48: Setup region information through Event Server

An example of a successful cluster registering information is shown in Figure 49.



```
[vm] from: 0x5b3...edd04 to: NemoTokenEstimationSetupContract.initializeNemoTokenEstimationInfo(string[],bool[],uint256[],uint256[],uint256[]) 0xae7...1a323 value: 0 wei
data: 0xc8f...186a8 logs: 0 hash: 0x57f...2bd64
```

status 0x1 Transaction mined and execution succeed

transaction hash 0x57f33042fba3550e4c7c6313a47556561e84a99296ca938e7d4cbe7f2bd64

block hash 0x831dc0bdc4c835e86029f9701e1b5c6310923f5722aa316f588d75c250372a2

block number 242

from 0x58380a6a701244545dcfc883fc8875f56bedd04

to NemoTokenEstimationSetupContract.initializeNemoTokenEstimationInfo(string[],bool[],uint256[],uint256[],uint256[]) 0xae742AB9883E1A4add3900c910F95e890031a323

gas 218199 gas

transaction cost 10728 gas

execution cost 165040 gas

input 0xc8f...186a8

output 0x

decoded input

```
{
  "string[] _region": [
    "east-1",
    "west-1"
  ],
  "bool[] _highDemand": [
    true,
    false
  ],
  "uint256[] _highDemandCost": [
    "1000000",
    "0"
  ],
  "uint256[] _regionalCpuCosts": [
    "5417000",
    "100000"
  ],
  "uint256[] _regionalMemoryCosts": [
    "23450000",
    "100000"
  ]
}
```

Figure 49: Logs of inserting cluster information

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 52 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

Table 5 shows the details of the contract.

| | | | | | |
|--|--|--|--|--|--|
| NemoTokenEstimationSetupContract.sol | | | | | |
| <pre> pragma solidity ^0.8.0; contract NemoTokenEstimationSetupContract { address private _owner; struct RegionInfo { bool isSet; bool highDemand; uint256 highDemandCost; uint256 regionalCpuCosts; uint256 regionalMemoryCosts; } mapping(string => RegionInfo) public regionalInfoMapping; modifier onlyOwner() { require(msg.sender == _owner, "Caller is not owner!"); _; } modifier validateRegionInfo(uint256 _regionLength, uint256 _highDemandLength, uint256 _highDemandCost, uint256 _regionalCpuCostsLength, uint256 _regionalMemoryCostsLength) { require(_regionLength == _highDemandLength && _regionLength == _highDemandCost && _regionLength == _regionalCpuCostsLength && _regionLength == _regionalMemoryCostsLength, "Input array lengths must match"); _; } constructor() { _owner = msg.sender; } function initializeNemoTokenEstimationInfo(string[] memory _region, bool[] memory _highDemand, uint256[] memory _highDemandCost, uint256[] memory _regionalCpuCosts, uint256[] memory _regionalMemoryCosts) public onlyOwner validateRegionInfo(</pre> | | | | | |

| | | | | | | |
|-----------------------|------|--|----|-----------------|--------------|----------------------|
| Document name: | | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 53 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

        _region.length,
        _highDemand.length,
        _highDemandCost.length,
        _regionalCpuCosts.length,
        _regionalMemoryCosts.length
    )
}
for (uint256 i = 0; i < _region.length; i++) {
    regionalInfoMapping[_region[i]] = RegionInfo({
        isSet: true,
        highDemand: _highDemand[i],
        highDemandCost: _highDemandCost[i],
        regionalCpuCosts: _regionalCpuCosts[i],
        regionalMemoryCosts: _regionalMemoryCosts[i]
    });
}
}
function isRegionSet(string memory _region) public view returns (bool) {
    RegionInfo memory info = regionalInfoMapping[_region];
    return info.isSet;
}
function getRegionInfo(
    string memory _region
) public view returns (bool, uint256, uint256, uint256) {
    RegionInfo storage info = regionalInfoMapping[_region];
    return (
        info.highDemand,
        info.highDemandCost,
        info.regionalCpuCosts,
        info.regionalMemoryCosts
    );
}
}
}

```

Table 5: The NemoTokenEstimation smart contract details

3.3.4.2 Handling transactions of clusters and workflows

The *NemoFunds* contract is responsible for keeping track of registered clusters and workflows, storing and emitting the transactions taking place and tracking the tokens available for every entity. When the usage of a workload is computed, the *NemoFunds* contract makes sure to appropriately change the balance of the actors involved (clusters, workflows, NEMO platform). Then, the changes become known to the Event Server though the use of events. Table 6 shows the details of the contract.

| | |
|--|--|
| NemoFunds.sol | |
| <pre> pragma solidity ^0.8.0; contract NemoFunds { address public owner; uint256 public nemoTotalBalance; uint256 public nemoActionsCounter; </pre> | |

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 54 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

```

uint256 public nemoRate;

enum TransactionType {
    deposit,
    withdrawal
}

struct NemoBalanceInfo {
    string customerId;
    uint256 customerTokens;
    TransactionType transactionType;
}

mapping(uint256 => NemoBalanceInfo) public nemoBalanceActions;
mapping(string => uint256) public customersBalance;
mapping(string => bool) public registeredCustomers;

event CustomerRegistered(
    string customerId,
    string customerType,
    uint256 balance,
    uint256 nemoBalanceActionId
);

event DepositTokens(
    string customerId,
    uint256 tokens,
    uint256 balance,
    uint256 nemoBalanceActionId
);

event WithdrawTokens(
    string customerId,
    uint256 tokens,
    uint256 balance,
    uint256 nemoBalanceActionId
);

modifier onlyOwner() {
    require(msg.sender == owner, "Caller is not the owner");
    _;
}

constructor() {
    nemoTotalBalance = 100000000000;
    nemoActionsCounter = 0;
    nemoRate = 20000;
}

function isCustomerRegistered(
    string memory customerId
) public view returns (bool) {
    return registeredCustomers[customerId];
}

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 55 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

}

function registerCustomer(
    string memory customerId,
    string memory _identifier
) public {
    require(
        !isCustomerRegistered(customerId),
        "The customer is already registered!"
    );

    if (
        keccak256(abi.encode(_identifier)) ==
        keccak256(abi.encode("ServiceProvider"))
    ) {
        registerServiceProvider(customerId);
    } else if (
        keccak256(abi.encode(_identifier)) ==
        keccak256(abi.encode("InfrastructureOwner"))
    ) {
        registerInfrastructureOwner(customerId);
    }
}

function registerServiceProvider(string memory customerId) private {
    registeredCustomers[customerId] = true;
    customersBalance[customerId] = 5000000000;
    nemoBalanceActions[nemoActionsCounter] = NemoBalanceInfo({
        customerId: customerId,
        customerTokens: customersBalance[customerId],
        transactionType: TransactionType.deposit
    });

    emit CustomerRegistered(
        customerId,
        "service",
        customersBalance[customerId],
        nemoActionsCounter
    );

    nemoActionsCounter++;
}

function registerInfrastructureOwner(string memory customerId) private {
    registeredCustomers[customerId] = true;
    customersBalance[customerId] = 10000000000;
    nemoBalanceActions[nemoActionsCounter] = NemoBalanceInfo({
        customerId: customerId,
        customerTokens: customersBalance[customerId],
        transactionType: TransactionType.deposit
    });
}

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 56 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |


```

});

emit CustomerRegistered(
    customerId,
    "infrastructure",
    customersBalance[customerId],
    nemoActionsCounter
);

nemoActionsCounter++;
}

function depositTokens(string memory customerId, uint256 tokens) public {
    require(nemoTotalBalance > tokens, "NEMO is out of funds!!!");

    nemoTotalBalance -= tokens;

    customersBalance[customerId] += tokens;

    nemoBalanceActions[nemoActionsCounter] = NemoBalanceInfo({
        customerId: customerId,
        customerTokens: customersBalance[customerId],
        transactionType: TransactionType.deposit
    });

    emit DepositTokens(
        customerId,
        tokens,
        customersBalance[customerId],
        nemoActionsCounter
    );

    nemoActionsCounter++;
}

function withdrawTokens(string memory customerId, uint256 tokens) public {
    require(
        customersBalance[customerId] > tokens,
        "Customer is out of funds!!!"
    );

    customersBalance[customerId] -= tokens;

    nemoTotalBalance += tokens;

    nemoBalanceActions[nemoActionsCounter] = NemoBalanceInfo({
        customerId: customerId,
        customerTokens: customersBalance[customerId],
        transactionType: TransactionType.deposit
    });
}

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 57 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

        emit WithdrawTokens(
            customerId,
            tokens,
            customersBalance[customerId],
            nemoActionsCounter
        );

        nemoActionsCounter++;
    }

    function nemoPayment(string memory customerId) public {
        require(
            customersBalance[customerId] > nemoRate,
            "Customer is out of funds!!!"
        );

        uint256 nemoPaymentFee = (customersBalance[customerId] * nemoRate) /
            10 ** 8;

        customersBalance[customerId] -= nemoPaymentFee;

        nemoTotalBalance += nemoPaymentFee;

        nemoBalanceActions[nemoActionsCounter] = NemoBalanceInfo({
            customerId: customerId,
            customerTokens: customersBalance[customerId],
            transactionType: TransactionType.withdrawal
        });

        emit WithdrawTokens(
            customerId,
            nemoPaymentFee,
            customersBalance[customerId],
            nemoActionsCounter
        );

        nemoActionsCounter++;
    }

    function getNemoBalance() public view returns (uint256) {
        return nemoTotalBalance;
    }

    function makeTransaction(
        string memory serviceId,
        string memory clusterId,
        uint256 _tokens
    ) public {
        withdrawTokens(serviceId, _tokens);
    }

```

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 58 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

        depositTokens(clusterId, _tokens);
        nemoPayment(clusterId);
    }
}

```

Table 6: The NemoFunds smart contract details

3.3.4.3 Cluster provision

The contract *InfrastructureOwnerModel* is responsible for registering the clusters joining NEMO. It communicates with the *NemoFunds* contract to complete the registration. The process provides the cluster with 10 tokens as an initialization sum. An example of the result of the registration of the cluster from the blockchain's side can be seen in Figure 50, where the transaction logs show the successful registration of the cluster. The logs show that an event was emitted (the *CustomerRegistered* event), to inform the Event Server of the action. It should be noted here that all the arithmetic values presented are normalized (multiplied with a constant variable 10^8), since Solidity¹⁸, the programming language for the smart contracts, cannot handle float values. Therefore, all the values are multiplied with a big enough constant to avoid issues with any float numbers.

```

[
  {
    "from": "0x150cA811e8fAA0cdE970133760Ed52AA82556EBb",
    "topic": "0xff55f5166e4d496b75baebf8fc225dfb9686dbad856b9b01e4d7dd83602fdd11",
    "event": "CustomerRegistered",
    "args": {
      "0": "cbcb208a-d535-434b-bb35-217a64bd516c",
      "1": "infrastructure",
      "2": "1000000000",
      "3": "0",
      "customerId": "cbcb208a-d535-434b-bb35-217a64bd516c",
      "customerType": "infrastructure",
      "balance": "1000000000",
      "nemoBalanceActionId": "0"
    }
  }
]

```

Figure 50: Example of the transaction logs of the cluster registration

A more thorough example of how the cluster registration is performed will be presented in Section 4 of the deliverable. Table 7 shows the details of the contract.

```

InfrastructureOwnerModel.sol

pragma solidity ^0.8.0;
import "./NemoFunds.sol";

contract InfrastructureOwnerModel {
    NemoFunds public nemoFunds;

    struct InfrastructureInfo {
        string cluster_name;
        uint256 totalCpu;
        uint256 totalMemory;
        uint256 totalDisk;
        string ipfsLink;
    }
}

```

¹⁸ <https://soliditylang.org/>

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 59 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

string availability;
string green_energy;
string cost;
uint256 cpu_base_rate;
uint256 memory_base_rate;
}

mapping(string => InfrastructureInfo) infrastructureInfo;

constructor(address _nemoFundsAddress) {
    nemoFunds = NemoFunds(_nemoFundsAddress);
}

function register(
    string memory clusterId,
    InfrastructureInfo memory info
) public {
    require(
        !nemoFunds.isCustomerRegistered(clusterId),
        "The customer is already registered!"
    );
    string memory _identifier = "InfrastructureOwner";
    nemoFunds.registerCustomer(clusterId, _identifier);

    infrastructureInfo[clusterId] = InfrastructureInfo({
        cluster_name: info.cluster_name,
        totalCpu: info.totalCpu,
        totalMemory: info.totalMemory,
        totalDisk: info.totalDisk,
        ipfsLink: info.ipfsLink,
        availability: info.availability,
        green_energy: info.green_energy,
        cost: info.cost,
        cpu_base_rate: info.cpu_base_rate,
        memory_base_rate: info.memory_base_rate
    });
}

function getInfrastructureInfo(
    string memory clusterId
)
    public
    view
    returns (
        string memory,
        uint256,
        uint256,
        uint256,
        string memory,
        string memory,
    )

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 60 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

        string memory,
        string memory,
        uint256,
        uint256
    )
{
    require(
        nemoFunds.isCustomerRegistered(clusterId),
        "The customer is not registered!"
    );

    InfrastructureInfo storage info = infrastructureInfo[clusterId];

    return (
        info.cluster_name,
        info.totalCpu,
        info.totalMemory,
        info.totalDisk,
        info.ipfsLink,
        info.availability,
        info.green_energy,
        info.cost,
        info.cpu_base_rate,
        info.memory_base_rate
    );
}
}

```

Table 7: The InfrastructureOwnerModel smart contract

3.3.4.4 Workload provision and usage calculation

The *ServiceProviderModel* contract is responsible for registering the NEMO workloads joining NEMO and calculating their impact on the cluster resources. It communicates with the *NemoFunds* contract to complete the registration and the *NemoTokenEstimation* to retrieve the costs associated with the regions. The registration process provides the workload with 5 tokens as an initialization sum.

An example transaction is available in Figure 51, where its logs show the emitted event with the registration's info.

```

[
  {
    "from": "0x15CcA811e8fAA0cdE970133760Ed52AA82556EBb",
    "topic": "0xff55f5166e4d496b75baebf8fc225dfb9686dbad856b9b01e4d7dd83602fdd11",
    "event": "CustomerRegistered",
    "args": {
      "0": "abcb208a-d535-434b-bb35-217a64bd516d",
      "1": "service",
      "2": "500000000",
      "3": "1",
      "customerId": "abcb208a-d535-434b-bb35-217a64bd516d",
      "customerType": "service",
      "balance": "500000000",
      "nemoBalanceActionId": "1"
    }
  }
]

```

Figure 51: Example of the transaction logs of the workload registration

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 61 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

The main function of the contract is to calculate the tokens that will be charged to the workload for the usage it made in a period, for example, 5 minutes. The next figures show the logs for the usage calculation, and it will be explained how these reflect on the involved actors (clusters, running workloads). Figure 52 shows how the balance of the workload was affected. In this simple example, for its usage of the cluster resources, it was charged 0,00001 tokens.

```
{
  "from": "0x15CcA811e8fAA0cdE970133760Ed52AA82556EBb",
  "topic": "0xfbb007563a8fb568bd9e39cc69606e59ec87d68bd711255899f01eb90d140e87",
  "event": "WithdrawTokens",
  "args": {
    "0": "abcb208a-d535-434b-bb35-217a64bd516d",
    "1": "1000",
    "2": "499999000",
    "3": "2",
    "customerId": "abcb208a-d535-434b-bb35-217a64bd516d",
    "tokens": "1000",
    "balance": "499999000",
    "nemoBalanceActionId": "2"
  }
},
```

Figure 52: Example of the transaction logs of the workload usage fee

These tokens are credited to the cluster, as shown in Figure 53.

```
{
  "from": "0x15CcA811e8fAA0cdE970133760Ed52AA82556EBb",
  "topic": "0x03ccd532b930745bb75b2c348177606dd9c4f5a3c07500bf1099f63a87c29438",
  "event": "DepositTokens",
  "args": {
    "0": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "1": "1000",
    "2": "1000001000",
    "3": "3",
    "customerId": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "tokens": "1000",
    "balance": "1000001000",
    "nemoBalanceActionId": "3"
  }
},
```

Figure 53: Example of the transaction logs of the cluster reward

Figure 54 shows that a 0.02% rate is rewarded to the NEMO account from the cluster's balance to reward NEMO with a small payment for the services provided.

```
{
  "from": "0x15CcA811e8fAA0cdE970133760Ed52AA82556EBb",
  "topic": "0xfbb007563a8fb568bd9e39cc69606e59ec87d68bd711255899f01eb90d140e87",
  "event": "WithdrawTokens",
  "args": {
    "0": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "1": "200000",
    "2": "999801000",
    "3": "4",
    "customerId": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "tokens": "200000",
    "balance": "999801000",
    "nemoBalanceActionId": "4"
  }
},
```

Figure 54: Example of the transaction logs of the NEMO fee paid by the cluster owner

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 62 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

Finally, Figure 55 gives a detailed account on the computation details (how many resources were used to justify the cost). In this example, the workload was charged 0,00001 tokens for utilizing 5,6 milliecycles of CPU and 0,235 MB RAM for a 5-minute time window that the metrics were collected.

The token calculations are examining the usage of the workload in question against the total resource usage of the deployment cluster, in order to reflect the real-time pressure on the cluster. Additionally, if the workload has exceeded the base resource limitations of the region it is assigned to, then that will be added to the total cost.

A more thorough example of how the workload usage calculation is performed end-to-end through MOCA will be presented in the next section.

```

{
  "from": "0x48185E8029eE9a253455e444aABC0F729540cf81",
  "topic": "0x6c87ac01bc017dd46ee0e6258ad463cbb81a50eec1813318c1ed13add55c8bf3",
  "event": "ServiceComputeTokens",
  "args": {
    "0": "abcb208a-d535-434b-bb35-217a64bd516d",
    "1": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "2": "560000000",
    "3": "23500000",
    "4": "1000",
    "serviceId": "abcb208a-d535-434b-bb35-217a64bd516d",
    "clusterId": "cbcb208a-d535-434b-bb35-217a64bd516c",
    "cpu": "560000000",
    "ram": "23500000",
    "tokens": "1000"
  }
}

```

Figure 55: Example of the transaction logs of the calculation of the workload usage fee

Table 8 shows the details of the contract.

| ServiceProviderModel.sol | |
|---|--|
| <pre> pragma solidity ^0.8.0; import "./NemoTokenEstimationSetupContract.sol"; import "./NemoFunds.sol"; contract ServiceProviderModel { NemoTokenEstimationSetupContract public nemoTokenEstimationSetup; NemoFunds public nemoFunds; address public owner; struct ServiceMetrics { string serviceId; string clusterId; string region; uint256 cpuUsage; uint256 memoryUsage; uint256 clusterCpuUsage; uint256 clusterMemoryUsage; } event ServiceComputeTokens(string serviceId, string clusterId, uint256 cpu, uint256 ram, </pre> | |

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 63 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

uint256 tokens
);

mapping(string => string[]) public ServiceProviderWorkflows;

constructor(
    address _nemoTokenEstimationSetupContractAddress,
    address _nemoFundsAddress
) {
    nemoTokenEstimationSetup = NemoTokenEstimationSetupContract(
        _nemoTokenEstimationSetupContractAddress
    );
    nemoFunds = NemoFunds(_nemoFundsAddress);
}

modifier checkRegistration(string memory serviceId) {
    require(
        !nemoFunds.isCustomerRegistered(serviceId),
        "The customer is already registered!"
    );
    _;
}

modifier checkRegionData(string memory region) {
    require(
        nemoTokenEstimationSetup.isRegionSet(region),
        "Data for region must be set before calling this function."
    );
    _;
}

function register(
    string memory serviceId
) public checkRegistration(serviceId) {
    string memory _identifier = "ServiceProvider";
    nemoFunds.registerCustomer(serviceId, _identifier);
}

function computeCredits(
    ServiceMetrics memory _metrics
) public checkRegionData(_metrics.region) {
    require(
        nemoFunds.isCustomerRegistered(_metrics.clusterId),
        "The cluster is not registered!"
    );
    require(
        nemoFunds.isCustomerRegistered(_metrics.serviceId),
        "The service is not registered!"
    );
}

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 64 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

(
    bool _highDemand,
    uint256 _highDemandCost,
    uint256 _regionalCpuCosts,
    uint256 _regionalMemoryCosts
) = nemoTokenEstimationSetup.getRegionInfo(_metrics.region);

string memory _serviceId = _metrics.serviceId;
string memory _clusterId = _metrics.clusterId;
uint256 _tokens = 0;

// CPU
uint256 _cpuTokens;
uint256 _cpuUsage = _metrics.cpuUsage * 10 ** 3;
uint256 _maxCpuUsage = _metrics.clusterCpuUsage;

//RAM
uint256 _ramTokens;
uint256 _ramUsage = _metrics.memoryUsage * 10 ** 3;
uint256 _maxRamUsage = _metrics.clusterMemoryUsage;

if (_cpuUsage > _regionalCpuCosts) {
    _cpuTokens = (_cpuUsage / _maxCpuUsage) * 10 ** 8;
} else {
    _cpuTokens = 0;
}

if (_ramUsage > _regionalMemoryCosts) {
    _ramTokens = (_ramUsage / _maxRamUsage) * 10 ** 8;
} else {
    _ramTokens = 0;
}

_tokens = _cpuTokens + _ramTokens;

if (_highDemand) {
    _tokens += _highDemandCost;
}

_tokens = _tokens / 1000;

nemoFunds.makeTransaction(_serviceId, _clusterId, _tokens);

emit ServiceComputeTokens(
    _serviceId,
    _clusterId,
    _cpuUsage,
    _ramUsage,
    _tokens
);

```

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 65 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

| |
|------------------|
| <pre> } } </pre> |
|------------------|

Table 8: The ServiceProviderModel smart contract

3.3.4.5 Accountability Service

In this section we present an example of how MOCA computes the tokens that will be charged for the resource usage of the deployment cluster end-to-end. For this example, we will use a workload deployed in the NEMO OneLab cluster (Figure 56).

```

Name:          lakka-echo-server-dd86f87f5-67b2h
Namespace:     nemo-workloads
Priority:       0
Service Account: lakka-echo-server
Node:          k8sworker5.onelab.eu/192.168.111.144
Start Time:    Fri, 08 Nov 2024 16:11:35 +0200
Labels:        app.kubernetes.io/instance=lakka-echo-server
               app.kubernetes.io/name=echo-server
               nemo.eu/workload=f306f303-2fe2-4534-911f-7d646936a3df
               pod-template-hash=dd86f87f5

```

Figure 56: OneLab workload details

This workload is first registered through the Intent-Based API and is deployed to the cluster through the Meta Orchestrator. After the successful deployment an event is published in the NEMO RabbitMQ which is consumed by MOCA. (Figure 57 shows for the workload of Figure 56 that the Meta Orchestrator has sent the payload which informs of the successful deployment.) Then, MOCA registers in the appropriate smart contract for the specific workload (for more details refer to section 3.3.4.4 [workload provision and usage calculation](#)) (Figure 58) and give the owner of the workload five initialization tokens (Figure 59). The registration event can also be viewed through the `/moca/api/v1/accounting_events` endpoint (Figure 60).

```

[*] Waiting for messages. To exit press CTRL+C
[x] Received {'workloadID': 'f306f303-2fe2-4534-911f-7d646936a3df', 'type': 'deployment', 'timestamp': '2024-11-08T16:11:35.000Z'}
[INFO] [{'id': 73, 'lifecycle_events': [{'id': 49, 'type': 'rendered', 'deployment_cluster': None, 'mgr': 'Meta Orchestrator'}]}]
The workload was registered transaction hash 0x7525af52cf9b4971a087e0ad5df5bba55ad986430d09120089c730f0f8addf60

```

Figure 57: RabbitMQ logs of workload deployment

```

Task app.tasks.nemo_register_workload[a7ddf82a-cc33-44a3-a3e3-5aabde553950] received
TaskPool: Apply <function fast_trace_task at 0x7f9202ca7ac0> (args=('app.tasks.nemo_register_workload', 'a7ddf82a-cc33-44a3-a3e3-5aabde553950', {'lang': 'py', 'task': 'register_workload'}))
[INFO] {'args': [{'customer_id': 'f306f303-2fe2-4534-911f-7d646936a3df', 'customer_type': 'service', 'balance': 500000000, 'nemo_balance_action_id': 1}], 'event': 'CustomerRegistered', 'timestamp': '2024-11-08T16:11:35.000Z'}
Task app.tasks.nemo_register_workload[a7ddf82a-cc33-44a3-a3e3-5aabde553950] succeeded in 5.749973624013364s: (1, '1d13e85e0cae3e4b84510135a489bcac7e2820a8288307')

```

Figure 58: Workload registration to blockchain

Request URL

https://intent-api.nemo.onelab.eu/moca/api/v1/nemo_user_info

Server response

Code

Details

200

Response body

```

{
  "username": "f306f303-2fe2-4534-911f-7d646936a3df",
  "balance": 5,
  "smart_contracts": [
    "NemoTokenEstimationSetupContract",
    "NemoFunds",
    "InfrastructureOwnerModel",
    "ServiceProviderModel"
  ]
}

```

Download

Response headers

Figure 59: User info for workload owner after registration

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 66 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

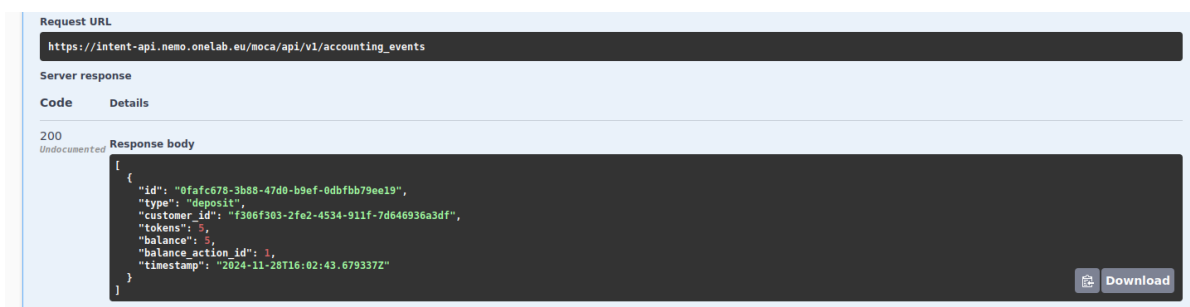


Figure 60: Accounting event for workload registration

At this point, it should be noted that the smart contracts are already deployed in the blockchain and ready to be executed when the right conditions are triggered (for example the registration of a workload that was examined before). The deployment of the contracts, at this point of the development, is performed with the help of the MOCA Helm chart available [here](https://gitlab.eclipse.org/eclipse-research-labs/nemo-project/nemo-service-management/monetization-and-consensus-based-accountability/moca/-/tree/main/bc-network?ref_type=heads)¹⁹. Figure 61 shows the Kubernetes jobs that are created to perform the deployment of the contracts and Figure 62 gives an example of the logs for the successful deployment of one of the contracts (in this case *NemoFunds*).

| | | |
|---|-----|-------------|
| nemo-quorum-node1-deploy-contracts-job-ngmr | 0/1 | 0 Completed |
| nemo-quorum-node1-deployment-848c466cf6-vscbk | 2/2 | 1 Running |
| nemo-quorum-node2-deploy-contracts-job-pzd8b | 0/1 | 0 Completed |
| nemo-quorum-node2-deployment-57bcd67769-qx6g4 | 2/2 | 1 Running |
| nemo-quorum-node3-deploy-contracts-job-ks6rk | 0/1 | 0 Completed |
| nemo-quorum-node3-deployment-6cf74dbcf-dfw9w | 2/2 | 1 Running |
| nemo-quorum-node4-deploy-contracts-job-b9qcd | 0/1 | 0 Completed |
| nemo-quorum-node4-deployment-769d4cf959-w6n7c | 2/2 | 1 Running |

Figure 61: Smart Contracts deployment through Helm chart

```

Logs(nemo-svc/nemo-quorum-node1-deploy-contracts-job-nk94v:nemo-quorum-node1-contracts-deployment)[5m]
Autoscroll:On FullScreen:OFF Timestamps:OFF Wrap:OFF

- Saving migration to chain.
> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0 ETH

2_migrate_handler.js
=====

Deploying 'NemoFunds'
> transaction hash: 0x4df7d4a6c422c0a4c01b1eb910b3c894afb7943b6608734579ffe57f7e4abaf9
- Blocks: 0 Seconds: 0
- Blocks: 1 Seconds: 4
> contract address: 0xa3EFdb3f086420cc463fa1C73BBF2B262236B16F
> block number: 665785
> block timestamp: 1732886480
> account: 0x7067769234792fed187e8e9643656f657e67681E
> balance: 10000000000
> gas used: 0 (0x0)
> gas price: 0 gwei
> value sent: 0 ETH
> total cost: 0 ETH

- Saving migration to chain.
> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0 ETH

```

Figure 62: Logs of deployment of NemoFunds contracts

¹⁹ https://gitlab.eclipse.org/eclipse-research-labs/nemo-project/nemo-service-management/monetization-and-consensus-based-accountability/moca/-/tree/main/bc-network?ref_type=heads

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 67 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

In this example we are using the region “*eu-west-1*” of the Onelab cluster, in which we are going to register the region as high demand and charge a fee of 0.01 tokens and set the base utilization limit as 0.05417 milliseconds for the CPU and 0.2345 MBs for the memory. If these limits are exceeded, the workload will be charged accordingly (more details on the costing mechanism can be found in section 3.3.4.4 workload provision and usage calculation). Figure 63 and Figure 64 show the successful registration of the region to the blockchain though MOCA.

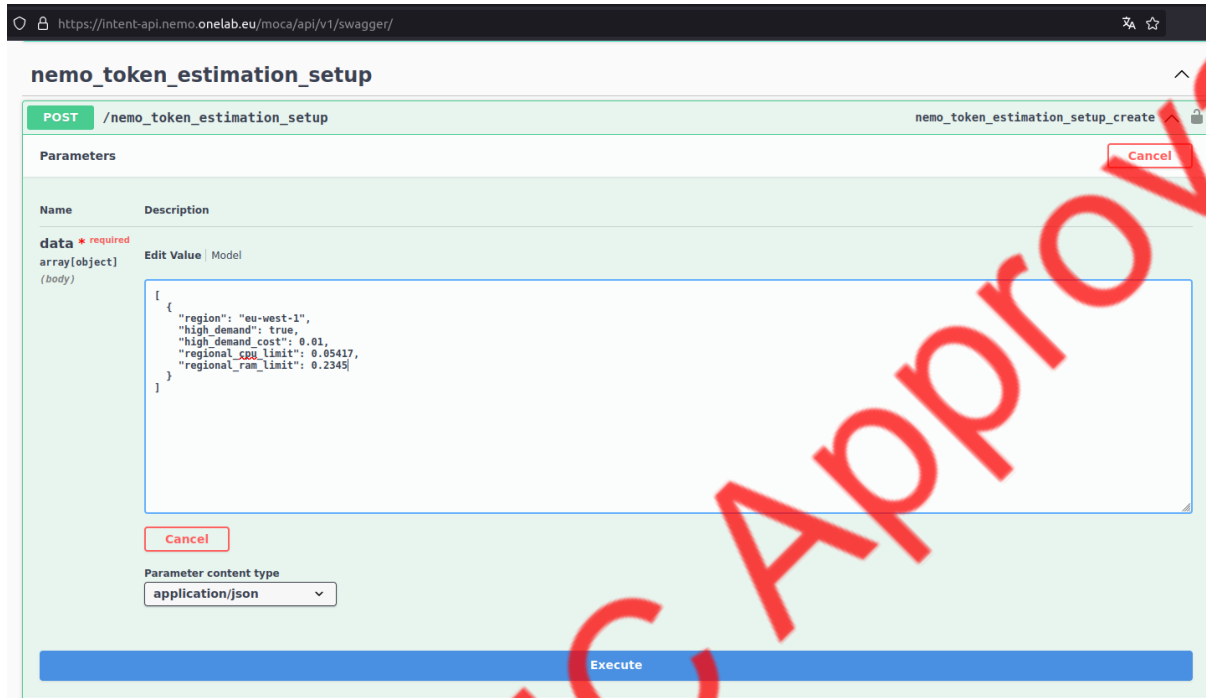


Figure 63: Registering NEMO OneLab Cluster regional info

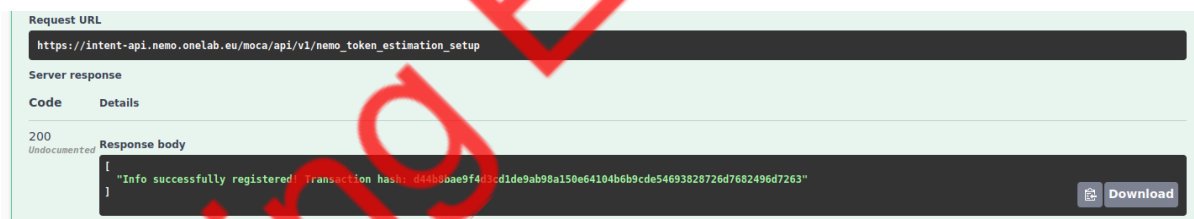


Figure 64: Response for successful registration

MOCA has in place automatic mechanisms that communicate with the PPEF component to acquire the resource usage of all the deployed workloads in periodic intervals (e.g. 5 minutes). Figure 65 shows the logs of MOCA, which has received from the appropriate smart contract the event with the calculation details.

```
ts:147] - [INFO] ('args': [{'serviceId': 'f306f303-2fe2-4534-911f-7d646936a3df', 'clusterId': '3022681b-35be-49b9-b663-37804e1661fc', 'cpu': 6240000000, 'ram': 274000000000}, {'serviceId': 'f306f303-2fe2-4534-911f-7d646936a3df', 'clusterId': '3022681b-35be-49b9-b663-37804e1661fc', 'cpu': 6240000000, 'ram': 274000000000}], 'task': 'Task app.tasks.nemo_workload_compute_tokens[7c1a5873-bba6-42c2-9f21-ca06f1a509b8] succeeded in 7.263470120728016s: (1, '23ea380582345e00ade0cb4a40c0fa218384759e', 'Task app.tasks.nemo_workload_compute_tokens[8f5ffc7d-d53d-4668-b4f7-5d225473b6eb] received', 'Taskpool: Apply <function fast_trace_task at 0x7f2864cabac0> (args:('app.tasks.nemo_workload_compute_tokens', '8f5ffc7d-d53d-4668-b4f7-5d225473b6eb', {'lang': 'py',
```

Figure 65: MOCA logs of the DApps component calculating the resource usage of a NEMO workload

Querying the endpoint for the accounting events activity (Figure 66), as the workload owner, we notice that the events hold information like the type of the transaction (deposit or withdraw), the workload ID

| | | | | | |
|----------------|---|----------------|----|----------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. | | | Page: | 68 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

(*customer_id*), the tokens the workload was charged depending on its usage (tokens), the state of the balance for the workload (balance), the ID of the accounting event as it is registered in the smart contract (*balance_action_id*) and the timestamp of the registration of the accounting event.

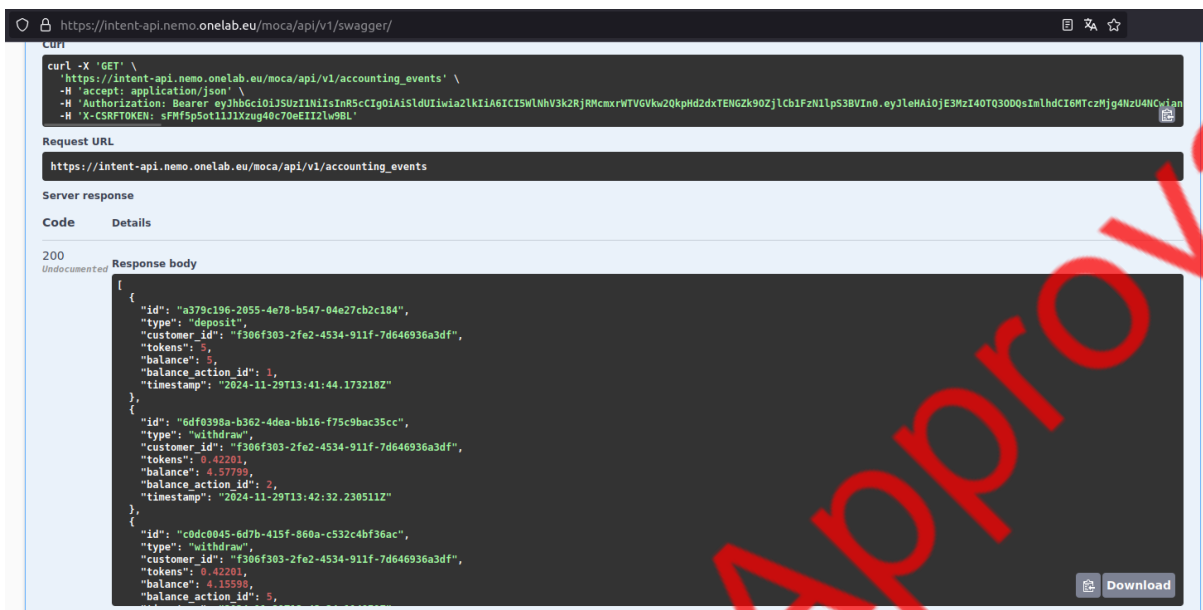


Figure 66: The accounting events of the workload user

Querying the endpoint available for the user's information, we can see that the balance has been updated accordingly for the previous transactions Figure 67.

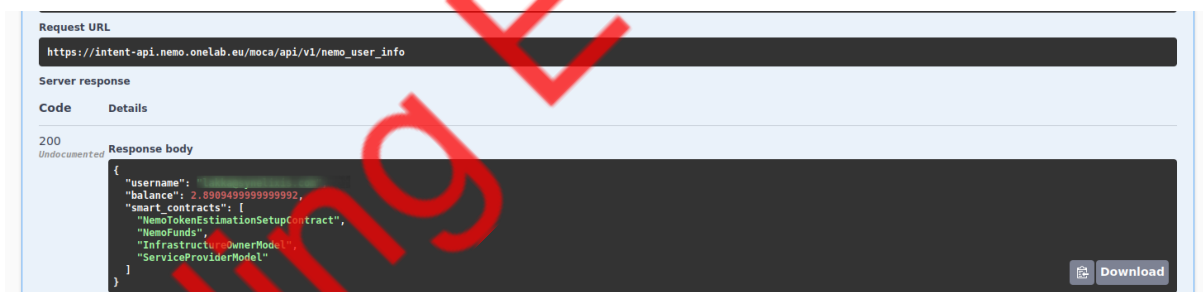


Figure 67: Workload user information

The details of the amount of resources used and the tokens charged, are also available through the `/moca/api/v1/workload_computations` endpoint (Figure 68).

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 69 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

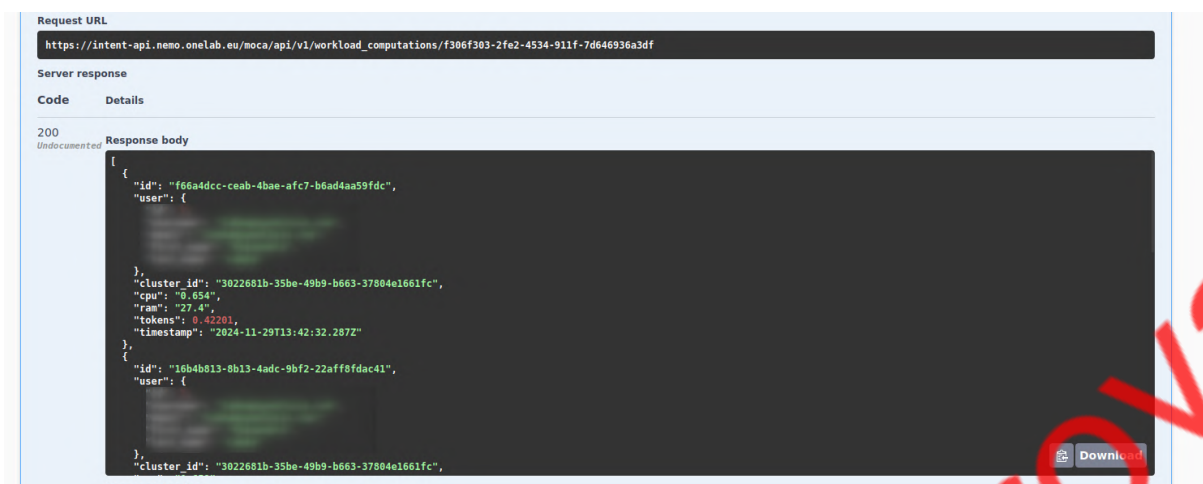


Figure 68: Details of the workload computation events

If we also query the accounting events as the cluster owner, we can see that events of depositing the payment to the owner are registered (Figure 69).

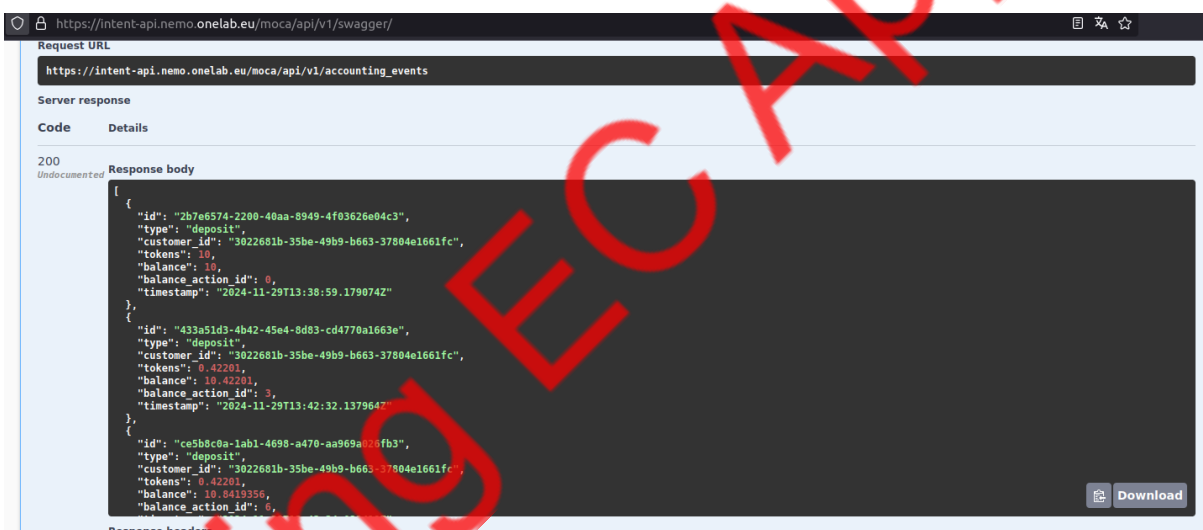


Figure 69: Cluster owner accounting events

Through the endpoint for the user's information, we can also check that the balance has been updated appropriately (Figure 70).



| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 70 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

Figure 70: Cluster owner user information

Now, we will examine the case of scaling up an already deployed workload from one to three pods (Figure 71).

| | | | |
|-----------------------------------|---|-----|-----------|
| lakka-echo-server-dd86f87f5-9tnr6 | ● | 2/2 | 0 Running |
| lakka-echo-server-dd86f87f5-67b2h | ● | 2/2 | 0 Running |
| lakka-echo-server-dd86f87f5-d8qp1 | ● | 2/2 | 0 Running |

Figure 71: Scaled up deployment

Since the deployment has been scaled up, its usage has increased, as well as the tokens it will be charged. Figure 72 shows the logs of MOCA when receiving the token computation for the workload resources.

```
[{"args": [{"serviceId": "f306f303-2fe2-4534-911f-7d646936a3df", "clusterId": "3022681b-35be-49b9-b663-37804e1661fc", "cpu": 240000000000, "ram": 631000000000, "tokens": 94001000}], [{"serviceId": "f306f303-2fe2-4534-911f-7d646936a3df", "clusterId": "3022681b-35be-49b9-b663-37804e1661fc", "cpu": 240000000000, "ram": 631000000000, "tokens": 94001000}], [{"event": "Task app.tasks.nemo_workload_compute_tokens[ee46bb1a-dae8-4a04-84c8-bfe5b54a03fc] succeeded in 7.264776561874896s: (1, '90716ff53387907852479bcbff8f0e4b1930f832191f44b678fcafa324766', '90716ff53387907852479bcbff8f0e4b1930f832191f44b678fcafa324766')"}]
```

Figure 72: MOCA logs for scaled workload usage

The last two entries in the `/moca/api/v1/workload_computations` endpoint show the computations before and after the workload scale, respectively. Both the memory and the CPU have increased and, as a result, the usage cost has a slight increase, as well (Figure 73).

| | | |
|-----------------|---------|--|
| Request URL | | https://intent-api.nemo.onelab.eu/moca/api/v1/workload_computations/f306f303-2fe2-4534-911f-7d646936a3df |
| Server response | | |
| Code | Details | |
| 200 | | |
| Response body | | <pre>{ "cluster_id": "3022681b-35be-49b9-b663-37804e1661fc", "cpu": "0.57499999", "ram": "27.9", "tokens": "0.41201", "timestamp": "2024-12-01T21:01:09.029Z" }, { "id": "a90f9384-de44-44e9-a06e-c76bed908290", "user": { "id": "a90f9384-de44-44e9-a06e-c76bed908290", "email": "a90f9384-de44-44e9-a06e-c76bed908290@nemo.onelab.eu", "password": "a90f9384-de44-44e9-a06e-c76bed908290" } }, { "cluster_id": "3022681b-35be-49b9-b663-37804e1661fc", "cpu": "2.4", "ram": "63.1", "tokens": "0.54001", "timestamp": "2024-12-01T21:25:30.402Z" }</pre> |

Figure 73: Comparison of workload usage results

3.3.5 Conclusion and roadmap

The conducted developments that materialized the first release of MOCA as part of the 1st integrated version of the NEMO meta-OS are summarized below.

- The development of smart contracts that facilitate the accounting process supporting several business models.
- The integration with the Service Management Layer components as part of the 1st integrated version of the NEMO meta-OS, namely the Intent-Based API and PPEF
- The development of MOCA to handle the different types of users and the calculations and updates made in a private blockchain network.

The associated results were presented in this section verified the functional competence of the component. Although majority of the required functionality and the corresponding integration with the meta-OS platform has already been achieved, the final release of the MOCA component in view of the final version of the NEMO meta-OS, concern the following activities:

| | | | | | |
|----------------|---|----------------|----|----------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 71 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

- Finalization of the accounting approach used in the smart contracts, by enriching the calculations.
- Provision of a management mechanism for adding, updating and deleting smart contracts via API.
- Enhancements on the provided functionalities to further enrich the information available for the NEMO users and their resources.
- Integration with the final version of the NEMO meta-OS.

3.4 Intent-based SDK/API

3.4.1 Overview

The NEMO Intent-based Application Programming Interface (IB-API) and Software Development Kit (SDK) act as the programmatic interface of NEMO to external users and/or services. It exposes NEMO functionality, as delivered by NEMO components, through RESTful calls to the API. It also supports configuration and for clusters and workloads in a declarative manner, realized as intent-based orchestration of network and computing loads. The Intent-based API stores NEMO workloads and their instances as API objects, which can be queried and managed via HTTP API calls. The Intent-based API can be accessed programmatically by NEMO users, as well as graphically via the LCM UI.

3.4.2 Architecture

The Intent-based API follows a modular architecture for delivering its main capabilities:

- Intent-based management that is consistent for both network and computing tasks
- Workload management and discovery
- NEMO functionalities exposure

The final version of the architecture features a simplified design and is depicted in Figure 74.

The IB-API services delivering intent-based orchestration include:

- NEMO Intent Manager
- NEMO Intent Validator
- NEMO Intent Collector

The IB-API services responsible for workload management and delivery include:

- NEMO Workload Manager
- NEMO Workload Validator
- NEMO Workload Registry, including the workload documents
- NEMO Intent Validator
- NEMO Intent Collector

In addition, the NEMO functionality exposure is directly offered through the Intent-based API Server, which provides RESTful endpoints for the supported operations.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 72 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

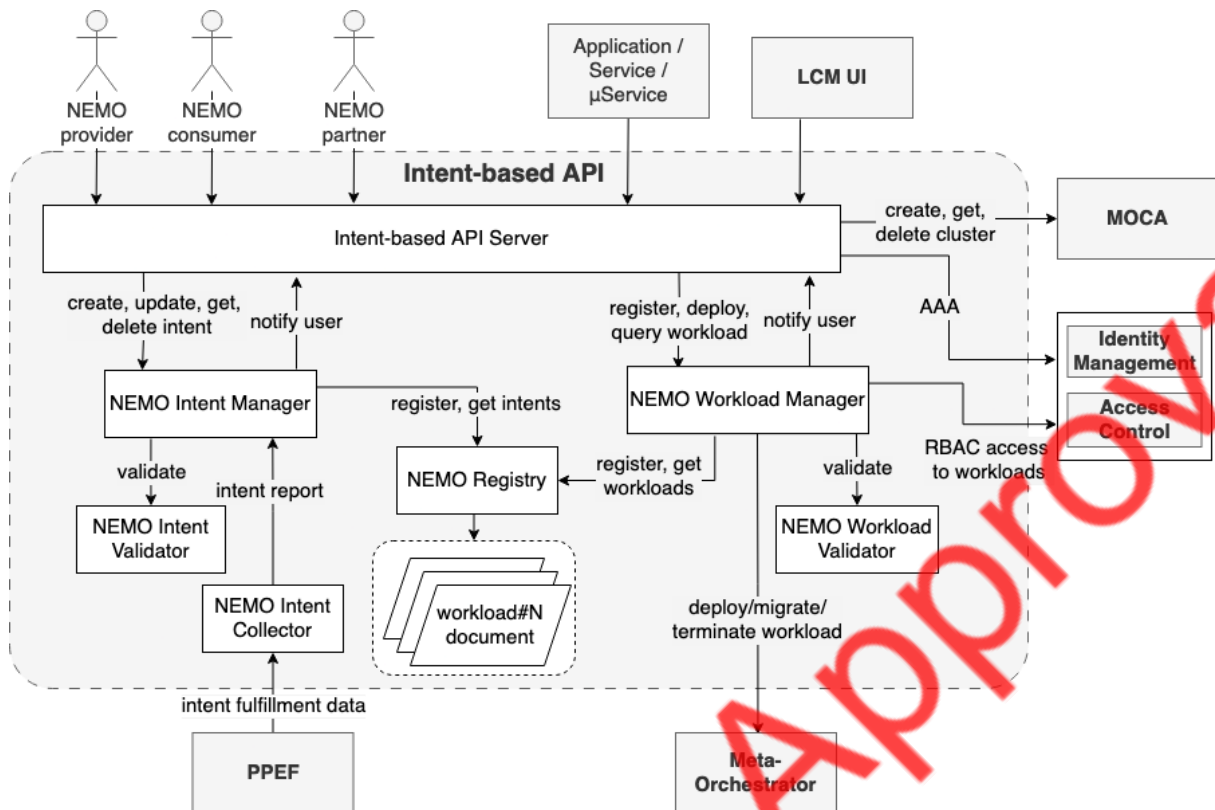


Figure 74: The final Intent-based API architecture

3.4.2.1 NEMO Intent Manager

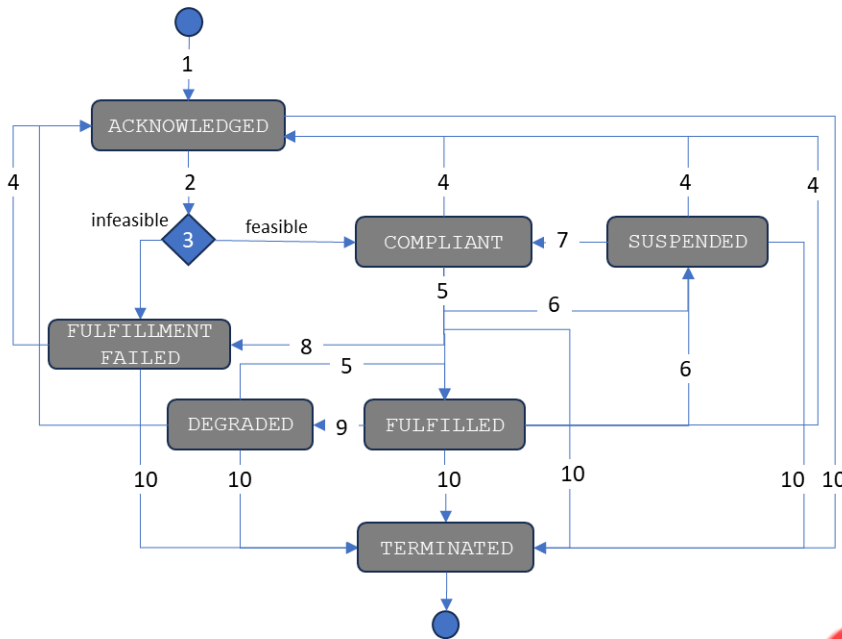
Intent-driven Management has been introduced originally for automating and adding intelligence into network systems. Network management in 5G systems is becoming too complex to deal with, considering increased human intervention or policy-based network management, present in 3G and 4G architectures. Intent-Driven Management (IDM) has, then, arisen to simplify network management and interaction of involved stakeholders, with operators having been alleviated from the burden of having technical awareness of network infrastructure, their policies, etc. [7]. 3GPP [8] defines an intent as an expression of the desired state of a system used to describe an intended network or service. In other words, an intent defines operator's expectations in a declarative yet concise manner in order for it to be understandable by both humans and machines.

Adopting this approach, in NEMO we extend intent-driven management to both network and computing workloads' management. Within the Intent-based API, the *NEMO Intent Manager* subcomponent undertakes the intents' -both for network and computing- lifecycle management within the NEMO ecosystem. This subcomponent provides backend logic for the management of the NEMO intents, following 3GPP TS 28.312 V18.3.0 (2024-03) [9]. Based on this technical specification, an intent has the following properties:

- It is typically understandable by humans and also needs to be interpreted by the machine without any ambiguity.
- It expresses in a declarative manner on the desired result ("what") and not the way it will be achieved ("how"). So, the intent includes metrics and target values, allowing alternative options to achieve them.
- The expectations expressed by an intent is agnostic to the underlying system implementation, technology and infrastructure.

Following TS 28.312, the *NEMO Intent Manager* subcomponent supports state management of the intents as per their lifecycle, as defined in Figure 75, borrowed from [9].

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 73 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |



State transitions

1. Intent creation
2. Feasibility check
3. Check feasibility outcomes
4. Intent updated
5. Fulfilment
6. Intent suspended or event occurs during fulfillment
7. Intent suspension lifted
8. Fulfillment failed
9. Event after fulfillment
10. Intent deleted

Figure 75: State transitions and reporting events for Intents delivered for fulfillment. [9], supported also in NEMO

So far, five intents have been defined and integrated into the Intent-based API, namely:

- Cloud continuum (network-oriented intent)
- Deliver computing workload (computing-oriented intent)
- Secure execution (computing-oriented intent)
- Federated learning (computing-oriented intent)
- Energy carbon efficiency (computing-oriented intent)

Indicatively, the “Deliver computing workload” intent is depicted in Figure 76.

```
- id: 1
  user_label: DeliverComputingWorkload
  intent_preemption_capability: 'FALSE'
  observation_period: 60
  intent_expectations:
  - id: 1
    expectation_id: '1'
    expectation_verb: ENSURE
    expectation_object:
      id: 1
      object_type: NEMO_WORKLOAD
      object_instance: b6a77b9a-4cb2-41e9-953b-0a0b569c8cdb
      context_selectivity: null
      object_contexts: []
    expectation_targets:
    - id: 1
      target_name: cpuUsage
      target_condition: IS_LESS_THAN
      target_value_range: '20'
      target_contexts: []
  - id: 2
```

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 74 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |


```

target_name: ramUsage
target_condition: IS_LESS_THAN
target_value_range: '200'
target_contexts: []
expectation_contexts: []
intent_report_reference:
  id: 1
  intent_fulfilment_report:
    id: 1
    intent_fulfilment_info:
      fulfilment_status: FULFILLED
      not_fulfilled_state: COMPLIANT
      not_fulfilled_reasons: []
    expectation_fulfilment_results:
      - expectation_fulfilment_info:
          fulfilment_status: FULFILLED
          not_fulfilled_state: COMPLIANT
          not_fulfilled_reasons: []
        expectation_id: 1
      target_fulfilment_results:
        - target: 1
          target_achieved_value: '0.307'
          target_fulfilment_info:
            fulfilment_status: FULFILLED
            not_fulfilled_state: COMPLIANT
            not_fulfilled_reasons: []
        - target: 2
          target_achieved_value: '1.2'
          target_fulfilment_info:
            fulfilment_status: FULFILLED
            not_fulfilled_state: COMPLIANT
            not_fulfilled_reasons: []
    intent_feasibility_check_report:
      id: 1
      feasibility_check_type: FEASIBLE
      infeasibility_reason: null
      last_updated_time: '2024-11-05T15:10:07.949361Z'
  intent_contexts: []

```

Figure 76: The DeliverComputingWorkload intent definition in NEMO Intent-based API

3.4.2.2 NEMO Intent Validator

The *NEMO Intent Validator* is performing a set of validation checks on intents that are newly defined or desired to be updated. It interacts with the Intent Manager enhancing its provided functionality offering validation checks which include:

- **Schema validation:** This process ensures that data conforms to a defined structure or format, typically described in a schema. A schema acts as a blueprint, specifying the expected data types, required fields, and constraints for a dataset. The schema used for intents follows 3GPP TS 28.312 V18.3.0 specification, so the component ensures that the required fields are provided, within the acceptable value range, as well as in acceptable combinations (e.g. compatible target names, intent types and expectation verbs). Schema validation is crucial for maintaining data

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 75 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

integrity, preventing malformed or unexpected data from causing errors or security vulnerabilities in applications. By enforcing these rules at an early stage, schema validation helps streamline data processing and reduces the likelihood of runtime issues.

- **Intent definition updates:** There are fields in the defined intents that are allowed to be updated, such as the period during which the intent will be active or the expectation target value range. However, such updates are allowed before the intent is activated, i.e. for intent states “FULFILMENTFAILED”, “TERMINATED” and “ACKNOWLEDGED”.
- **Intent operations:** Operations on intents can be performed, depending on both the desired intent action and the “NotFulfilledState” value. Intent operations include “RESUME”, “SUSPEND”, “TERMINATE”, etc. The “Not_Fulfilled_State” field represents the state of the intent, while it is not fulfilled. The attempted action must comply with the state transition schema in Figure 75.
- **Intent expectation updates:** Are allowed only for expectations defined for each intent. Depending on the selected “userLabel” (intent type), the corresponding allowed expectation targets are checked for validity. For example, “DeliverComputingWorkload” supports targets of type ram usage and/or cpu usage.
- **Additional Feasibility checks:** After intent validation, an asynchronous validation step is performed in order to test the feasibility of the intent in question. Reasons for failure in this step include “serviceStartTimes” & “serviceEndTimes” out of order and/or collisions with an already existing Intent for the given workload.

3.4.2.3 NEMO Intent Collector

The *NEMO Intent Collector* is a significant modality of the Intent-based API which facilitates the collection of the intent associated measurements governed by the PPEF component. More specifically, the NEMO Intent Collector receives as an input the intent related measurements that correspond to the expectation targets that have been set by the NEMO user, as they are reported from the PPEF. The communication with the PPEF is achieved via a RabbitMQ listener service that is provided by the NEMO Intent Collector. Then the consumed information is processed and structured as intent fulfillment data which correspond to intents expectation targets’ achieved values. Finally, the intent report is consumed by the NEMO Intent Manager which updates the corresponding information in the NEMO Registry.

3.4.2.4 NEMO Workload Manager

The *NEMO Workload Manager* modality is the heart of the Intent-based API component. Its functionality concerns the management and the governance of the NEMO workloads as it is dictated by the NEMO user. Specifically, the *NEMO Workload Manager* manages the processes for registration/deregistration, deployment and migration of workloads, including NEMO annotations, workflow execution, provisioning, logging and notification of external entities. As it is illustrated in Figure 74, the *NEMO Workload Manager* handles the workload requests that are dispatched by the Intent-based API. The requests can be triggered either from the LCM UI or directly from the Intent-based API Server. The workload Manager validates the workload registration and/or deployment configuration files that are issued through the *NEMO Workload Validator* (its respective activities are detailed in section 3.4.2.5). At the same time, the NEMO Workload Manager facilitates the update of the workload state in the NEMO registry and responds to workload queries. Once the workload request is pre-processed the workload’s status changes to “onboarding” and subsequently it’s communicated through the RabbitMQ message queue to the Meta-Orchestrator (MO). In case the workload validation process fails, the workload status is changing to rejected and the request is terminated. The MO executes the requested action and dispatches back the result of the requested activity (acknowledgement message). The NEMO Workload Manager updates the NEMO Registry accordingly.

Finally, the *NEMO workload Manager* supports automated provisioning, triggering authorization requests (RBAC access) for the workload to *Access Control component*.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 76 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

3.4.2.5 NEMO Workload Validator

The *NEMO Workload Validator's* offered functionality, in the context of the *NEMO Workload Manager*, is described above. This section sheds light into the specific validation tests that are performed by the module. The validation tests are listed below.

1. **Helm chart structural validation.** The uploaded *.tgz helm* chart that corresponds to the *workload upload* request triggered by the NEMO user through LCM UI or directly through the Intent-based API (invokes the */workload/upload/* endpoint; described in Annex B) is validated by checking the existence of *Chart.yaml*, *values.yaml*, templates folder (with appropriate *.tpl .yaml* template files) alongside with the contents of *Chart.yaml* (matching version and naming scheme during the invocation of the */workload/* POST endpoint). In addition, all the *.yaml* files pass a syntax check.
2. **Helm chart template validation.** The uploaded *.tgz helm* chart is extracted and undergoes a rendering phase of the templates with the default provided *values.yaml*. (by using the standard helm template sub-command)
3. **Docker image access validation.** For every generated manifest (*Deployment*, *Statefulset*, *DaemonSet*) the container image of every mentioned image is tested for access. Provided *imagePullSecrets* are taken into consideration with additional secrets of type *kubernetes.io/dockerconfigjson* or against well-known docker repositories (e.g. NEMO repository)
4. **Ingress support validation.** If the uploaded NEMO workload supports ingress via the Access Control component, the validator checks for the existence of a Kubernetes Service with the annotations described in the figure below.

| Annotation | Type | Default value | Description |
|------------------------------|----------|--|---|
| nemo.eu/ingress-expose | Required | "true" | Marks the service as exposable via NEMO Ingress |
| nemo.eu/ingress-service-port | Optional | If not set it defaults to the port of the Service marked above | The associated Service port |
| nemo.eu/ingress-path | Optional | "/" | Ingress path to expose |
| nemo.eu/ingress-path-type | Optional | "ImplementationSpecific" | Ingress path type |

Figure 77: Kubernetes Service Annotations²⁰

3.4.3 Initial results

The Intent-based API associated results stemming from the integration tests that are conducted in view of the 1st integrated version of the NEMO framework are documented in section 4.

3.4.4 Conclusion and Roadmap

The implementation of the core functionality that is offered by the Intent-based API in the context of the 1st integrated NEMO framework is considered completed. The component was integrated with the rest of the NEMO Service Management Layer components namely, the MOCA, PPEF, LCM and the Access Control. Moreover, the Intent-based API demonstrated its integration with the NEMO Kernel and the MO supporting the workload deployment and migration process.

With respect to the next steps, the Intent-based API as part of the 1st integrated version of the NEMO framework, will be deployed in NEMO pilots' infrastructures and will be further validated through the

²⁰ https://gitlab.eclipse.org/eclipse-research-labs/nemo-project/nemo-service-management/intent-based-sdk_api/intent-api#exposing-a-workload-document-instance-via-nemo

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 77 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

NEMO pilots' specific use cases and also via the integration with the NEMO OC1 offered meta-OS extensions and OC2 provided NEMO services.

In view of the final version of the NEMO meta-OS framework, the Intent-based API aims to further enhance its provided functionality where necessary (according also the feedback that will be gained through the abovementioned activities) improving the quality of its provided services.

Pending EC Approval

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 78 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

4 NEMO scenario-driven verification & results

The purpose of this section is to provide insights on the integration tests conducted in the framework of the integration activities that materialized the 1st integrated version of the NEMO platform. NEMO integration verification approach that is defined and adopted, as described in section 2.3, establishes the foundation upon which the integration activities are conducted and documented. Emphasizing on cross-cutting functions of the NEMO meta-OS, NEMO defined four (4) system-level integration scenarios that aim to illustrate the technical readiness of the NEMO developed components. These scenarios, as detailed below, define the context of the integration activities conducted for the realization of the 1st integrated version of the NEMO meta-OS. The four (4) scenarios are the following:

- *NEMO cluster registration*
- *NEMO workload registration and provisioning*
- *NEMO workload scheduling and orchestration*
- *NEMO workload lifecycle management*

For each of the abovementioned test cases, the respective scenario that is followed is described. Based on that, the resulting process diagram highlights the steps that materialize the integration objective in each case. The scenario-driven process diagrams reflect the latest iteration/evolution of the process that was described in D1.3 [10]. Finally, the results that are collected for each of the steps are detailed and subsequently the summary checklist, presents the outcome of the conducted tests.

4.1 NEMO Cluster registration

This section describes the NEMO Cluster registration integration scenario. The Cluster registration workflow aims to provide technical details of the process that is followed allowing the NEMO partner (infrastructure owner/provider) to access the NEMO meta-OS service management layer through either via Intent-API or LCM UI and register a new resource (infrastructure) to be utilized and governed by the NEMO meta-OS. The associated sequence diagram is presented in Figure 78.

| | | | | | | | |
|----------------|--|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 79 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

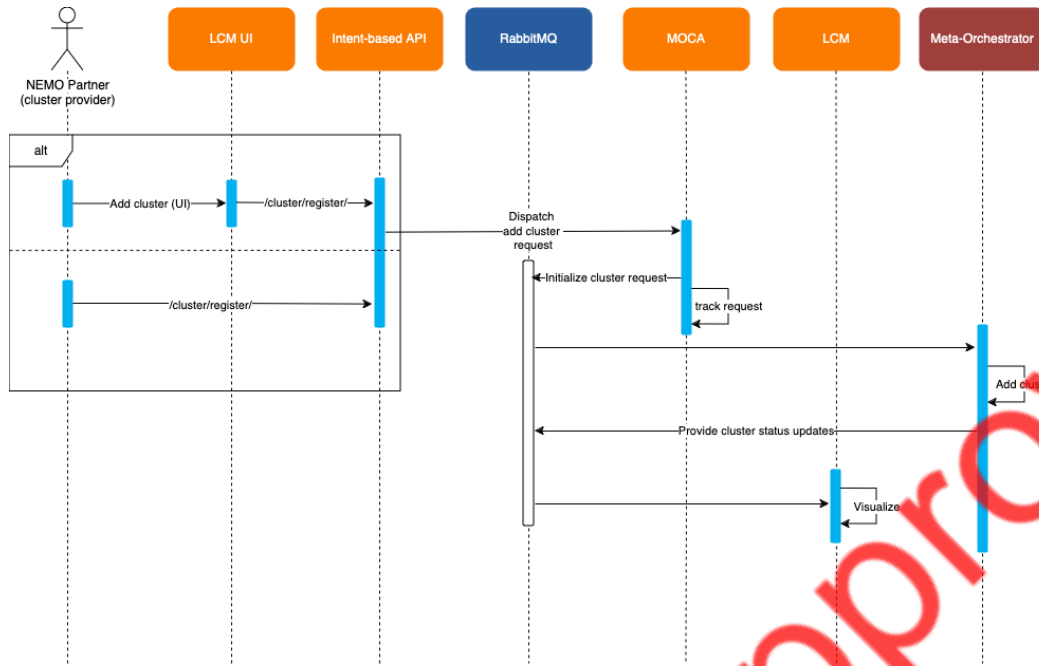


Figure 78: Process diagram for cluster registration

4.1.1 Verification scenario

| Test 1: NEMO Cluster registration | |
|-----------------------------------|---|
| Objective | To verify the cluster registration process in NEMO that facilitates the resource provisioning triggered by the NEMO partner (resource owner) |
| Components | <ul style="list-style-type: none"> • LCM • Intent-based API • MO • MOCA • RabbitMQ |
| Features to be tested | The feature that this scenario aims to test are the cluster registration process which is initiated by the NEMO partner (cluster provider) through the LCM UI & Intent-based API. Then, the newly registered cluster is added into the NEMO meta-OS ecosystem by the MO. The results (status) of this process are then visualized to the user. |
| Test setup | All the participating components were deployed in the NEMO meta-OS cloud/edge infrastructure at OneLab (dev cluster 1). |
| Steps | <ol style="list-style-type: none"> 1. Cluster registration through the LCM UI 2. Cluster registration through the Intent-based API (realized in MOCA) 3. Cluster registration message communication to MO 4. Cluster addition process by MO 5. Cluster status provisioning to RabbitMQ 6. Cluster status update visualization in LCM UI |

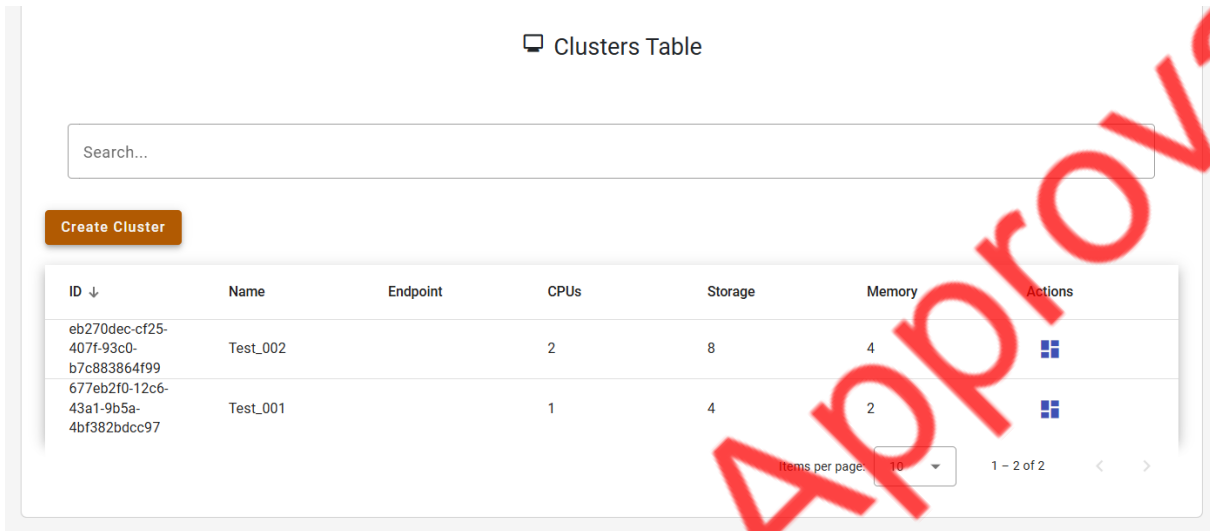
| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 80 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

4.1.2 Results

This section documents the process that is described in the scenario above step by step.

4.1.2.1 Cluster registration through the LCM UI

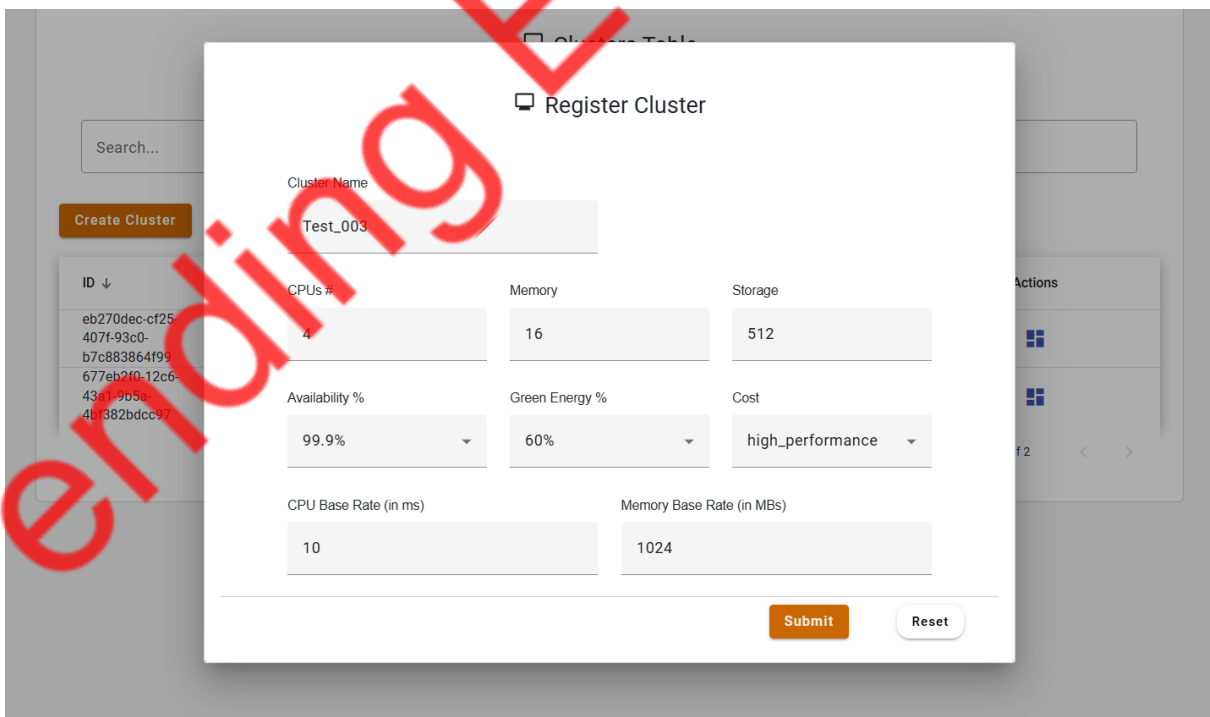
The following figure (Figure 79) depicts the cluster table summary that the NEMO meta-OS governs. Through this interface the user is able to overview some high-level information that describes each of the provided infrastructures.



| ID ↓ | Name | Endpoint | CPUs | Storage | Memory | Actions |
|--------------------------------------|----------|----------|------|---------|--------|---------|
| eb270dec-cf25-407f-93c0-b7c883864f99 | Test_002 | | 2 | 8 | 4 | |
| 677eb2f0-12c6-43a1-9b5a-4bf382bdcc97 | Test_001 | | 1 | 4 | 2 | |

Figure 79: Cluster summary view on LCM GUI

Figure 80, illustrates the form that corresponds to the “*create cluster*” button of the dashboard. Through this form the NEMO user is able to add the cluster description and subsequently initiate the cluster registration process. This form, once filled in by the user, triggers the corresponding endpoint that is provided through the MOCA API (section 3.3.3.1).



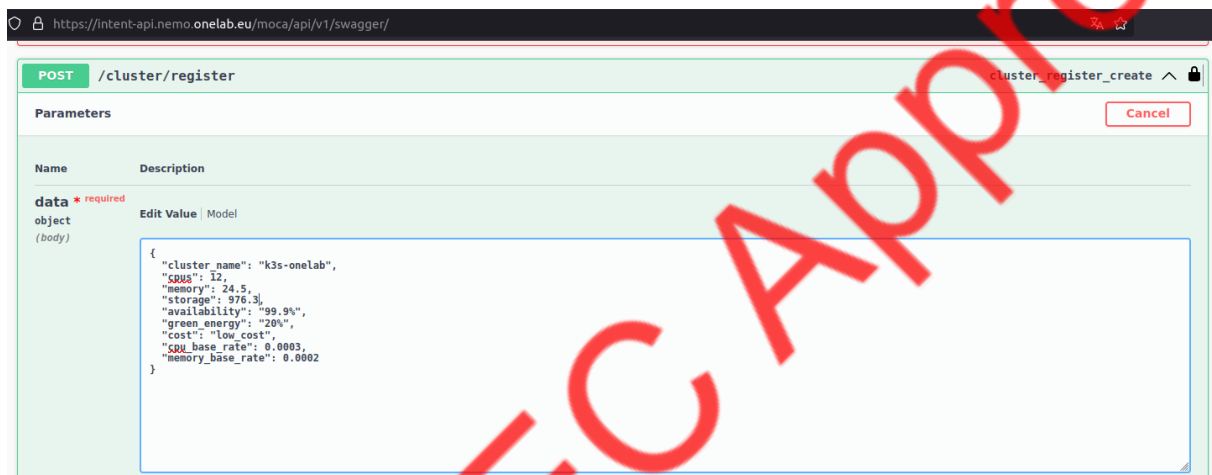
| Register Cluster | | |
|--|-----------------------------------|--------------------------|
| Cluster Name Test_003 | | |
| CPUs # 4 | Memory 16 | Storage 512 |
| Availability % 99.9% | Green Energy % 60% | Cost high_performance |
| CPU Base Rate (in ms) 10 | Memory Base Rate (in MBs) 1024 | |
| <input type="button" value="Submit"/> <input type="button" value="Reset"/> | | |

Figure 80: Cluster registration page on LCM GUI

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 81 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

4.1.2.2 MOCA operations for cluster registration

When a new cluster comes for registration, the `/moca/api/v1/cluster/register` endpoint can be used through the LCM UI. Alternatively, as indicated in the process diagram above, the user can trigger the associated endpoint directly from the Intent-based API (Figure 81). For both of these cases, the cluster registration request is executed via the MOCA API provided endpoint that is mentioned above. The cluster provider needs to provide the specifications of the cluster that is to be added, like its name, the resources it provides (CPUs, memory, disk), the availability percentage, the green energy percentage that reflects the amount of energy that used to power the cluster and comes from renewable energy sources, its cost category and the associated costs for its available resources. Figure 81 shows the registration payload of a cluster named “*k3s-onelab*”. Figure 82 shows the response of the successful registration to MOCA. The response is the cluster’s id.



POST /cluster/register

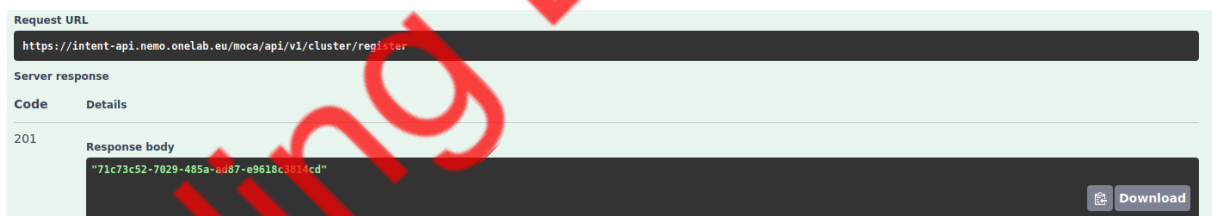
Parameters

Name Description

data * required
object
(body)

```
{
  "cluster_name": "k3s-onelab",
  "cpus": 12,
  "memory": 24.5,
  "storage": 976.3,
  "availability": "99.9%",
  "green_energy": "20%",
  "cost": "low_cost",
  "cpu_base_rate": 0.0003,
  "memory_base_rate": 0.0002
}
```

Figure 81: MOCA Cluster registration demonstration



Request URL

https://intent-api.nemo.onelab.eu/moca/api/v1/cluster/register

Server response

Code Details

201

Response body

```
"71c73c52-7029-405a-a0d7-e9618c3854cd"
```

Download

Figure 82: MOCA Cluster registration response

4.1.2.3 MO cluster registration operation

When MOCA receives a new request performs the necessary validation checks and sends the request to the NEMO Meta Orchestrator, through the NEMO RabbitMQ, in order to join the cluster with the NEMO platform. Figure 83 shows this step of the cluster registration workflow.

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 82 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

Autoscroll:On  FullScreen:Off  Timestamps:Off  Wrap:Off
Operations to perform:
  Apply all migrations: app
Running migrations:
  Applying app.0052_clusterresources_managed_api... OK
Operations to perform:
  Synchronize unmigrated apps: corsheaders, drf_yasg, messages, nemo, postgres, rest_framework, runserver_nostatic, staticfiles, thanos
  Apply all migrations: admin, app, auth, authtoken, contenttypes, sessions, sites
Synchronizing apps without migrations:
  Creating tables...
  Running deferred SQL...
Running migrations:
  No migrations to apply.
[2024-12-04 08:46:01 +0000] [11] [INFO] Starting gunicorn 20.1.0
[2024-12-04 08:46:01 +0000] [11] [INFO] Listening at: http://0.0.0.0:8000 (11)
[2024-12-04 08:46:01 +0000] [11] [INFO] Using worker: gthread
[2024-12-04 08:46:01 +0000] [13] [INFO] Booting worker with pid: 13
[04/Dec/2024 08:52:00] - [nemo.rabbitmq:49] - [INFO] [x] Sent to Meta Orchestrator!

```

Figure 83: MOCA sends cluster details to Meta Orchestrator

Figure 84 shows that the Meta Orchestrator has successfully received the cluster's details:

```

2024/12/04 09:08:39 Request Payload: {"availability":"99.9%","cluster_name":"k3s-onelab","managed_api":"https://api.s2.nemo.onelab.eu:6443"}

```

Figure 84: Meta Orchestrator receives the cluster registration request

After the request is processed successfully by the Meta Orchestrator, it informs MOCA again through the RabbitMQ. Figure 85 shows that MOCA received the Meta Orchestrator validation message for the successful registration. We can, additionally, see that the cluster was also registered in the blockchain.

```

Autoscroll:On  FullScreen:Off  Timestamps:Off  Wrap:Off
[INFO] [*] Waiting for messages. To exit press CTRL+C
[INFO] [x] Received ('managed_api': 'https://api.s2.nemo.onelab.eu:6443', 'cluster_name': 'k3s-onelab', 'id': '71c73c52-7029-485a-ad87-e9618c3814cd', 'timestamp': '2024-12-04T10:08:39+01:00', 'status': 'ok')
[INFO] The cluster was registered with transaction hash 0x944dd9ac27bf7b10f2ff93b5df7fc9d307ca57365b267fc9d307ca57365b26

```

Figure 85: MOCA receives the Meta Orchestrator response

4.1.2.4 MOCA cluster registration to the blockchain

To register the cluster in the blockchain, MOCA communicates with the DApps deployed, specifically the one responsible for handling the cluster registration (see section 3.3.4.3). Figure 86 shows MOCA calling the contract and successfully registering the cluster, receiving back the appropriate response (the cluster registration initial tokens).

```

[INFO] [*] Revert reason detected
[INFO] Task app.tasks.nemo_workload_compute_tokens[e5a38483-952f-4c23-93b9-0a276f7ab32f] succeeded in 4.74949499219656s: (0, 'd83e2f96b6aedd4e79da40cd7f549f5c29a6b2ade
Task app.tasks.nemo_register_cluster[f578a2e-f40b-46ac-b8a3-7381bca72639] received
TaskPool: Apply <function fast_trace_task at 0x7f272f007408> (args:('app.tasks.nemo_register_cluster', 'f578a2e-f40b-46ac-b8a3-7381bca72639', {'lang': 'py', 'task'
s:111}) - [INFO] ('args': [{'customer_id': '71c73c52-7029-485a-ad87-e9618c3814cd', 'customer_type': 'infrastructure', 'balance': 1000000000, 'nemoBalanceActionId': 42}],
[INFO] ('args': [{'customer_id': '71c73c52-7029-485a-ad87-e9618c3814cd', 'customer_type': 'infrastructure', 'balance': 1000000000, 'nemoBalanceActionId': 42}], 'event':
[INFO] Task app.tasks.nemo_register_cluster[f578a2e-f40b-46ac-b8a3-7381bca72639] succeeded in 3.3040527780540287s: (1, '944dd9ac27bf7b10f2ff93b5df7fc9d307ca57365b26
Task app.tasks.nemo_workload_compute_tokens[2c54069d-ed3b-47b0-97e0-6aba4d88a438] received

```

Figure 86: Register cluster to blockchain

| | | | | | |
|----------------|--|----------------|----|----------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 83 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

MOCA, then, appropriately, updates the details of the cluster and the user. Figure 87 shows that the status of the cluster has been updated, as well as the tokens provided to the cluster.

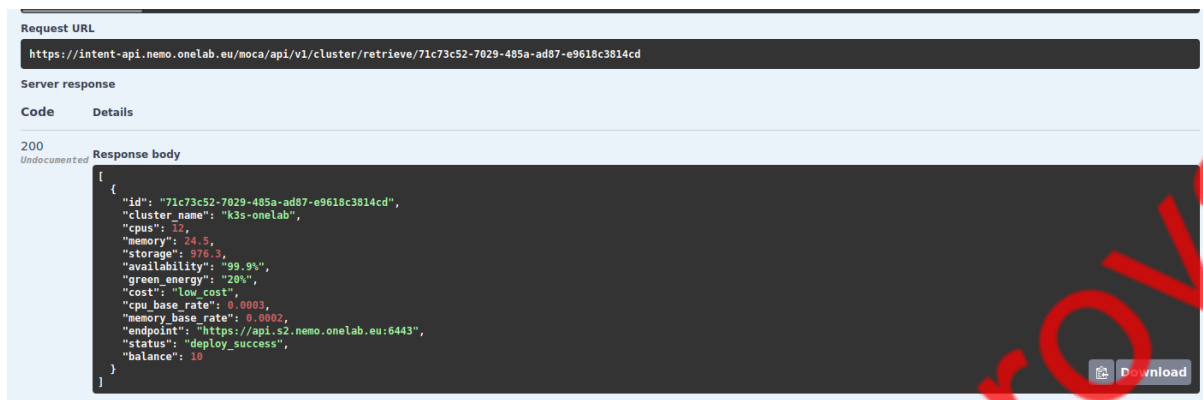


Figure 87: Updated cluster details

Finally, Figure 88 shows that the cluster provider can view through the `/moca/api/v1/accounting_events` endpoint the event of the registration, and more specifically the deposit of the ten initialization tokens. It should be noted that the `balance` field is the total amount of tokens owned by a user. Here, the user owns a number of resources. Every time a user registers a new resource, the registration reward tokens will be added to his/her total balance.

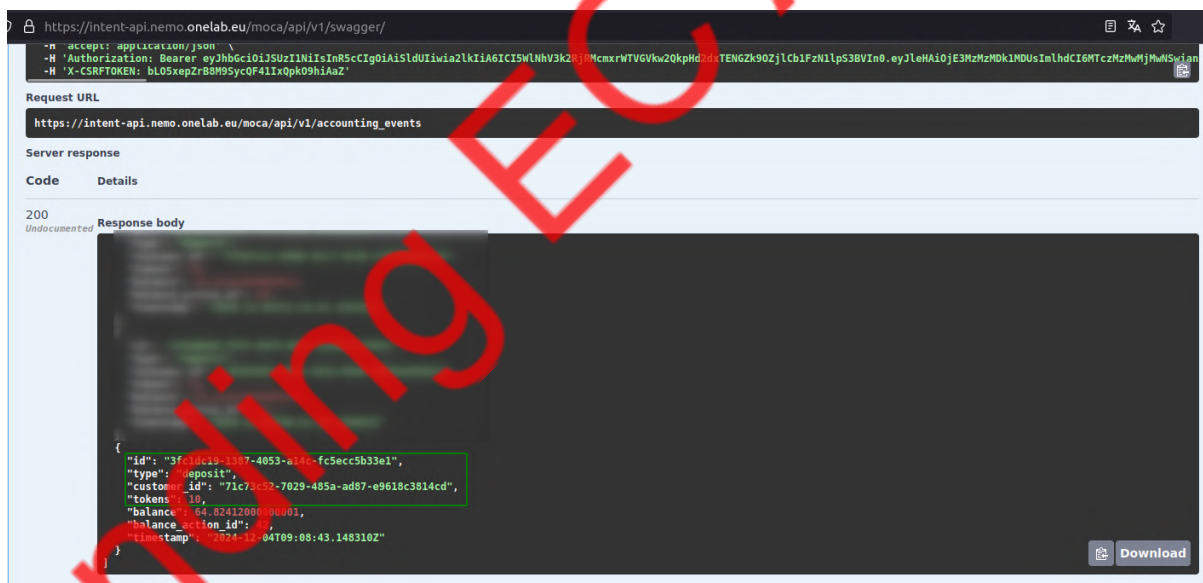


Figure 88: MOCA accounting event for cluster registration

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 84 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

4.1.3 Verification summary checklist

| Checklist for Cluster registration scenario | | | | |
|---|---|-----|----|--|
| | | Yes | No | Comments |
| 1 | Is the cluster registered by the user through the LCM UI | ✓ | | Success |
| 2 | Is the cluster registered through the Intent-based API (realized in MOCA) | ✓ | | Success |
| 3 | Is the registration process communicated to the MOCA | ✓ | | Success |
| 4 | Is the MOCA validation check successfully executed? | ✓ | | Success |
| 5 | Is MOCA communicating successfully the request to MO? | ✓ | | Success |
| 6 | Is the MO provided functionality successfully executed? | ✓ | | Success |
| 7 | Is the cluster successfully added to the NEMO meta-OS? | ✓ | | Success |
| 8 | Is the updated status communicated successfully to MOCA through RabbitMQ? | ✓ | | Success |
| 9 | Is the new cluster registered to the MOCA operated BC | ✓ | | Success |
| 10 | Are the new cluster details available through the MOCA API? | ✓ | | Success |
| 11 | Are the information visible to the LCM UI | | ✓ | The provisioning of the accounting events to the LCM UI. Feature to be available in the final version. |
| 12 | Is the updated status visible to the NEMO user? | ✓ | | Success |

Table 9: Checklist for cluster registration scenario

4.2 NEMO workload registration, deployment & provisioning

This scenario concerns two specific operations. The first one is the NEMO workload registration process and the second one is the NEMO workload deployment process. For both of these activities, the corresponding sequence diagrams dictating the integration scenario that is followed are presented in Figure 89 and Figure 90, respectively. The former describes the steps necessary for the workload registration process while the latter the workload deployment and provisioning steps. The workload provisioning step that is facilitated by the Access Control is illustrated in Figure 91.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 85 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

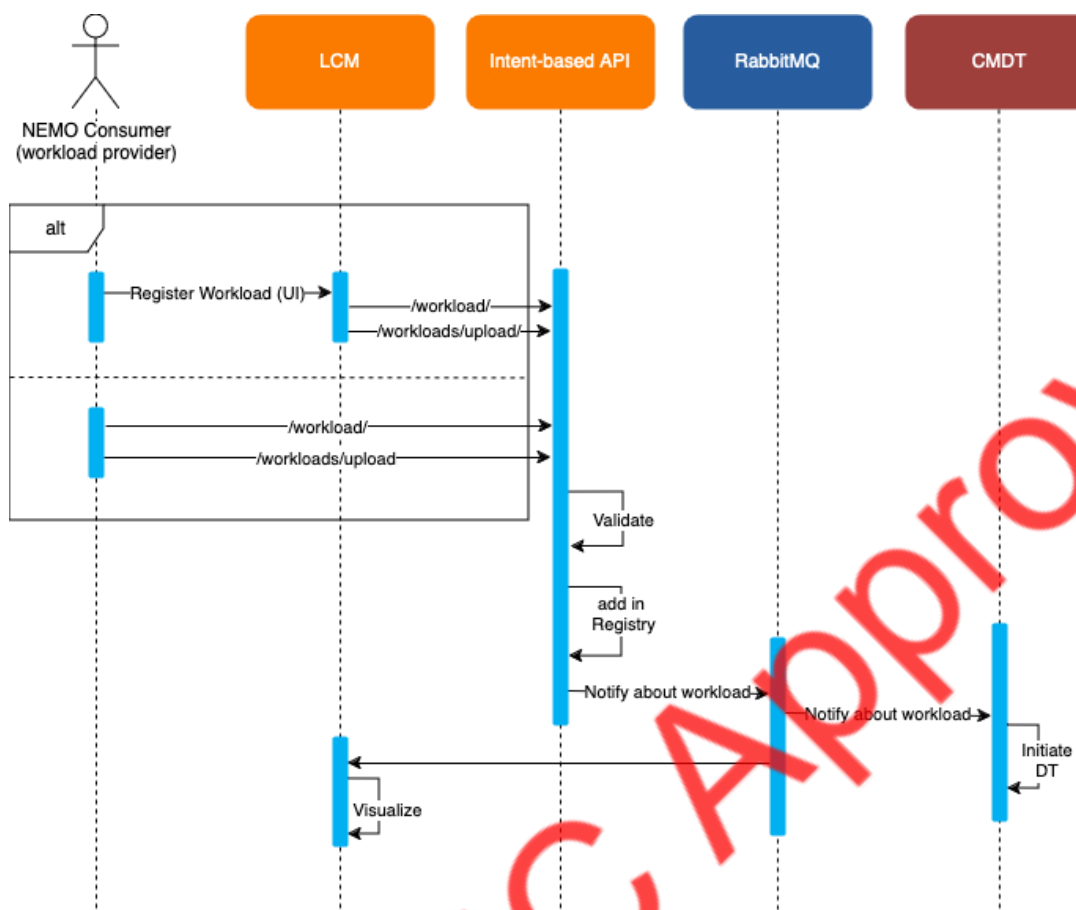


Figure 89: Process diagram for workload registration

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 86 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

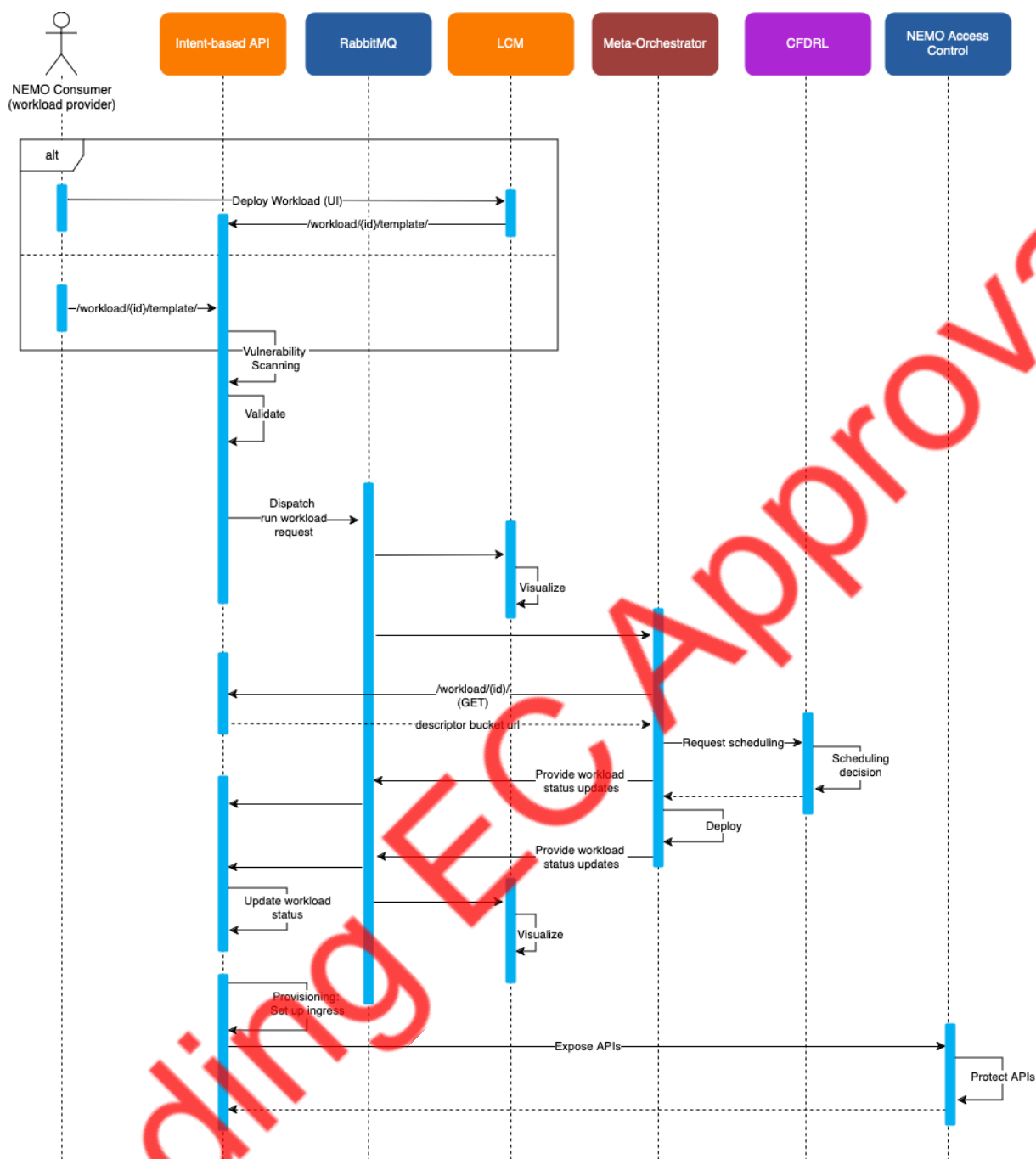


Figure 90: Process diagram for workload deployment (provisioning)

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 87 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

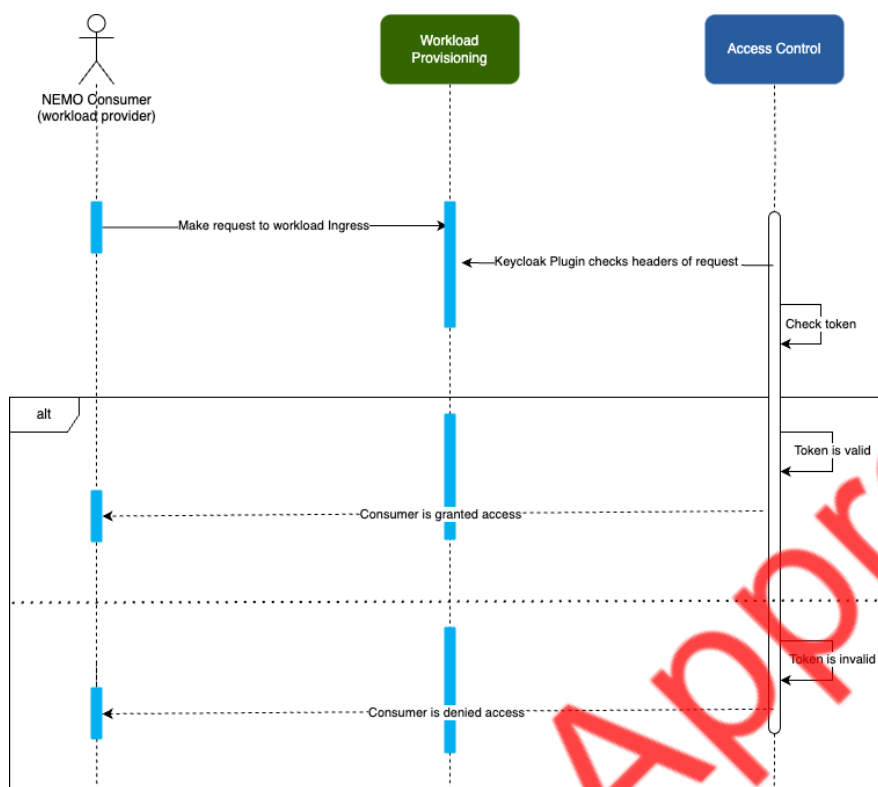


Figure 91: Access control sequence diagram - detailed view

4.2.1 Verification scenario

Test 2: NEMO workload registration and provisioning

| | |
|------------|---|
| Objective | Verify the NEMO workload registration, deployment and provisioning process |
| Components | <ul style="list-style-type: none"> NEMO Workload Registration <ul style="list-style-type: none"> LCM Intent-based API RabbitMQ NEMO Workload deployment <ul style="list-style-type: none"> LCM Intent-based API CMDT RabbitMQ Meta-Orchestrator CFDRL NEMO Access Control NEMO Workload provisioning <ul style="list-style-type: none"> Intent-based API |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 88 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | |
|-----------------------|--|
| | <ul style="list-style-type: none"> ○ Access Control |
| Features to be tested | <p>To verify the workload registration, deployment and provisioning process in NEMO, the following features will be tested:</p> <ul style="list-style-type: none"> • Workload Registration • Workload Deployment • Workload Provisioning <p>The feature that this scenario aims to test are the workload registration process which is initiated by the NEMO consumer (workload provider) through the LCM UI & Intent-based API. Then, the newly registered workload is requested to be deployed into the NEMO meta-OS ecosystem by the MO. Finally the workload provisioning process is triggered which is facilitated by the Intent-based API and the Access Control components. The results including the workload status are visualized to the user.</p> |
| Test setup | The associated components are deployed in OneLab facilities (dev cluster 1 & staging cluster 2) |
| Steps | <p>The steps identified in the associated sequence diagrams are listed below:</p> <ol style="list-style-type: none"> 1. NEMO workload registration <ol style="list-style-type: none"> a. Workload registration by the NEMO user through the LCM UI b. Execution of workload validation process in Intent-based API c. Notification of the LCM UI about the status of the workload registration 2. NEMO workload deployment <ol style="list-style-type: none"> a. Workload deployment by the NEMO user through the LCM GUI b. Execution of workload validation in Intent-based API c. Communication of the deployment request to the LCM UI d. Communication of the deployment request to the MO e. Deployment operation process triggered by MO f. Request scheduling by the CFDRDL component g. Deployment operation process executed by MO h. Communication and update of the deployment operation status to the Intent-API i. Visualization of the updated status to the LCM UI 3. NEMO workload provisioning <ol style="list-style-type: none"> a. NEMO workload provisioning is triggered by the Intent-based API b. NEMO Access Control workload setup c. NEMO Access Control Keycloak plugin functionality d. Performance resilience of Kong Plugin |

4.2.2 Results

This section documents the process that is described in the scenario above step by step.

4.2.2.1 Workload registration through LCM GUI

The workload registration process is facilitated by the LCM component and its UI as it's described in section 3.2.4. The LCM component utilizes the Intent-based API provided functionality in order to realize the NEMO user triggered operations for the workload registration process. Figure 92, presents the form that corresponds to the workload registration and Figure 93 presents the list of the registered workloads.

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 89 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

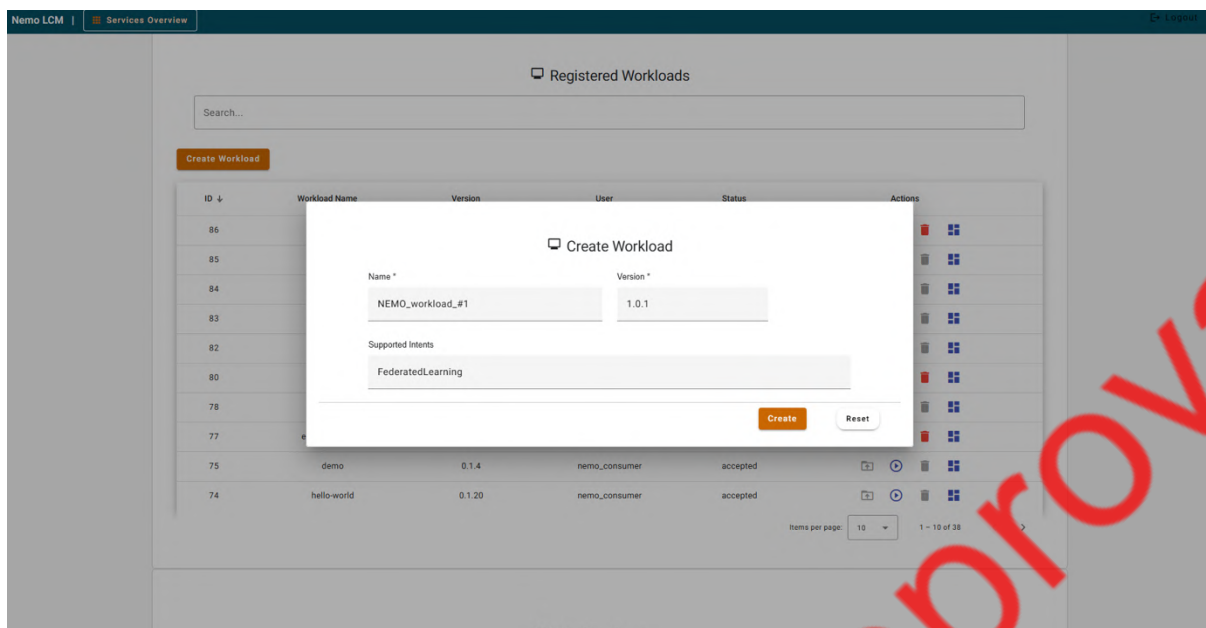


Figure 92: NEMO workload registration through LCM UI

Registered Workloads

Search...

Create Workload

| ID ↓ | Workload Name | Version | User | Status | Actions |
|------|------------------|---------|------------|----------|---|
| 89 | NEMO_workload_#1 | 0.1.0 | [REDACTED] | pending | + ▶ 🗑 ⌵ |
| 88 | hello-world | 0.1.22 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |
| 87 | NEMO_workload_#1 | 1.0.1 | [REDACTED] | pending | + ▶ 🗑 ⌵ |
| 86 | nginx | 0.1.0 | [REDACTED] | pending | + ▶ 🗑 ⌵ |
| 85 | hello-world | 0.1.21 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |
| 84 | echo-server | 0.5.7 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |
| 83 | echo-server | 0.5.6 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |
| 82 | echo-server | 0.5.10 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |
| 80 | test | 0.1.15 | [REDACTED] | pending | + ▶ 🗑 ⌵ |
| 79 | echo-server | 0.5.1 | [REDACTED] | accepted | + ▶ 🗑 ⌵ |

Items per page: 10 1 - 10 of 41

Figure 93: NEMO registered workloads

Intent Management provides the privileged user with the interfaces to create and manage intents. Figure 94 shows the create workload instance form.

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 90 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

Create Workload Instance

Release Name *

echo-server-integration1

Namespace *

default

Intent Type *

DeliverComputingWorkload

Remove Intent

Start Date-time

04/12/2024, 12:00:00

End Date-time

31/12/2024, 00:00:00

Target Name

cpuUsage

Target Condition

IS_LESS_THAN

Target Value

20

ramUsage

IS_LESS_THAN

50

Add Target

Create Intent Template

Render

Reset

Figure 94: NEMO workload instance creation (workload deployment process) through LCM UI

The newly created NEMO workload instance is visible in the *workload instances* table in LCM UI (Figure 95).

Workload Instances

| ID | instance_id | release_name | workload_document_id | status | Actions |
|----|--------------------------------------|----------------------------|----------------------|----------|---------|
| 87 | 707b5c50-f46c-471c-b2af-5f64324b159d | echo-server-integration1 | 82 | rendered | |
| 86 | 304fa738-d4b8-42b6-a88f-9dfab81ead5f | hello-test-002 | 85 | deployed | |
| 85 | 30aa8464-7c3a-499c-ba47-7033e6d086dd | hello-test-002 | 85 | deployed | |
| 84 | 44834c32-d064-4c06-ad51-bcd76c491c34 | hello-test-001 | 85 | deployed | |
| 83 | 82b1f6b3-4a55-4d86-b47c-37f88f950523 | test-002 | 85 | deployed | |
| 82 | 406a78b-3c6c-4438-b0bf-e87027e9e8fa | hello-test | 85 | deployed | |
| 81 | e9da2ab9-8864-405f-87ff-73a8ba2b6e0d | echo-server-integration-14 | 82 | rendered | |
| 80 | 518ba3ca-386a-48b1-935b-13a9390175da | echo-server-integration-13 | 82 | rendered | |
| 79 | 4f2463d1-9a2b-40e6-b0a2-8080e6d01e19 | echo-server-integration-12 | 82 | rendered | |
| 78 | 094846de-3b32-4339-91e5-a1d07f5e8e03 | echo-server-integration-11 | 82 | rendered | |

Figure 95: NEMO workload instances and their respective status in LCM UI

The workload is getting validated by the NEMO Workload Validator. Its functionality is described in section 3.4.2.5 which presents the validation checks that are performed by the validator. Once the tests have been successfully passed (Figure 96) then the workload upload request is dispatched.

```
api.serializers.workload DEBUG 2024-12-05 14:02:12,974 workload Workload Document echo-server v0.5.13 passed helm chart structural validation
api.serializers.workload DEBUG 2024-12-05 14:02:13,066 workload Workload Document echo-server v0.5.13 passed helm chart template validation
api.serializers.workload DEBUG 2024-12-05 14:02:16,725 workload Workload Document echo-server v0.5.13 passed docker image access validation
api.serializers.workload DEBUG 2024-12-05 14:02:16,726 workload Workload Document echo-server v0.5.13 passed ingress support validation
django_server INFO 2024-12-05 14:02:30,333 basehttp "POST /api/v1/workload/upload/ HTTP/1.1" 201 0
```

Figure 96: NEMO workload validation

| | | | | | |
|----------------|---|----------------|-------|----------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 91 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | Status: | final | | |

Finally, the NEMO workload instance is deployed in the NEMO meta-OS platform as confirmed by the acknowledgment message received by the MO. The *workload ID* matches with the one that is visible in the LCM UI (Figure 97).

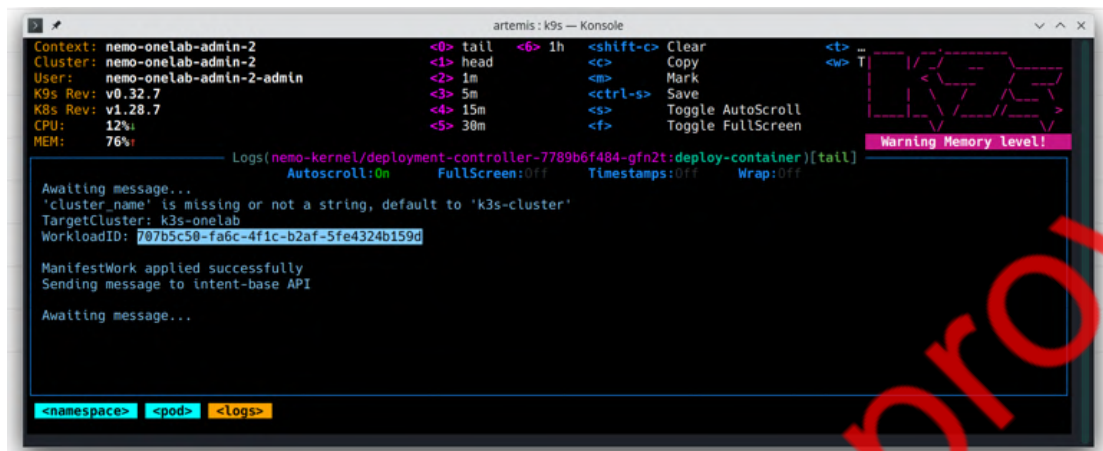


Figure 97: Workload deployment confirmation through RabbitMQ for the newly created workload instance

4.2.2.2 Meta-Orchestrator & Deployment Controller

The *Meta-Orchestrator (MO)* within its architecture has several subcomponents, including the *Deployment Controller (DC)*. This component handles communication, processes workload deployment configuration files, turns them into workloads' instances, and finally deploys those workloads' instances in a selected cluster. The *Intent-based API* sends the message with the workload to be deployed by the MO through RabbitMQ in JSON format, (Figure 98), and the message body created in the RabbitMQ queue, (Figure 99). The MO then processes that message, decoding it to adapt the JSON into a data structure within the programming language. It checks for metadata like labels and namespaces in the manifests and assigns defaults if they are missing.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 92 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |



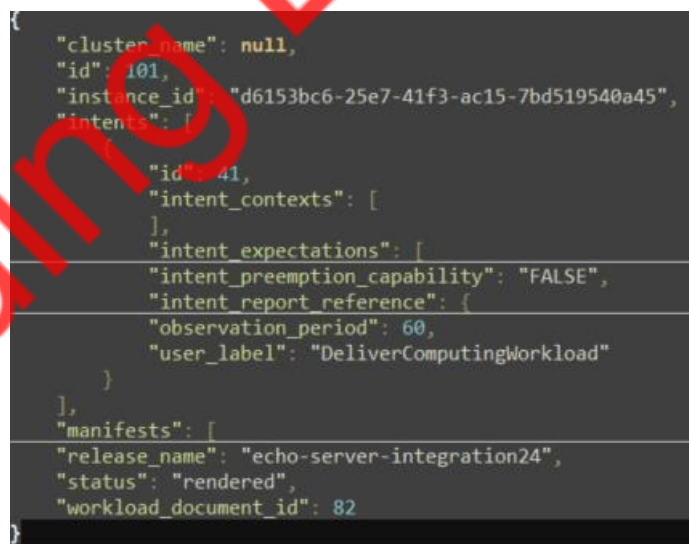
```

1  {
2    "release_name": "echo-server-integration24",
3    "values_override": {},
4    "include_crds": false,
5    "is_upgrade": false,
6    "namespace": "default",
7    "no_hooks": false,
8    "ingress_enabled": false,
9    "intents": [
10     {
11       "intent_type": "DeliverComputingWorkload",
12       "targets": [
13         {
14           "target_name": "cpuUsage",
15           "target_condition": "IS_LESS_THAN",
16           "target_value_range": "20"
17         },
18         {
19           "target_name": "ramUsage",
20           "target_condition": "IS_LESS_THAN",
21           "target_value_range": "100"
22         }
23       ]
24     }
25   ]
26 }

```

Figure 98: Intent-based API endpoint where the scenario starts.

Moreover from Figure 98 is important remark that the future workload has the name “echo-server-integration24” and it has a workload id with the value: “bf4c24eb-c263-4854-b886-51b915d79264”.



```

{
  "cluster_name": null,
  "id": 101,
  "instance_id": "d6153bc6-25e7-41f3-ac15-7bd519540a45",
  "intents": [
    {
      "id": 41,
      "intent_contexts": [
      ],
      "intent_expectations": [
        {
          "intent_preemption_capability": "FALSE",
          "intent_report_reference": {
            "observation_period": 60,
            "user_label": "DeliverComputingWorkload"
          }
        }
      ],
      "manifests": [
        {
          "release_name": "echo-server-integration24",
          "status": "rendered",
          "workload_document_id": 82
        }
      ]
    }
  ]
}

```

Figure 99: JSON published in RabbitMQ to be consume by MO.

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 93 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```
PS C:\Projects\NEMO\deployment-controller\deployment-controller> k get pods --context k3s-onelab
NAME                                READY   STATUS    RESTARTS   AGE
external-dns-6449884664-whp9r      1/1     Running   1785 (11m ago)   30d
ingress-nginx-controller-d49697d5f-df6jw  1/1     Running   1 (29d ago)      29d
```

Figure 100: Target cluster without the workload.

In Figure 100, before the workload deployment we can check that any workload is already deployed. Once the MO triggers the workload deployment, we can see in Figure 101 the pods of the workload “echo-server-integration24” deployed in the *k3s-onelab* as the target cluster chosen.

```
PS C:\Projects\NEMO\deployment-controller\deployment-controller> k logs -f deployment-controller-5dddcbbcf8-z5djc
Awaiting messages...
ManifestWork 'bf4c24eb-c263-4854-b886-51b915d79264' applied successfully in namespace 'k3s-onelab'
Deployment info sent: {WorkloadID:bf4c24eb-c263-4854-b886-51b915d79264 Type:deployment TargetCluster:k3s-onelab Timestamp:20241205163401}
```

Figure 101: Deployment Controller log, receiving the workload petition and deploying it.

In parallel, the MO asks the CFDRl which cluster is best for deploying a workload; the CFDRl recommends a specific cluster in direct communication with the MO. The recommendation of the CFDRl which is the product of the inference that stems from AI model that CFDRl incorporates, was simulated as the development of the component is in progress. Once the MO has the workload ready and the CFDRl recommendation, the MO deploys the workload in the chosen cluster using the OCM libraries and the NEMO cluster network. The results are showed in Figure 102 up and running as pods.

```
PS C:\Projects\NEMO\deployment-controller\deployment-controller> k get pods --context k3s-onelab
NAME                                READY   STATUS    RESTARTS   AGE
echo-server-integration24-757bf868fb-cd7gg  1/1     Running   0           12m
echo-server-integration24-test-connection  0/1     Completed 0           12m
external-dns-6449884664-whp9r      1/1     Running   1785 (27m ago)   30d
ingress-nginx-controller-d49697d5f-df6jw  1/1     Running   1 (29d ago)      29d
```

Figure 102: Workload already deployed in the cluster selected.

Finally, in Figure 103, the DC publishes the previous message into RabbitMQ to update the workload status. The Intent-Base API and other components will read this status.

```
{
  "workloadID": "bf4c24eb-c263-4854-b886-51b915d79264",
  "type": "deployment",
  "targetCluster": "k3s-onelab",
  "timestamp": "2024-12-04 12:28:20"
}
```

Figure 103: Deployment Controller (MO) final response.

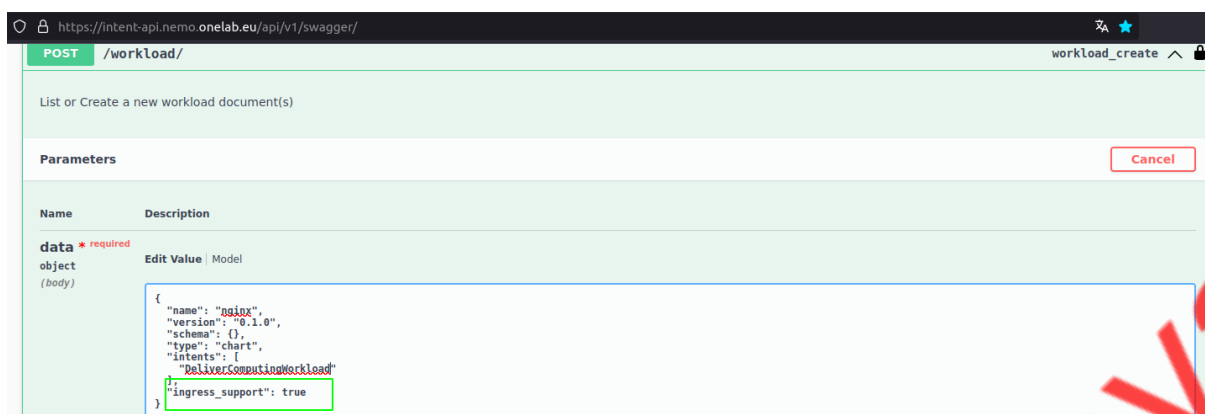
4.2.2.3 Access control provisioning

In deliverable D4.1 [2], the deployment and integration of the *NEMO Access Control* with the *Intent-based API* were presented. For the integration part, we had developed an API that would receive a payload with the necessary information to properly set the workload in the Access Control (set the Kong services, routes and plugins). In this section, we will present the improvements made to the Access Control workload provisioning to better automate and simplify the workflow.

During the initial creation of a workload, the Intent-based API offers the ability to choose whether to deploy the workload with an Ingress or not. For this scenario, we will create a workload for a simple NGINX²¹ server (Figure 104).

²¹ <https://nginx.org/>

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 94 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |



POST /workload/

List or Create a new workload document(s)

Parameters

Cancel

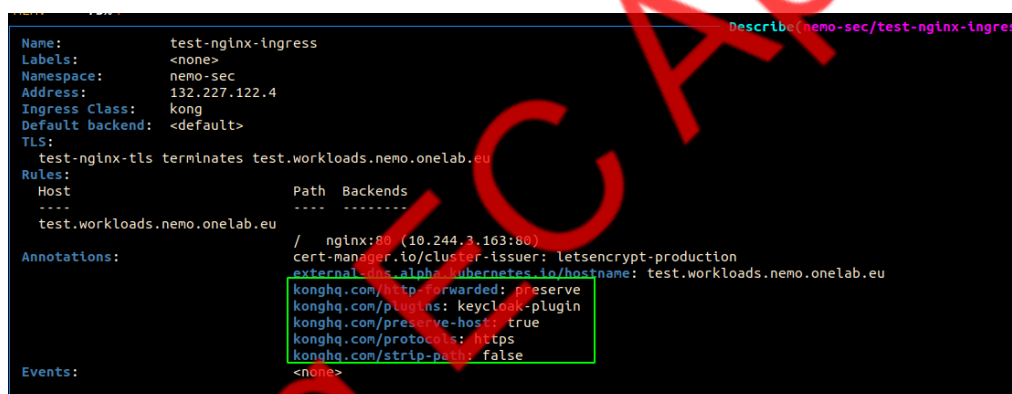
Name Description

data * required object (body)

```
{
  "name": "nginx",
  "version": "0.1.0",
  "schema": {},
  "type": "chart",
  "intents": [
    "DeliverComputingWorkload"
  ],
  "ingress_support": true
}
```

Figure 104: Create workload through Intent-based API with Ingress support

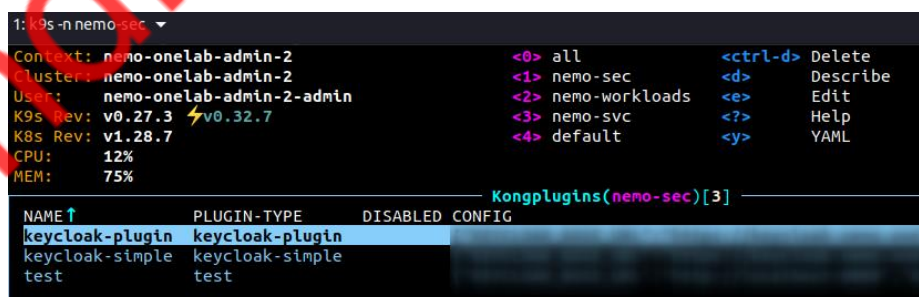
When the workload is successfully created, the *Intent-based API* produces an Ingress that holds specific annotations to integrate with the NEMO Access Control and enable the *oAuth 2.0 plugin* for limiting access to unauthorized users. Figure 105 shows the annotations added to the Ingress to specify details for the services and routes that will be created in the Access Control (*konghq.com/protocols*, *konghq.com/http-forwarded* etc), as well as the plugins that will be applied to the deployment workload (*konghq.com/plugins*).



```
Name: test-nginx-ingress
Labels: <none>
Namespace: nemo-sec
Address: 132.227.122.4
Ingress Class: kong
Default backend: <default>
TLS:
  test-nginx-tls terminates test.workloads.nemo.onelab.eu
Rules:
  Host Path Backends
  ----
  test.workloads.nemo.onelab.eu / nginx:80 (10.244.3.163:80)
Annotations:
  cert-manager.io/cluster-issuer: letsencrypt-production
  external-dns.alpha.kubernetes.io/hostname: test.workloads.nemo.onelab.eu
  konghq.com/http-forwarded: preserve
  konghq.com/plugins: keycloak-plugin
  konghq.com/preserve-host: true
  konghq.com/protocols: https
  konghq.com/strip-path: false
Events: <none>
```

Figure 105: Workload Ingress annotations for integrating with Access Control

The *oAuth 2.0* configuration (named *keycloak-plugin*) is deployed in the OneLab cluster as a *Kongplugin* [61] resource. This allows for the plugin to already be configured and ready to apply to new Ingresses. Figure 106 shows the Kong plugins resources available in the OneLab cluster.



```
1: k9s -n nemo-sec
Context: nemo-onelab-admin-2
Cluster: nemo-onelab-admin-2
User: nemo-onelab-admin-2-admin
K9s Rev: v0.27.3
K8s Rev: v1.28.7
CPU: 12%
MEM: 75%

NAME↑ PLUGIN-TYPE DISABLED CONFIG
keycloak-plugin keycloak-plugin
keycloak-simple keycloak-simple
test test
```

Figure 106: The Onelab KongPlugin resources

| | | | | | |
|----------------|---|----------------|----|----------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 95 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

When the Ingress is successfully deployed to the cluster, the Access Control is automatically updated with the specifications in the Ingress annotations. Figure 107 and Figure 108 show the Access Control service that was created and its details, respectively.

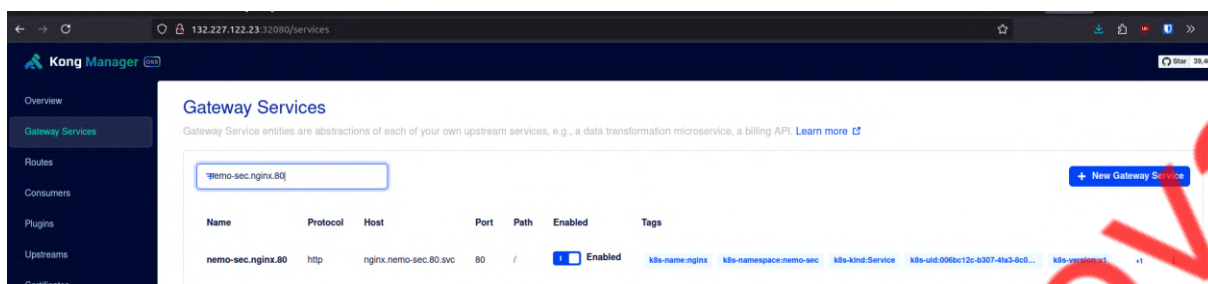


Figure 107: Workload Access Control service

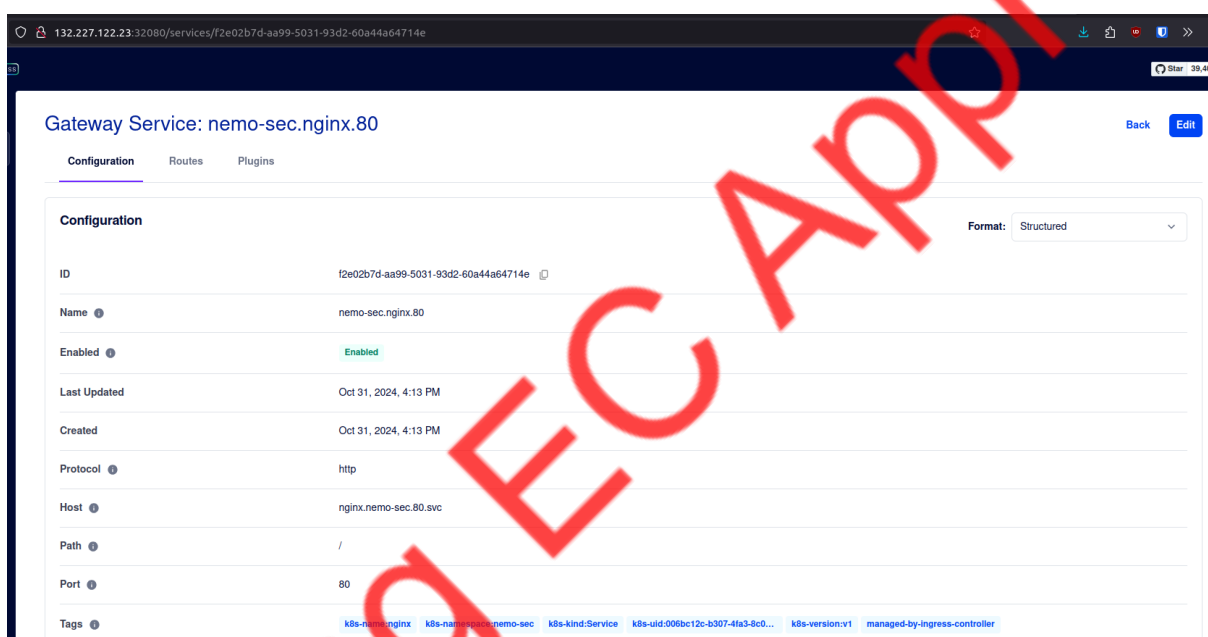


Figure 108: Workload Access Control service details

Figure 109 and Figure 110 show the Access Control route that was created and its details, respectively. In both the service and the route details, we can see that the name of the workload, its path and its Ingress host name have all been registered successfully.

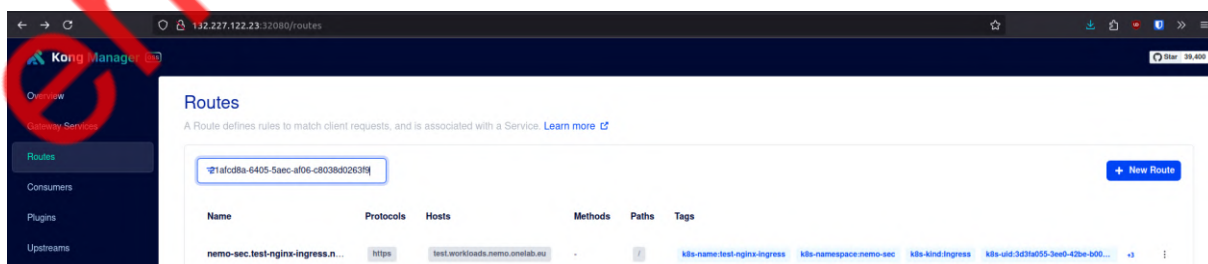


Figure 109: Workload Access Control route

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 96 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

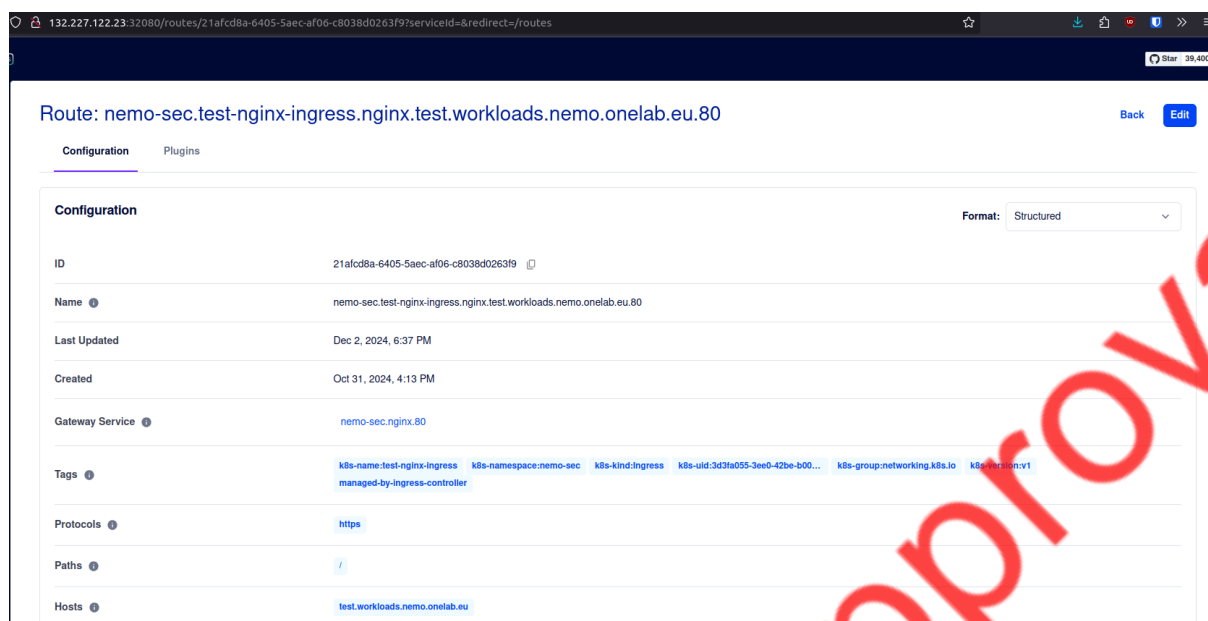


Figure 110: Workload Access Control route details

Finally, Figure 111, Figure 112 and Figure 113 present the oAuth2.0 plugin and its details.

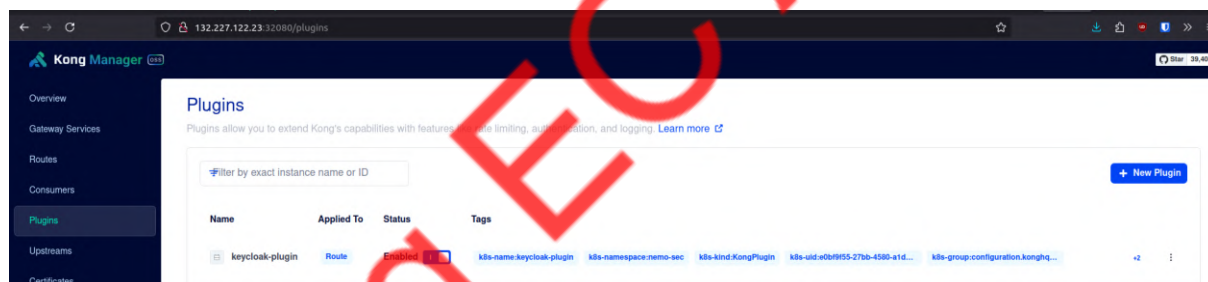


Figure 111: Workload oAuth2.0 plugin

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 97 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

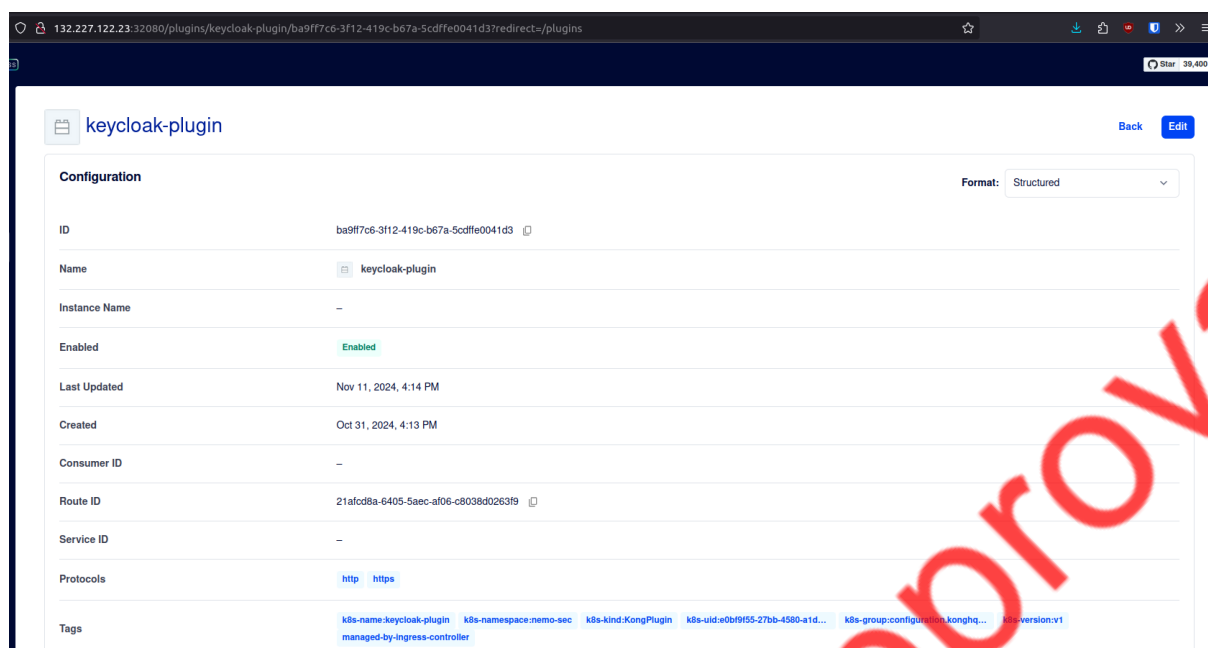


Figure 112: Workload oAuth2.0 plugin details

Plugin Specific Configuration

| | |
|--|------------------------------------|
| KEYCLOAK DB HOST | postgres.default.svc.cluster.local |
| KEYCLOAK DB PORT | 5432 |
| KEYCLOAK DB NAME | |
| KEYCLOAK BASE URL | https://keycloak.nemo.onelab.eu |
| KEYCLOAK REALM NAME | |
| KEYCLOAK CLIENT ID | |
| KEYCLOAK CLIENT SECRET | |
| KEYCLOAK BYPASS SERVER TOKEN INTROSPECTION | True |
| KEYCLOAK HASHING ALGORITHM | |
| KEYCLOAK PUBLIC KEY | |
| KEYCLOAK DB USER | |
| KEYCLOAK DB PASSWORD | |

Figure 113: Workload oAuth2.0 plugin (cont'd)

Now that the NEMO Access Control has been configured properly, we can test how the access to the workload works. In order to access the protected resource, all the requests should provide an *authorization* header with a Bearer token from the *NEMO Identity Management component*. The oAuth2.0 plugin is configured to test the provided token and, depending on its validity, deny or grant access. In Figure 114, the request provides no authorization header, therefore the user is denied access.

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|-----------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 98 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

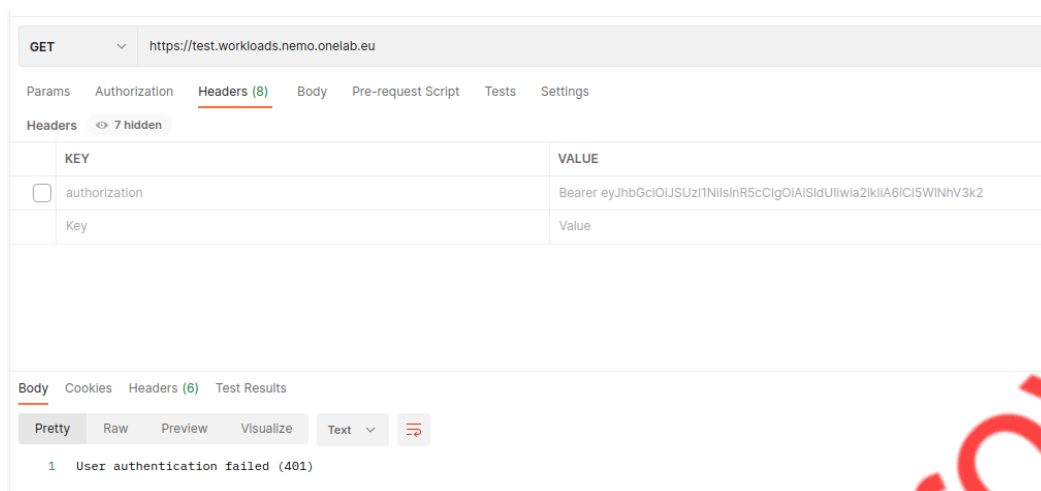


Figure 114: NEMO oAuth2.0 plugin test

In Figure 115, even if the token is provided, it could have expired, been tampered with or come from an unknown source. The plugin, again, denies access to the NGINX server.

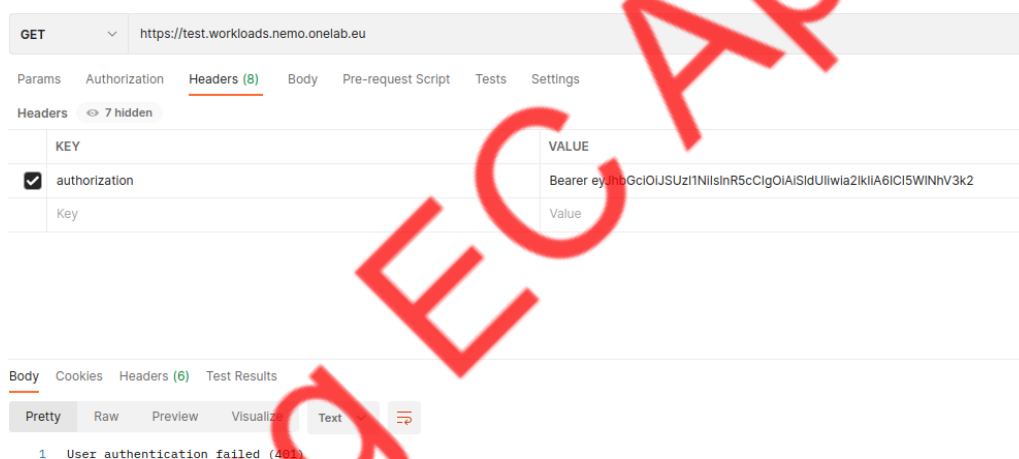


Figure 115: oAuth2.0plugin test - expired or false token

Finally, if the token is valid, the user can view the resource, in this case the welcome page of the NGINX server (Figure 116).

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 99 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

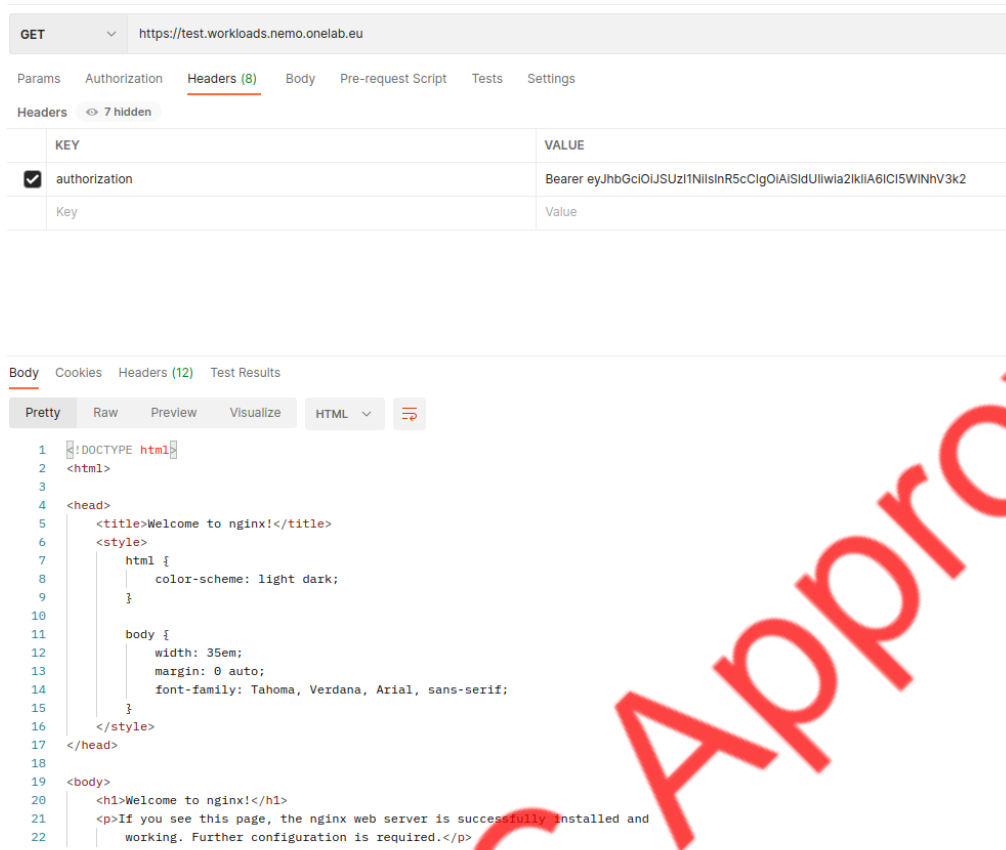


Figure 116: oAuth2.0 plugin test - success

4.2.2.4 NEMO Access Control plugin response performance

The plugin, in order to reduce the time needed to validate the provided token, instead of using an HTTP request to connect to the *NEMO Identity Management component*, it connects directly to its database to perform the user token validation.

This decision has come from studying the code of the Identity Management component for the token introspection²². The outcome of this study indicated that the Identity Management component performs several checks which are not necessary for the project use cases. These can add significantly to the plugin response time, particularly under load-testing scenarios. Therefore, the oAuth2.0 implementation with the direct database connection performs only the necessary checks to verify the *token*, *user*, *realm* and *client_id* validity.

Notably, in order to increase the overall system performance and redundancy and avoid security risks that stem from directly connecting to the Identity Management database, the latter has been configured with streaming replication (WAL). Under this framework, the plugin does not directly connect to the master instantiation of the database itself but, instead, it connects to a ready-only live replica. Additionally, the database user used by the plugin has been configured with restricted access to the tables that are strictly necessary for the token introspection.

We will now present the performance of the plugin when it directly connects to the database, versus when it uses requests to perform the token validation. To conduct the performance tests, we are using

²²

<https://github.com/keycloak/keycloak/blob/83f8622d15d9a3559ee6d99a4c57033190a5392d/services/src/main/java/org/keycloak/protocol/oidc/endpoints/TokenIntrospectionEndpoint.java#L72>

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 100 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

Locust²³, an open-source tool, which gives information on metrics, like the percentiles of the response times, the number of requests, minimum, max and average request times. For both experiments we will use the same setup parameters (Figure 117): (a) the max number of users will reach 20, the users will increase every one second and (b) each experiment will last 5 minutes. Each request will be executed randomly between the span of 1 and 5 seconds. The first run will use the implementation of the oAuth2.0 plugin that make a direct connection to the database (*standard oAuth2.0 plugin*), while in the second one we will apply to the same Ingress the version of the plugin which makes requests to the Identity Management API (*simplified oAuth2.0 plugin*).

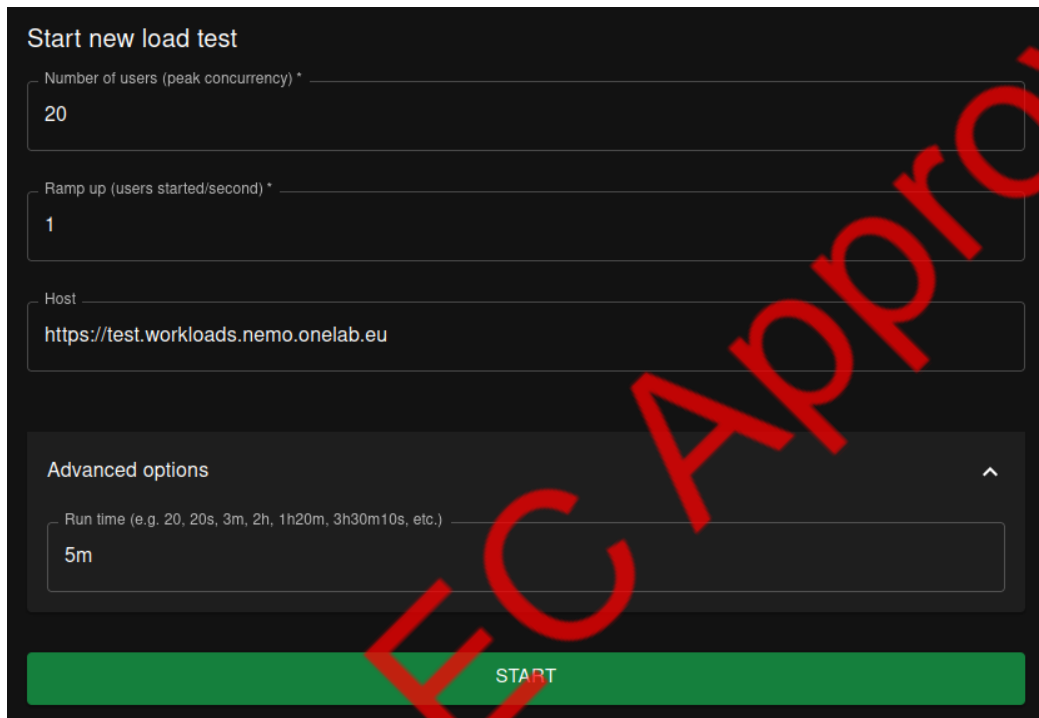


Figure 117: Locust experiments' setup

Figure 118 and Figure 119 present the request and response statistics of Locust for the standard plugin. In the request statistics table, we can see the total number of requests, the average, minimum and max times of the requests. If we compare the two averages, we can see that the average request time of the simplified plugin is approximately four times larger than the standard implementation. The percentiles, also, in the response statistics table, show that the simplified version has greater response times in comparison.

²³ <https://locust.io>

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 101 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

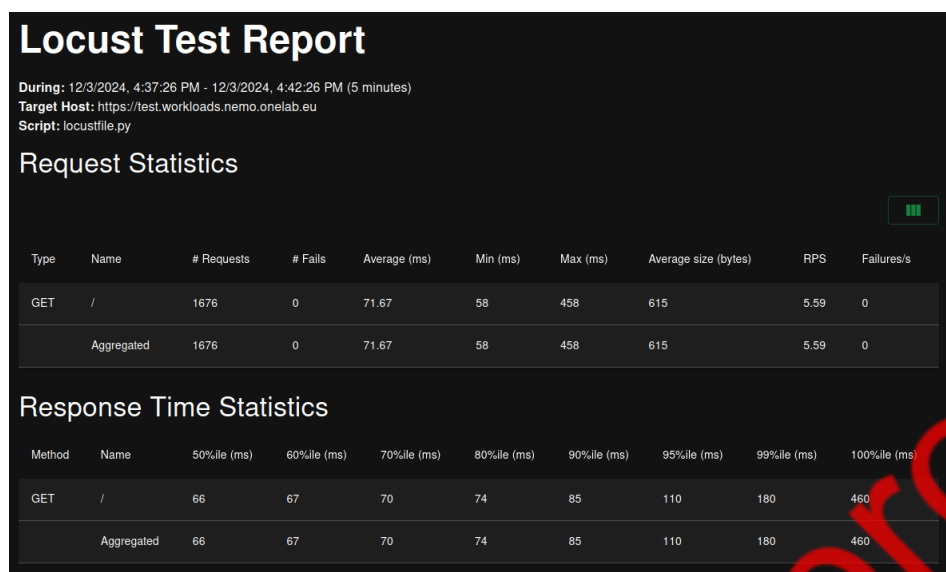


Figure 118: Locust request and response statistics for standard *oAuth2.0* plugin

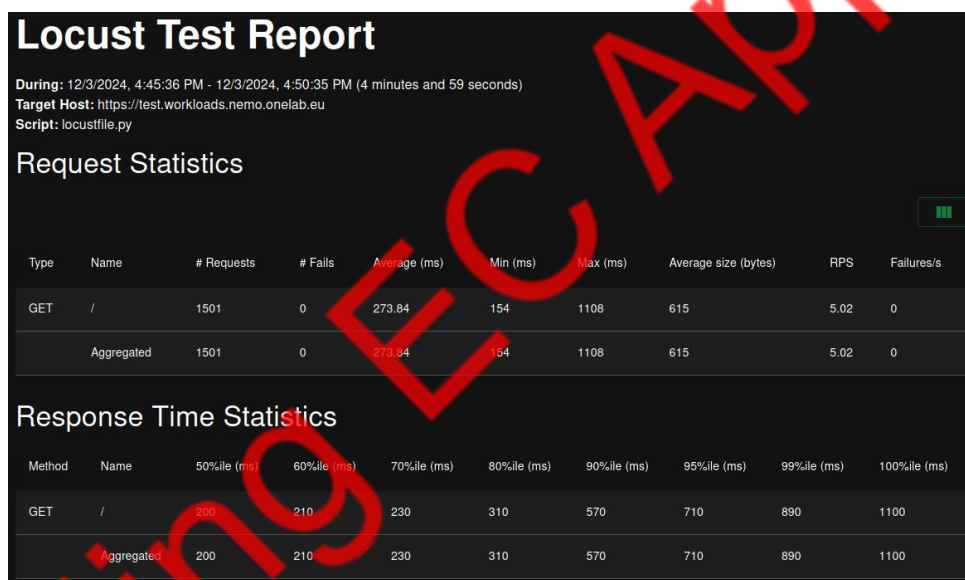


Figure 119: Locust request and response statistics for simplified *oAuth2.0* plugin

Figure 120 compares side by side the two runs. If we compare the response times, we can see that in the first run, the plugin requires less time to stabilize the response times when the number of users have peaked for the experiment. During the first execution the response times were almost half of the second one. Finally, the deviation of the 50th and 95th percentiles in the first run has a smaller deviation, meaning that the majority of the users will experience a stable experience.

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 102 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

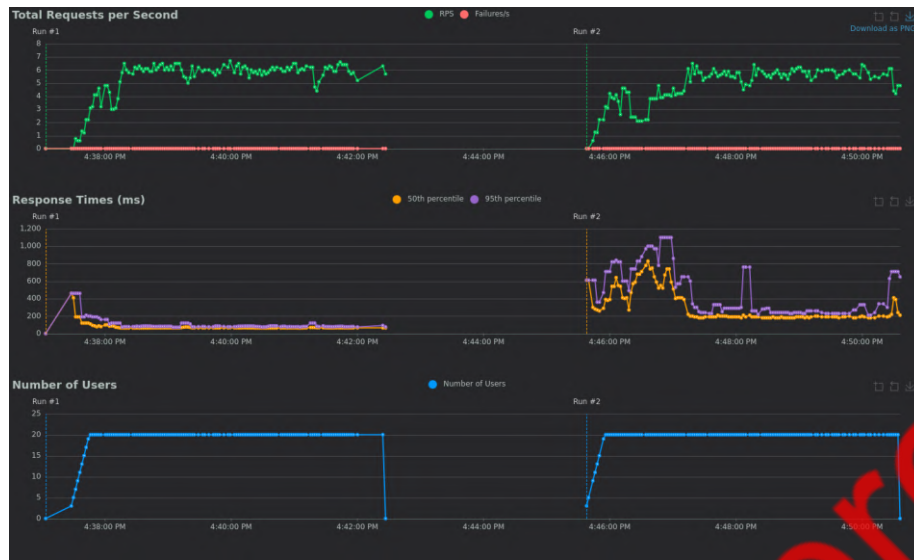


Figure 120: Locust charts for the *OAuth2.0* plugin implementations

4.2.3 Verification summary checklist

| Checklist for Test2: NEMO workload registration, deployment and provisioning | | | | |
|--|---|-----|----|----------------|
| | | Yes | No | Comments |
| 1 | NEMO workload registration by the NEMO user through LCM UI | ✓ | | Success |
| 2 | NEMO workload registration through the Intent-based API | ✓ | | Success |
| 3 | Notification of the LCM UI about the status of the workload registration | ✓ | | Success |
| 4 | Execution of workload validation in Intent-based API | ✓ | | Success |
| 5 | NEMO workload deployment by the NEMO user through LCM UI | ✓ | | Success |
| 6 | Communication of the deployment acknowledgement to LCM UI | ✓ | | Success |
| 7 | Communication of the deployment request to the MO | ✓ | | Success |
| 8 | CFDRL deployment recommendation to MO | | ✓ | Simulated step |
| 9 | Deployment operation process executed by the MO | ✓ | | Success |
| 10 | Communication and update of the deployment operation status to the Intent-based API | ✓ | | Success |
| 11 | Visualization of the updated status to the LCM UI | ✓ | | Success |
| 12 | NEMO workload provisioning triggered by the Intent-based API | ✓ | | Success |
| 13 | NEMO Access Control workload setup process | ✓ | | Success |
| 14 | NEMO Access Control Keycloak plugin functionality executed | ✓ | | Success |
| 15 | Performance resilience of the Kong Plugin | ✓ | | Success |

Table 10: Checklist for workload registration, deployment and provisioning scenario

| | | | | | |
|----------------|---|----------------|----|----------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 103 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

4.3 NEMO workload migration

This integration scenario aims to illustrate the workload migration process. The process is facilitated by the Intent-based API, the MO and the Intent-based Migration Controller (IBMC). Figure 121, depicts the steps executed to complete the task.

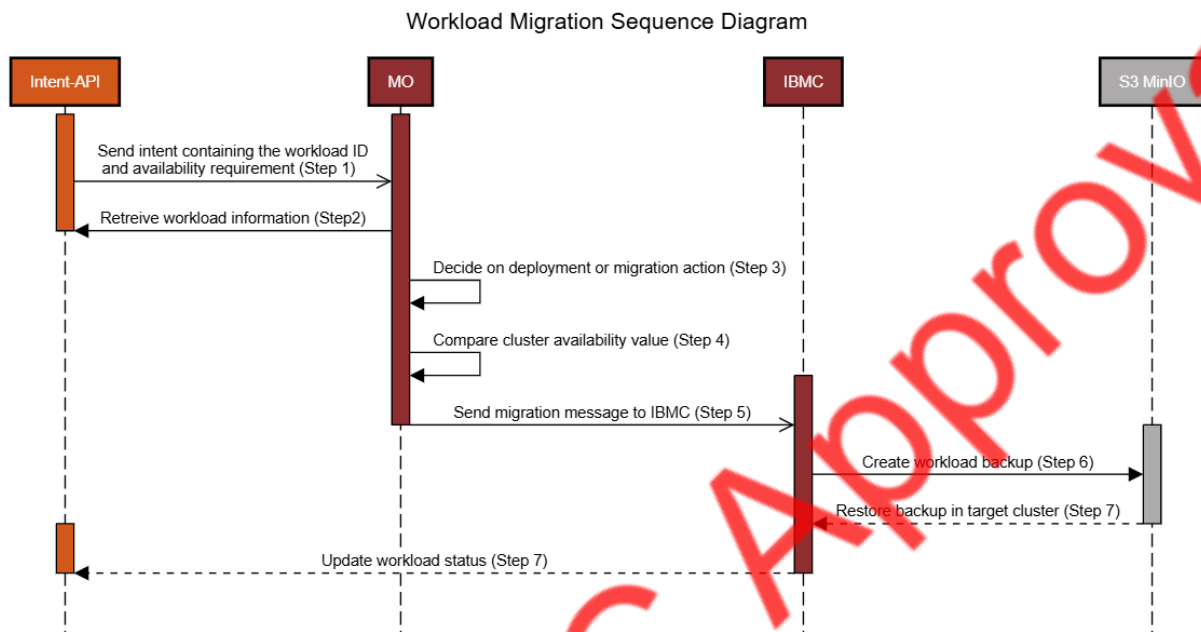


Figure 121: NEMO workload migration sequence diagram

4.3.1 Verification scenario

| Test 3: NEMO workload migration | |
|---------------------------------|--|
| Objective | The objective of this task is to validate the NEMO workload migration process. |
| Components | IBMC Intent-API MO MinIO |
| Features to be tested | The NEMO workload migration process is triggered by the CFDRl component once its inference states that it is preferable for the workload's optimal operation to be moved from cluster A to cluster B. Once the request is communicated to the Meta-Orchestrator component then the workload migration is executed. |
| Test setup | All the participating components were deployed in the NEMO meta-OS cloud/edge infrastructure at OneLab (dev cluster 1 and staging cluster 1). |
| Steps | <ol style="list-style-type: none"> 1. Intent-API publishes a workload intent with an availability requirement. 2. MO retrieves workload status from the Intent-API. |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 104 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | |
|--|---|
| | <ol style="list-style-type: none"> 3. If the workload is already deployed in any cluster, then a migration action is triggered. 4. Check cluster availability. 5. MO sends a message to the IBMC containing the workload ID and the target cluster. 6. IBMC backs up the workload resources and uploads them MinIO. 7. IBMC restores the resources in the target cluster. After the resources get restored, the workload is removed from the source cluster. 8. The IBMC sends a message to the Intent-API updating the workload status, specifying the cluster where it has been deployed. |
|--|---|

Table 11: Test 3 - NEMO workload migration

4.3.2 Results

The section below presents the steps that concern the execution of the workload migration integration workflow.

4.3.2.1 Step 1

An intent is published by the Intent-Based API in RabbitMQ and reaches the Meta-Orchestrator:

```
Awaiting message...
Received message:
intent_type: Availability
target_condition: IS_EQUAL_TO
target_value_range: 99.9
instance_id: 518ba8ca-386a-48c1-935b-13e939b77db6
```

Figure 122: Intent message reaches MO

4.3.2.2 Step 2 & 3

The Meta-Orchestrator sends a query back to the Intent-Based API to retrieve a json with the workload status. As shown in the following code snippet, the workload appears to be already deployed in oneLab cluster, hence a migration action will take place:

```
{
  "id": 6,
  "instance_id": "a3177d01-863d-415b-a998-180c87113z50",
  "workload_document_id": 9,
  "release_name": "migration-workload",
  "status": "deployed",
  "manifests": [],
  "cluster_name": onelab
}
```

This can be verified in the oneLab cluster by executing `kubectl get pods --context onelab`. The list of pods shows the workload running:

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 105 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

| NAME | READY | STATUS | RESTARTS | AGE |
|--|-------|---------|---------------|-------|
| deployment-controller-5dddcbcbf8-qns8t | 1/1 | Running | 0 | 68m |
| ibmc-599b9689dd-vfns8 | 1/1 | Running | 4 (27d ago) | 43d |
| migration-workload | 1/1 | Running | 0 | 108s |
| mo-api-db77c94c4-zk9bq | 3/3 | Running | 3 (2d19h ago) | 2d19h |
| mo-cluster-mgm-agent-75459f84d7-5j2ms | 1/1 | Running | 0 | 2d18h |
| mo-metrics-586545cbd9-5qnn2 | 1/1 | Running | 0 | 55d |
| undeployment-controller-5c8c475ccc-72kvd | 1/1 | Running | 0 | 20h |
| velero-68fccd5dc4-fwvll | 1/1 | Running | 0 | 44d |

Figure 123: Pods currently running in onelab

When `kubectl describe pod migration-workload` is executed, the workload ID can be found in the “labels” metadata:

```

Name: migration-workload
Namespace: nemo-kernel
Priority: 0
Service Account: default
Node: k8sworker5.onelab.eu/192.168.111.144
Start Time: Thu, 05 Dec 2024 15:07:21 +0100
Labels: nemo.eu/workload=a3177d01-863d-415b-a998-180c87113250
Annotations: k8s.v1.cni.cncf.io/network-status:
  [{
    "name": "cbr0",
    "interface": "eth0",
    "ips": [
      "10.244.5.13"
    ],
    "mac": "aa:75:d8:66:b2:39",
    "default": true,
    "dns": {},
    "gateway": [
      "10.244.5.1"
    ]
  }]
Status: Running

```

Figure 124: Workload ID inspection

4.3.2.3 Step 4

The MO proceeds to check its internal database, which contains the availability information of every cluster. The availability value specified in the intent is compared with the one from the cluster where the workload is currently deployed. As seen in the Figure 125, the availability of the OneLab cluster (90%) is lower than the one required (99.9%). This will trigger the migration of the workload to a more suitable cluster. In this case, this is the k3s one.

```

nemo_database> db.clusters.find().pretty()
[
  {
    _id: ObjectId('6750606c737ba9f7e2c1a0fe'),
    creation_at: ISODate('2024-12-04T14:00:12.375Z'),
    cluster_name: 'k3s-onelab',
    managed_api: 'https://api.s2.nemo.onelab.eu:6443',
    availability: '99%'
  },
  {
    _id: ObjectId('6750606c737ba9f7e2c1a0ff'),
    creation_at: ISODate('2024-08-07T12:06:00.200Z'),
    cluster_name: 'onelab-hub',
    managed_api: 'https://api.main.nemo.onelab.eu:6443',
    availability: '90%'
  }
]

```

Figure 125: Availability check

4.3.2.4 Step 5 & 6

The MO sends a message via RabbitMQ to the IBMC containing the source and target clusters for the migration and the workload to be migrated. When the message reaches the IBMC, the migration process begins with the backup of the workload’s resources, which is stored in the OneLab’ MinIO instance.

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 106 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```
Awaiting message...
Received a message:
  sourceCluster: onelab
  targetCluster: onelab-k3s
  migrationAction: backup
  workload: a3177d01-863d-415b-a998-180c87113z50
Backup is still in progress. Waiting...

Backup completed succesfully

Awaiting message...
```

Figure 126: Migration message reaches source cluster's IBMC instance

The backup status can be checked at any moment by executing `velero get backup -n nemo-kernel`.

| NAME | STATUS | ERRORS | WARNINGS | CREATED | EXPIRES | STORAGE LOCATION | SELECTOR |
|-----------------------------|-----------|--------|----------|-------------------------------|---------|------------------|---|
| onelab-backup20241205154728 | Completed | 0 | 0 | 2024-12-05 16:47:28 +0100 CET | 29d | default | nemo.eu/workload=a3177d01-863d-415b-a998-180c87113z50 |

Figure 127: Backup status

Once the backup is completed, a message is sent to the target cluster in order to continue with the migration process.

4.3.2.5 Step 7

The IBMC instance running in the target cluster receives the message from the source cluster and proceeds to restore the workload resources.

```
Awaiting message...
Received a message:
  backupName: onelab-backup20241205144728
  sourceCluster: onelab
  targetCluster: onelab-k3s
  migrationAction: restore
  workload: a3177d01-863d-415b-a998-180c87113z50
Restore is still in progress. Waiting...
```

Figure 128: Restore message reaches target cluster's IBMC instance

If `kubectl get pods -context onelab-k3s` is executed before the migration, it can be observed that the workload doesn't exist in the cluster:

| NAME | READY | STATUS | RESTARTS | AGE |
|-------------------------|-------|---------|----------|-----|
| ibmc-b9c8cc886-2mn24 | 1/1 | Running | 0 | 28d |
| velero-7954d57cdc-2g9bq | 1/1 | Running | 0 | 29d |

Figure 129: k3s cluster status before migration

Once the restore is completed, the workload is correctly deployed:

| NAME | READY | STATUS | RESTARTS | AGE |
|-------------------------|-------|---------|----------|-----|
| ibmc-b9c8cc886-2mn24 | 1/1 | Running | 0 | 28d |
| migration-workload | 1/1 | Running | 0 | 14m |
| velero-7954d57cdc-2g9bq | 1/1 | Running | 0 | 29d |

Figure 130: k3s cluster status after migration completion

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 107 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

Name: migration-workload
Namespace: nemo-kernel
Priority: 0
Service Account: default
Node: nemo-k3s-node-1/132.227.122.88
Start Time: Thu, 05 Dec 2024 16:47:52 +0100
Labels: nemo.eu/workload=a3177d01-863d-415b-a998-180c87113z50
        velero.io/backup-name=onelab-backup20241205154728
        velero.io/restore-name=onelab-k3s-restore20241205154748
Annotations: k8s.v1.cni.cncf.io/network-status:
              [{
                "name": "cbr0",
                "interface": "eth0",
                "ips": [
                  "10.244.5.48"
                ],
                "mac": "5e:25:78:4a:09:fe",
                "default": true,
                "dns": {},
                "gateway": [
                  "10.244.5.1"
                ]
              }]
Status: Running

```

Figure 131: Description of workload in k3s cluster

The source cluster can be checked to verify that the workload has been deleted from it:

| NAME | READY | STATUS | RESTARTS | AGE |
|--|-------|---------|---------------|-------|
| deployment-controller-5dddcbbcf8-qn58t | 1/1 | Running | 0 | 3h14m |
| ibmc-599b9689dd-hnkct | 1/1 | Running | 1 (76m ago) | 119m |
| mo-api-db77c94c4-zk9bq | 3/3 | Running | 3 (2d21h ago) | 2d21h |
| mo-cluster-mgm-agent-75459f84d7-5j2ms | 1/1 | Running | 0 | 2d21h |
| mo-metrics-586545cbd9-5qnn2 | 1/1 | Running | 0 | 56d |
| undeployment-controller-5c8c475ccc-72kvd | 1/1 | Running | 0 | 23h |
| velero-68fcd5dc4-fwvll | 1/1 | Running | 0 | 44d |

Figure 132: OneLab cluster after migration

4.3.2.6 Step 8

When the migration is successfully completed, a message is sent to the Intent-Based API updating the workload status:

```

SourceCluster: onelab
TargetCluster: onelab-k3s
WorkloadID: a3177d01-863d-415b-a998-180c87113z50
Status: migrated

```

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 108 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

4.3.3 Verification summary checklist

| Checklist for Test3: NEMO workload migration | | | | |
|--|---|-----|----|----------|
| | | Yes | No | Comments |
| 1 | The CFDRDL issues the workload migration request to the MO | ✓ | | Success |
| 2 | Intent-API publishes a workload intent with an availability requirement. | ✓ | | Success |
| 3 | MO retrieves workload status from the Intent-API. | ✓ | | Success |
| 4 | If the workload is already deployed in any cluster, then a migration action is triggered. | ✓ | | Success |
| 5 | Check cluster availability. | ✓ | | Success |
| 6 | MO sends a message to the IBMC containing the workload ID and the target cluster. | ✓ | | Success |
| 7 | IBMC backs up the workload resources and uploads them MinIO. | ✓ | | Success |
| 8 | IBMC restores the resources in the target cluster. After the resources get resotred, the workload is removed from the source cluster. | ✓ | | Success |
| 9 | The IBMC sends a message to the Intent-API updating the workload status, specifying the cluster where it has been deployed. | ✓ | | Success |

Table 12: Checklist for Test3 - NEMO workload migration

4.4 NEMO workload lifecycle management

NEMO incorporates the concept of intents for the declarative description of requirements for workload execution and operation within the meta-OS. Intent management processes are integrated by design in NEMO operations. Once the NEMO workload is registered by the NEMO user through the LCM UI and the NEMO workload instance specified the intents that define the required operation for the NEMO workload the monitoring process starts. The monitoring process concerns both the cluster and the NEMO workloads' dynamic resource consumption properties that adhere to the defined expectation targets. The abovementioned measurements are collected by the PPEF component. The PPEF's specific functionality is described in detail in D3.2 [11]. In addition, complementary metrics about the network and health (among others) characteristics of a workload are also collected via the CMDT component. The latter will be documented in D2.3 [4] in full detail. The collected information is visualized through the LCM UI and is available to the NEMO user (workload provider). This particular scenario concerns also the policy enforcement and notification of the NEMO user in case of a policy breach (for an intent). As indicated in the process diagram below, this information is captured by the Intent-based API (which is notified by the PPEF in advance) and is communicated to the LCM component.

4.4.1 Process diagram

The process diagram presented below summarizes the NEMO workload lifecycle management that concerns the workload monitoring and management of the asset by the NEMO user. The NEMO components that provide cluster level and workload level measurements are included in the relevant scenario. The collected information is communicated through the RabbitMQ and the Intent-based API to the LCM UI where they are visualized to the NEMO user.

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 109 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

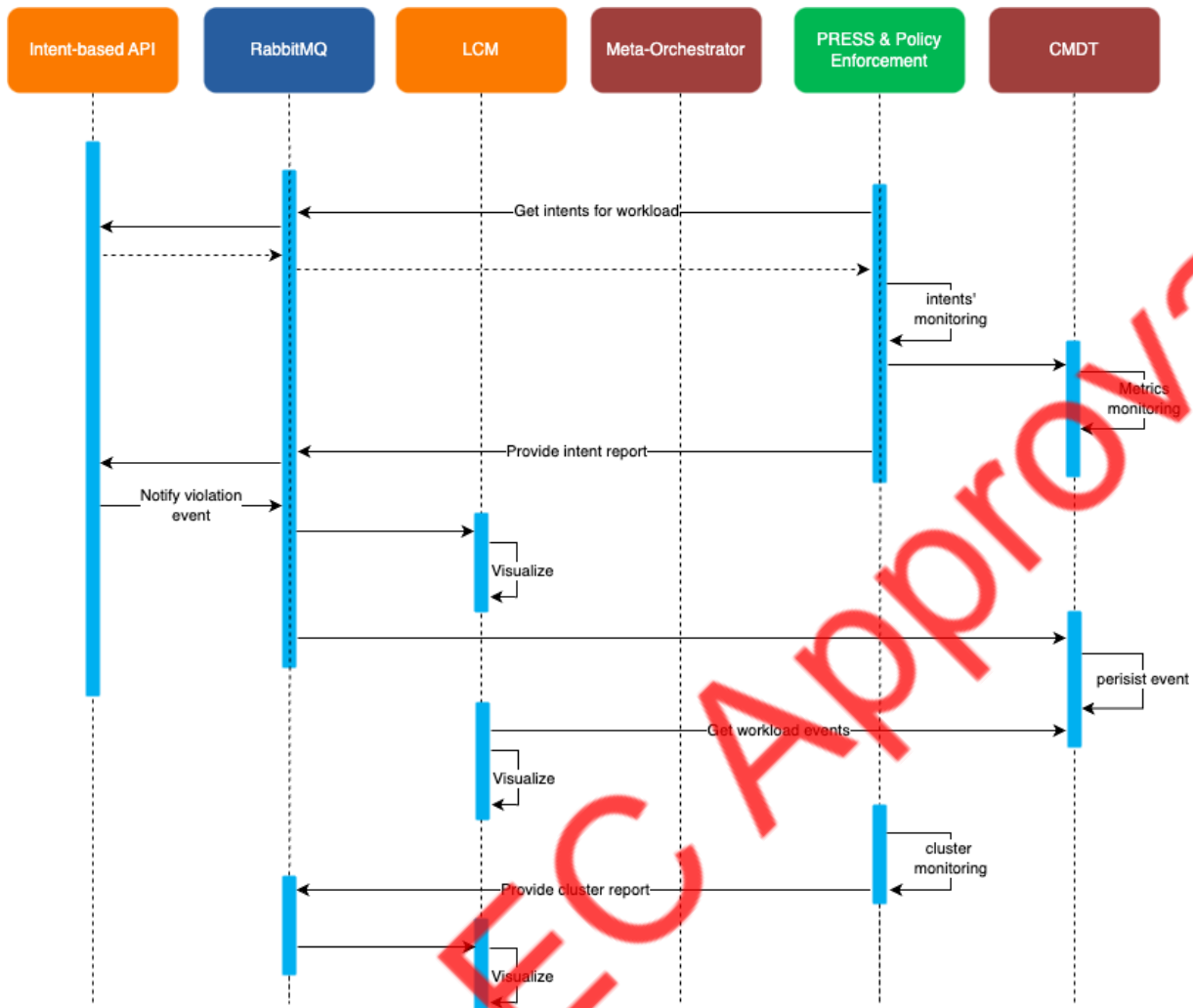


Figure 133: Process diagram for workload monitoring and enforcement

4.4.2 Verification scenario

| Test 4: NEMO workload lifecycle management | |
|--|--|
| Objective | To verify the workload lifecycle management process in NEMO covering all the steps identified. |
| Components | <ul style="list-style-type: none"> • LCM • Intent-API • PPEF • CMTD • RabbitMQ |
| Features to be tested | This integration scenario aims to validate the workload lifecycle management. The NEMO workload intents and complementary measurements that concern the resources' consumption and the resulting performance and liveness of a workload are collected by the PPEF and the CMTD |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 110 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | |
|------------|---|
| | components. From there they are communicated through the RabbitMQ to the LCM UI where there are visualized to the NEMO user. |
| Test setup | The associated components are deployed in OneLab facilities at NEMO dev cluster 1 The CFDRL component which is undergoes its final stages of development. |
| Steps | <ol style="list-style-type: none"> 1. The NEMO user accesses the LCM UI 2. NEMO workload monitoring collects metrics that correspond to the NEMO workload (PPEF) 3. The collected workload metrics are communicated to the Intent-API 4. The NEMO Cluster monitoring collects measurements that concern the NEMO meta-OS operated clusters (PPEF) 5. The collected cluster metrics are communicated to the RabbitMQ 6. NEMO workload complementary monitoring (CMDT) 7. The collected metrics are communicated to the Rabbit MQ 8. The LCM aggregates the collected metrics and visualize them to the NEMO user 9. The PPEF report intent violations to the Intent-based API |

4.4.3 Results

This section documents the process that is described in the scenario above step by step.

4.4.3.1 NEMO workload monitoring – CMDT

The CMDT collects network traffic characteristics and observes Kubernetes pod history. This is done through querying Kubernetes API, and Thanos/Prometheus. More detailed description of CMDT functionalities is available in D2.3 [4].

The Figure 134 illustrates how part of information is obtained through Prometheus to gain insight into network traffic characteristics, which were collected by Linkerd²⁴ service. The first query concerns pod's response rate per minute sorted by HTTP status code (5xx server side error, 2xx success), the second query provides the summary of maximum response latency for 99%, 95%, 75%, and 50% of connections, and the third the incoming request rate per minute.

²⁴ <https://linkerd.io/>

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 111 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

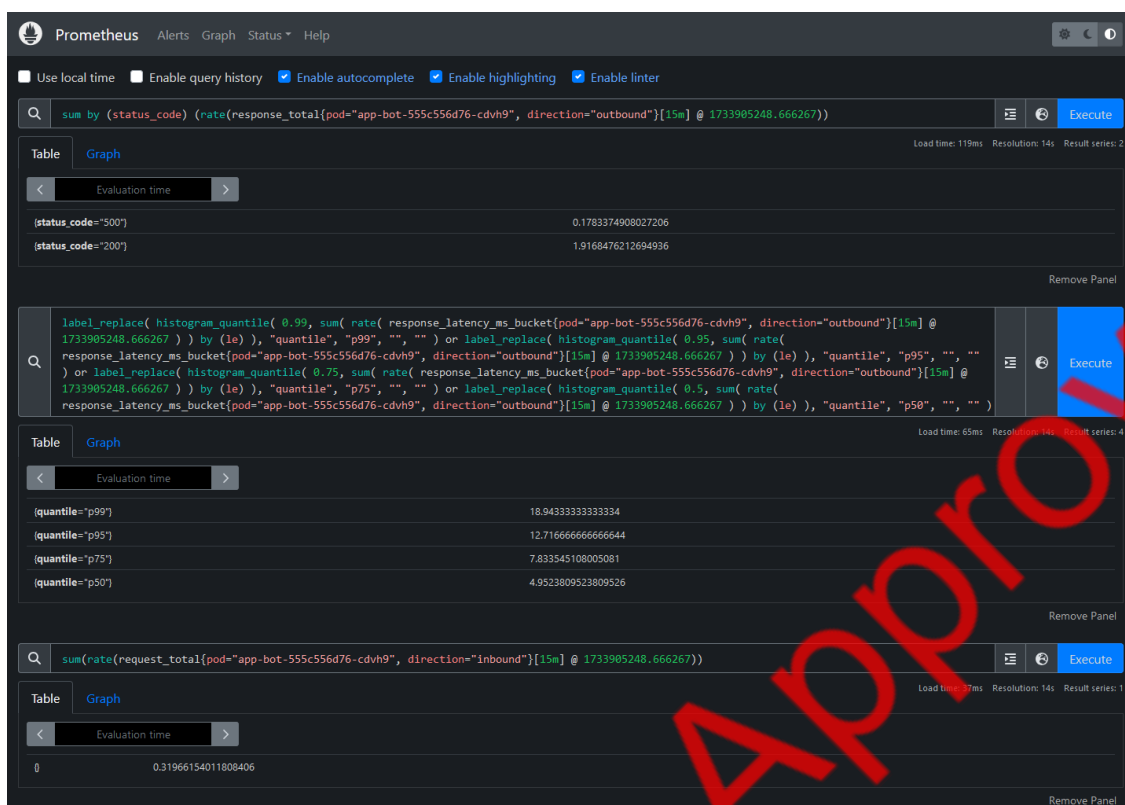


Figure 134: Three queries to obtain network traffic stats collected by CMDT through Linkerd

The CMDT instances then fuse data from different parts of monitoring infrastructure and produce per-pod summary message sent through RabbitMQ to other services (Figure 135) that contains pod's history, pod labels and traffic measurements i.e. request/response rate per minute, and latency.

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 112 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

```

1 {
2   "pod_name": "app-bot-555c556d76-cdvh9",
3   "timestamp": "2024-12-11T08:32:13.956429Z",
4   "workload_id": "455ccf73-6d05-4f30-b05e-6a43dc211181",
5   "status": [
6     {
7       "status_type": "Ready",
8       "status": "True",
9       "last_transition_time": "2024-12-11T08:01:55Z",
10      "message": null
11    },
12    {
13      "status_type": "ContainersReady",
14      "status": "True",
15      "last_transition_time": "2024-12-11T08:01:55Z",
16      "message": null
17    },
18    {
19      "status_type": "PodReadyToStartContainers",
20      "status": "True",
21      "last_transition_time": "2024-12-11T08:00:10Z",
22      "message": null
23    },
24    {
25      "status_type": "Initialized",
26      "status": "True",
27      "last_transition_time": "2024-11-15T12:24:47Z",
28      "message": null
29    },
30    {
31      "status_type": "PodScheduled",
32      "status": "True",
33      "last_transition_time": "2024-11-15T12:24:46Z",
34      "message": null
35    }
36  ],
37  "labels": {
38    "app": "app-bot",
39    "linkerd.io/control-plane-ns": "linkerd",
40    "linkerd.io/proxy-deployment": "app-bot",
41    "linkerd.io/workload-ns": "demoapp",
42    "nemo.eu/workload": "455ccf73-6d05-4f30-b05e-6a43dc211181",
43    "pod-template-hash": "555c556d76",
44    "version": "v11"
45  },
46  "traffic_stats": {
47    "req_rate": 0.32079628767407103,
48    "res_rate": 2.1029078850633545,
49    "res_rate_by_code": {
50      "500": 0.17153690382571854,
51      "200": 1.9314609820376358
52    },
53    "res_time_quantiles_us": {
54      "p99": 18.702803738317762,
55      "p95": 11.644850813084098,
56      "p75": 7.683397683397683,
57      "p50": 4.876404494382022
58    }
59  }
60 }

```

Figure 135: Expected RabbitMQ message data model

4.4.3.2 NEMO workload monitoring – PPEF

In the following paragraphs the workload monitoring associated results are presented.

Computing workload intent

The PPEF collects the computing workload intent measurements (CPU and RAM) by querying the deployed monitoring tool (Thanos²⁵). The detailed description of the PPEF architecture and provided functionality is included in D3.2 [11]. The Figure 136 below illustrates CPU measurement collection and Figure 137 the RAM measurement collection.

²⁵ <https://thanos.io/>

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 113 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |



Figure 136: Workload – CPU usage

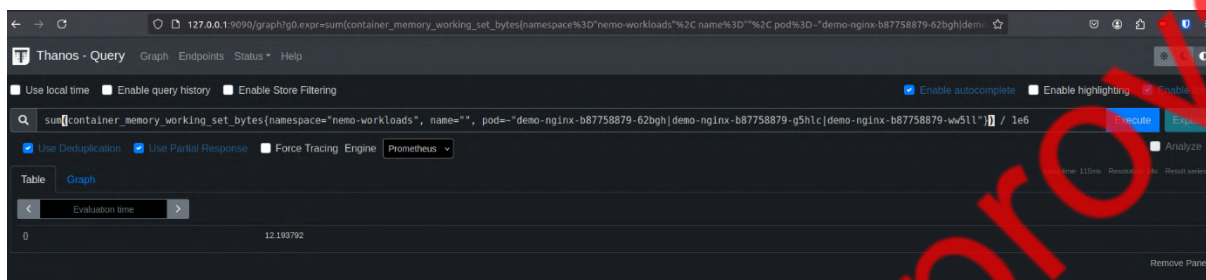


Figure 137: Workload - RAM usage

EnergyEfficiency intent

The Figure 138 below depicts the *Green Energy Consumption Rate* (the containers of the deployment “demo-nginx” are consuming approximately 1,23 joules per second averaged over the last 5 minutes).

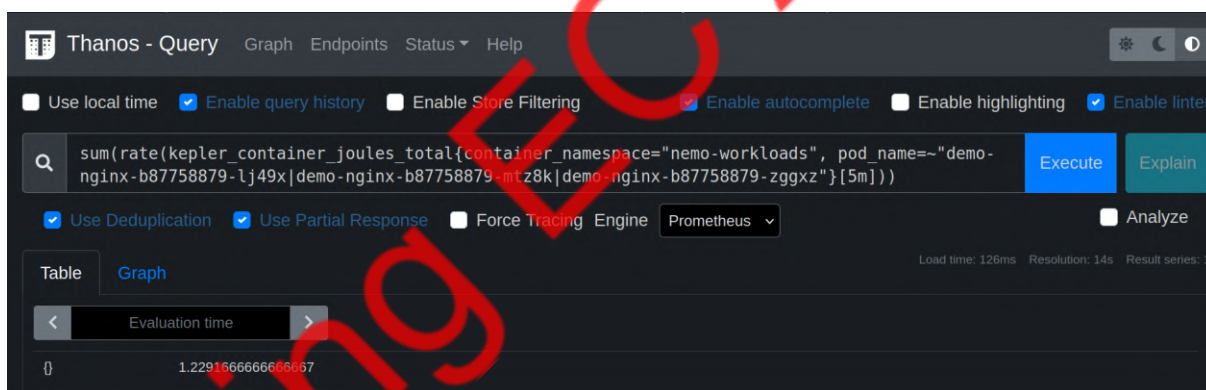


Figure 138: Workload - Energy consumption rate

NEMO workload Energy Efficiency shows that the service consumes 40k Joules for every second of CPU time as illustrated in the collected query below (Figure 139).

| | | | | | |
|----------------|---|----------------|----|----------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 114 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

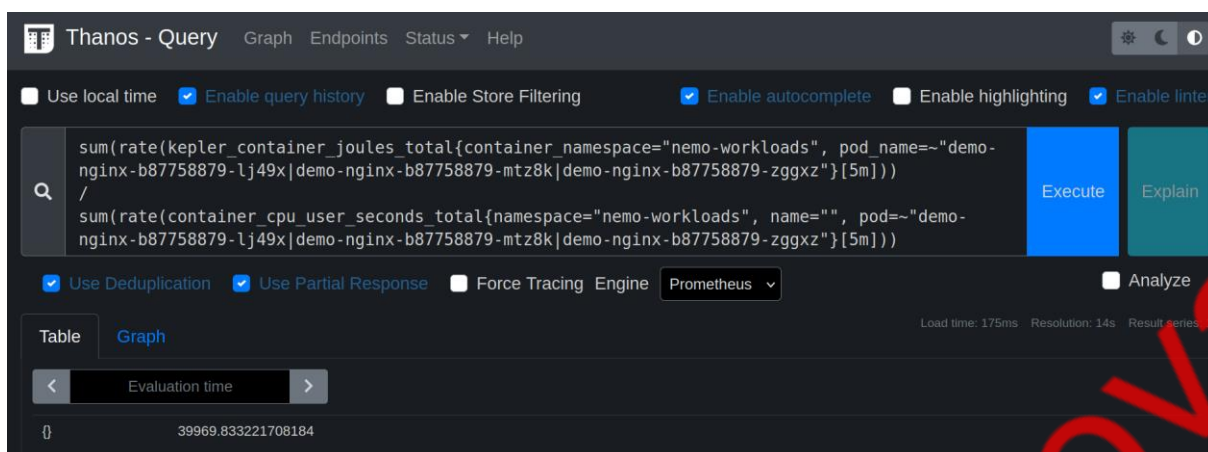


Figure 139: Workload - Energy efficiency

Energy consumption

The Energy consumption of a particular workflow is depicted in Figure 140 below.

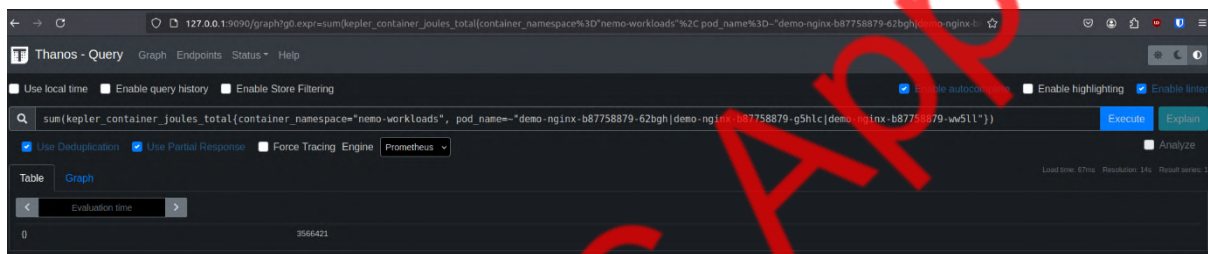


Figure 140: Workload - Energy consumption

The Energy Efficiency intents provisioning in *Intent-based API* is presented below. Here the NEMO user can assign expectation targets for the Energy Efficiency related expectations.

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 115 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

```

GET https://intent-api.nemo.onelab.eu/api/v1/intent/?not_fulfilled_state=COMPLIANT&not_fulfilled_state=DEGRADED

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize Text

112     feasibility_check_type: FEASIBLE
113     infeasibility_reason: null
114     last_updated_time: "2024-07-10T12:22:22.676617Z"
115     intent_contexts: []
116   - id: 4
117     user_label: EnergyCarbonEfficiency
118     intent_preemption_capability: "FALSE"
119     observation_period: 60
120     intent_expectations:
121       - id: 4
122         expectation_id: "1"
123         expectation_verb: ENSURE
124         expectation_object:
125           id: 4
126           object_type: NEMO_WORKLOAD
127           object_instance: d3c90585-371b-42e1-9f31-cd340fc126c1
128           context_selectivity: null
129           object_contexts: []
130         expectation_targets:
131           - id: 5
132             target_name: compEnergyEfficiency
133             target_condition: IS_GREATER_THAN
134             target_value_range: "40000"
135             target_contexts: []
136           - id: 6
137             target_name: compEnergyConsumption
138             target_condition: IS_LESS_THAN
139             target_value_range: "900"
140             target_contexts: []
141           - id: 7
142             target_name: greenEnergyConsumptionRate
143             target_condition: IS_GREATER_THAN
144             target_value_range: "90"
145             target_contexts: []
146         expectation_contexts: []

```

Figure 141: Intent-API EnergyEfficiency metrics update

4.4.3.3 NEMO Cluster monitoring

The Figure 142, Figure 143 and Figure 144 below summarize the cluster level metrics that are collected by the PPEF component for CPU, RAM and Disk storage respectively and communicated to the RabbitMQ.

| | | | | | | |
|----------------|--|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 116 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

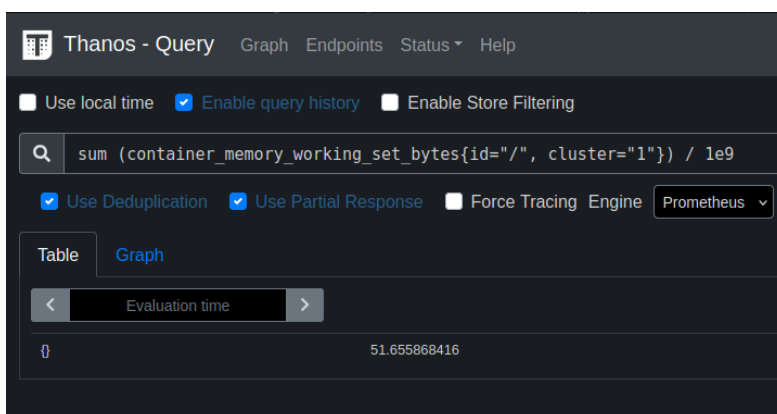


Figure 142: Cluster RAM usage

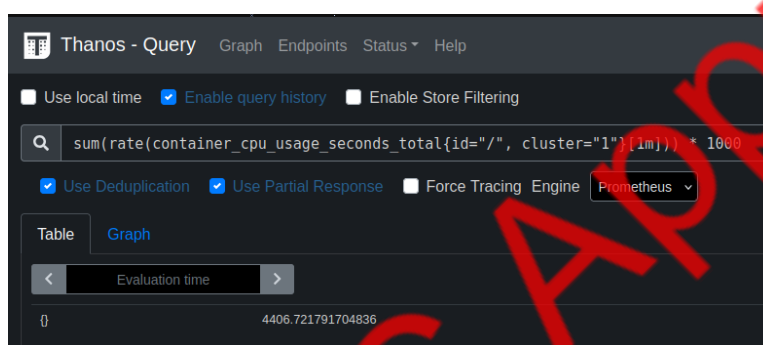


Figure 143: Cluster CPU usage

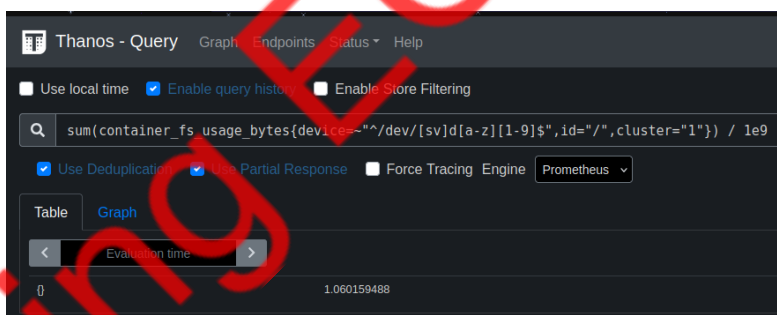


Figure 144: Cluster Disk usage

4.4.3.4 Cluster metrics to RabbitMQ

Figure 145 below presents the cluster level metrics communication to the RabbitMQ from the PPEF component.

```
+ pipenv run python3 cluster-metrics.py
Loading .env environment variables...
{"cluster": "1", "cpu_usage": "4406", "memory_usage": "51.657", "disk_usage": "1.6"}
[x] Sent evaluated intent!
```

Figure 145: cluster metrics published to RabbitMQ

| | | | | | | |
|----------------|---|----------------|----|----------|-------|---------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 117 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

4.4.4 Verification summary checklist

| Checklist for Test4: NEMO workload lifecycle management | | | | |
|---|--|-----|----|--|
| | | Yes | No | Comments |
| 2 | NEMO workload monitoring collects metrics that correspond to the NEMO workload (PPEF) | ✓ | | Success |
| 3 | The collected workload metrics are communicated to the Intent-API | ✓ | | Success |
| 4 | The NEMO Cluster monitoring collects measurements that concern the NEMO meta-OS operated clusters (PPEF) | ✓ | | Success |
| 5 | The collected cluster metrics are communicated to the RabbitMQ | ✓ | | Success |
| 6 | NEMO workload complementary monitoring (CMDT) | ✓ | | Success |
| 7 | The collected metrics are communicated to the Rabbit MQ | ✓ | | Success |
| 8 | The LCM aggregates the collected metrics and visualize them to the NEMO user | | ✓ | The LCM UI view that corresponds to this aspect is under development |

Table 13: Checklist for Test4

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 118 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

5 Conclusions

This deliverable provided insights on the scenario-driven integration activities that produced the first integrated NEMO meta-OS. In addition, the NEMO meta-OS cloud/edge infrastructure established in OneLab facilities that supported the integration activities along with the CI/CD environment and configuration was presented in detail.

Moreover, the NEMO meta-OS components that belong into the NEMO Service Management Layer namely, the Intent-based API, the IBMC, the LCM and the MOCA were described presenting their provided functionalities, the updated architectures, the interfaces and data models and initial results.

Finally, the document provided a comprehensive description of the integration steps that were followed as part of end-to-end scenarios that reflected the technical capacity of the first integration version of the NEMO meta-OS.

The verification results will feed enhancements in the development of the NEMO meta-OS components for the next integration cycle that will produce the final version of the NEMO meta-OS platform and will be documented in D4.3 “Advanced NEMO platform & laboratory testing results. Final version”.

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 119 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

6 References

- [1] NEMO, "D4.3 - Advanced NEMO platform & laboratory testing results. Final version," HORIZON - 101070118 - NEMO Deliverable Report, 2025.
- [2] NEMO, "D4.1 - Integration guidelines & initial NEMO WP4 integration," HORIZON - 101070118 - NEMO Deliverable Report, 2023.
- [3] NEMO, "D1.2 - NEMO meta-architecture, components and benchmarking. Initial version," HORIZON - 101070118 - NEMO Deliverable Report, 2023.
- [4] NEMO, "D2.3 - Enhancing NEMO Underlying Technology. Final version," HORIZON - 101070118 - NEMO Deliverable Report, 2024.
- [5] Z. Anastasakis, T.-H. Velivassaki, A. Voulkidis, S. Bourou, K. Psychogyios, D. Skias and T. Zahariadis, "FREDY: Federated Resilience Enhanced with Differential Privacy," *Future Internet*, vol. 15, no. 9, 2023.
- [6] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," *arxiv*.
- [7] S. Mwanje, A. Banerjee, J. Goerge, A. Abdelkader, G. Hannak, P. Szilágyi, T. Subramanya, J. Goser and T. Foth, "Intent-Driven Network and Service Management: Definitions, Modeling and Implementation," *ITU Journal on Future and Evolving Technologies*, vol. 3, no. 3, 2022.
- [8] R. Xu, M. Scott and S. Mwanje, "Enabling intelligence and automation for 5G Advanced Networks," 3GPP, 2023. [Online]. Available: <https://www.3gpp.org/technologies/intent>.
- [9] 3GPP, "3GPP TS 28.312 V18.3.0 (2024-03) - Technical Specification Group Services and System Aspects - Management and orchestration - Intent driven management services for mobile networks (Release 18)," 3GPP, 2024.
- [10] NEMO, "D1.3 - NEMO meta-architecture, components and benchmarking. Final version," HORIZON - 101070118 - NEMO Deliverable Report, 2024.
- [11] NEMO, "D3.2 - NEMO Kernel. Initial version," HORIZON - 101070118 - NEMO Deliverable Report, 2024.

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 120 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

7 Annex A – MOCA API & data models

The MOCA API swagger page (OpenApi) is online available in this location (<https://intent-api.nemo.onelab.eu/moca/api/v1/swagger/>)

7.1 MOCA Data models

MOCA allows for the exposure of the cluster registration process, as well as management of resource and accounting details. This section will provide the API documentation and the data models used by MOCA.

| Attribute | Data type | Description |
|-------------------|-----------|---|
| id | String | The id of the accounting event |
| type | String | The type of the accounting event |
| customer_id | String | The id of the customer (workloads, clusters) |
| tokens | number | The NEMO tokens that were deposited or withdrawn |
| balance | number | The NEMO tokens that the customer currently has left |
| balance_action_id | Integer | The id to retrieve the accounting event from the blockchain |
| timestamp | String | |

Table 14: MOCA AccountingEvents Data Model

| Attribute | Data type | Description |
|---------------|-----------|--|
| id | String | The id of the Cluster |
| cluster_name | String | The name of the Cluster that will be deployed |
| cpus | Integer | The number of CPUs of the Cluster |
| memory | Integer | The RAM of the Cluster in GB |
| storage | Integer | The disk storage of the Cluster in GB |
| availability | String | The percentage of time that the cluster is up (99.9%, 99%, 90%) |
| green_energy | String | The percentage of RES powering the cluster. (0%,20%,40%,60%,80%,100%) |
| cost | String | The cost type of a cluster (low cost, high performance) |
| cpu_base_rate | number | The CPU cost of the cluster by the CPU capacity of the cluster (in milliseconds) |

| | | | |
|-----------------------|---|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 121 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |

| | | |
|------------------|--------|---|
| memory_base_rate | number | The memory cost of the cluster by the memory capacity of the cluster (in MBs) |
| timestamp | String | |
| balance | number | The NEMO tokens of the cluster |

Table 15: MOCA ClusterResources Data Model

| Attribute | Data type | Description |
|-------------------|-----------|---|
| cluster_resources | String | The id of the ClusterResources |
| status | Array | The status of the deployment of the Cluster |
| timestamp | String | |

Table 16: MOCA ClusterState Data Model

| Attribute | Data type | Description |
|--------------|-----------|---|
| id | String | The id of the workload |
| cluster_name | String | The name of the cluster the workload is deployed to |
| status | String | The status of the deployment of the Cluster |
| cpus | number | The number of CPUs of the Application |
| memory | number | The RAM of the Application in MB |
| storage | number | The space of the volume in GB |
| timestamp | String | |
| balance | number | The NEMO tokens of the workload |
| user | Integer | The id of the Workload User |

Table 17: MOCA Workload Data Model

| Attribute | Data type | Description |
|-----------|-----------|--|
| cluster | String | The id of the ClusterState |
| link_cid | String | The CID of the Cluster config stored in IPFS |
| ipfs_link | String | The link to retrieve the Cluster config |
| timestamp | String | |

Table 18: MOCA IPFS Handler Data Model

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 122 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

| Attribute | Data type | Description |
|-------------|-----------|---|
| id | String | |
| user_id | String | The id of the user |
| workload_id | String | The id of the Workload the computation took place for |
| cluster_id | String | The id of the cluster |
| cpu | String | The cpu used by the workload |
| ram | Array | The ram used by the workload |
| tokens | Array | The NEMO tokens that were charged to the workload |
| timestamp | String | |

Table 19: MOCA WorkloadComputeTokensEvents Data Model

| Attribute | Data type | Description |
|-----------------|-----------|--|
| username | String | The user's username |
| balance | number | The balance of the user |
| smart_contracts | Array | The names of the smart contracts related to a user |

Table 20: MOCA UserSmartContracts Data Model

| Attribute | Data type | Description |
|--------------------|-----------|-------------|
| region | String | |
| high_demand | Boolean | |
| high_demand_cost | number | |
| regional_cpu_limit | number | |
| regional_ram_limit | number | |

Table 21: MOCA NemoTokenSetup Data Model

7.2 MOCA API endpoints

7.2.1 GET /api/v1/accounting_events

Returns all the accounting events related to a user (GET).

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

| | | | |
|-----------------------|--|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 123 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |

| | | | |
|-----|---------------------------------|--------|------------------|
| 201 | The accounting events of a user | Object | AccountingEvents |
|-----|---------------------------------|--------|------------------|

Table 22: GET Accounting Events responses

7.2.2 DELETE /cluster/delete/{id}

Delete a cluster based on its ID (DELETE).

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | The cluster has been deleted | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 23: DELETE Cluster responses

| Parameters | | | | |
|------------|----------------|----------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | The cluster id | True | string |

Table 24: DELETE Cluster parameters

7.2.3 POST /cluster/register

Register a cluster in NEMO meta-OS (POST)

| Parameters | | | | |
|--------------|----------------|------------------------------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| cluster_name | body | The cluster name | True | string |
| cpus | body | The cluster # of cpus | True | integer |
| memory | body | The cluster # of memory | True | float |
| storage | body | The cluster total storage capacity | True | float |
| availability | body | The cluster availability % | True | string |
| green_energy | body | The cluster RES powered % | True | string |
| cost | body | The cost category of the cluster | True | string |
| cpu_base | body | The cpu base cost for the cluster | True | float |

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 124 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

| | | | | |
|------------------|------|--------------------------------------|------|-------|
| memory_base_rate | body | The memory base cost for the cluster | True | float |
|------------------|------|--------------------------------------|------|-------|

Table 25: REGISTER cluster parameters

| Responses | | | |
|-----------|---|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |
| 406 | The smart contract rolled back (declined) the transaction | | |
| 201 | The Cluster ID | | |

Table 26: POST Cluster responses

7.2.4 GET /cluster/retrieve

Retrieve all the clusters' details related to a user (GET).

| Parameters | | | | |
|------------------|----------------|--------------------------------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | body | The id of the cluster | False | string |
| cluster_name | body | The cluster name | True | string |
| cpus | body | The cluster # of cpus | True | integer |
| memory | body | The cluster # of memory | True | float |
| storage | body | The cluster total storage capacity | True | float |
| availability | body | The cluster availability % | True | string |
| green_energy | body | The cluster RES powered % | True | string |
| cost | body | The cost category of the cluster | True | string |
| cpu_base | body | The cpu base cost for the cluster | True | float |
| memory_base_rate | body | The memory base cost for the cluster | True | float |
| timestamp | body | Timestamp | False | string |

Table 27: GET cluster parameters

| Responses | | | |
|-----------|---------------------------------------|-------------|------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The details of all records | Object | ClusterResources |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 28: GET Clusters responses

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 125 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

7.2.5 GET /cluster/retrieve/{id}

Retrieve a cluster's details related to a user (GET).

| Parameters | | | | |
|------------|----------------|----------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | The cluster id | True | string |

Table 29: GET Cluster with ID parameters

| Responses | | | |
|-----------|---------------------------------------|-------------|------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The details of selected records | Object | ClusterResources |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 30: GET Cluster with ID responses

7.2.6 PUT, PATCH /cluster/update/{id}

Update a cluster's attributes (PUT, PATCH).

| Parameters | | | | |
|------------|----------------|------------------------|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | The cluster id | True | |
| data | body | The cluster attributes | True | UpdateClusterResources |

Table 31: PUT, PATCH Cluster parameters

| Responses | | | |
|-----------|---|-------------|------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object | UpdateClusterResources |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |
| 406 | The smart contract rolled back (declined) the transaction | | |

Table 32: PUT, PATCH Cluster responses

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 126 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

7.2.7 POST /nemo_token_estimation_setup

Setup the region costs for that will be used in the workload usage calculation (POST).

| Responses | | | |
|-----------|---|-------------|-----------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The generated transaction hash | Object | TransactionHash |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |
| 406 | The smart contract rolled back (declined) the transaction | | |

Table 33: POST Region Costs responses

7.2.8 GET /nemo_token_setup_retrieve/{region}

Retrieve the information on the region costs (GET).

| Parameters | | | | |
|------------|----------------|-----------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| region | path | The region name | True | string |

Table 34: GET Region Costs parameters

| Responses | | | |
|-----------|---|-------------|------------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The region info | Object | NemoTokenSetupRetrieveRegion |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |
| 406 | The smart contract rolled back (declined) the transaction | | |

Table 35: GET Region Costs responses

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 127 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

7.2.9 GET /nemo_user_info

Retrieve the information of a logged in user (GET).

| Responses | | | |
|-----------|---------------------------------------|-------------|--------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The information of the user | Object | NemoUserInfo |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 36: GET user information responses

7.2.10 GET /workload/retrieve

Retrieve all the details of the workloads related to a user (GET).

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The details of all records | Object | Workload |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 37: GET workloads' details responses

7.2.11 GET /workload/retrieve/{id}

Retrieve the details of a workload based on its ID (GET).

| Parameters | | | | |
|------------|----------------|-----------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | The workload id | True | string |

Table 38: GET workload's details parameters

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The details of all records | Object | Workload |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 39: GET workload's details responses

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 128 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

7.2.12 GET /workload_computations/{id}

Retrieve all the events of a workload that show its resource usage details (GET).

| Parameters | | | | |
|------------|----------------|-----------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | The workload id | True | string |

Table 40: GET workload computation details parameters

| Responses | | | |
|-----------|---------------------------------------|-------------|-----------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | The details of all records | Object | WorkloadComputeTokensEvents |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Invalid permissions | | |

Table 41: GET workload computation details responses

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 129 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

8 Annex B – Intent-based API & data models

8.1 NEMO Intent-based API

The Intent-based API Server allows for exposure of NEMO functionalities, as well as management of intents and workloads. The API is available online at <https://intent-api.nemo.onelab.eu/api/v1/swagger/>.

8.2 Intent-API data models

| Attribute | Data type | Description | Comments |
|-----------|-----------|-------------|----------|
| username | String | | |
| password | String | | |
| token | String | | |

Table 42: Data model description: AuthToken

| Attribute | Data type | Description | Comments |
|--------------|-----------|----------------------------------|----------|
| cluster_name | String | The name of the cluster resource | |
| cpus | Integer | The number of the CPUs | |
| memory | Integer | The RAM of the cluster in GB | |
| storage | Integer | The storage of the cluster in GB | |

Table 43: Data model description: ClusterRegister

| Attribute | Data type | Description |
|-----------|-----------|--|
| link_id | String | The IPFS link id |
| ipfs_link | String | The IPFS link to retrieve the cluster config |

Table 44: Data model description: ClusterIpfs

| Attribute | Data type | Description |
|-----------|-------------|----------------------------------|
| id | String | The ID of the cluster |
| vm_name | String | The name of the cluster resource |
| cpus | Integer | The number of the CPUs |
| memory | Integer | The RAM of the cluster in GB |
| storage | Integer | The storage of the cluster in GB |
| endpoint | String | The endpoint of the Cluster |
| ipfs | ClusterIpfs | |

Table 45: Data model description: Cluster

| | | | |
|-----------------------|---|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 130 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |

| Attribute | Data type | Description |
|---------------------|-----------|-------------|
| id | Integer | |
| context_attribute | String | |
| context_condition | String | |
| context_value_range | String | |

Table 46: Data model description: Context

| Attribute | Data type | Description |
|---------------------|-----------|---|
| id | Integer | |
| object_type | String | It describes the expectation object type which can be supported by a specific intent handling function of MnS producer. |
| object_instance | String | |
| context_selectivity | String | How to select among the stated expectationContexts |
| object_contexts | Array | |

Table 47: Data model description: ExpectationObject

| Attribute | Data type | Description |
|--------------------|-----------|-------------|
| id | Integer | |
| target_name | String | |
| target_condition | String | |
| target_value_range | String | |
| target_contexts | Array | |

Table 48: Data model description: ExpectationTarget

| Attribute | Data type | Description |
|----------------------|-------------------|--|
| id | Integer | |
| expectation_id | String | A unique identifier of the intentExpectation within the intent |
| expectation_verb | String | |
| expectation_object | ExpectationObject | |
| expectation_targets | Array | |
| expectation_contexts | Array | |

Table 49: Data model description: IntentExpectation

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 131 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| Attribute | Data type | Description |
|------------------------------|-----------|---|
| id | Integer | |
| user_label | String | A user-friendly (and user assignable) name of the intent. |
| intent_preemption_capability | String | |
| observation_period | Integer | In seconds |
| intent_expectations | Array | |
| intent_report_reference | String | |
| intent_contexts | Array | |

Table 50: Data model description: Intent

| Attribute | Data type | Description |
|---------------------|-----------|-------------|
| context_attribute | String | |
| context_condition | String | |
| context_value_range | String | |

Table 51: Data model description: ContextInput

| Attribute | Data type | Description |
|---------------------|-----------|---|
| object_type | String | It describes the expectation object type which can be supported by a specific intent handling function of MnS producer. |
| object_instance | String | |
| context_selectivity | String | How to select among the stated expectationContexts |
| object_contexts | Array | |

Table 52: Data model description: ExpectationObjectInput

| Attribute | Data type | Description |
|--------------------|-----------|-------------|
| target_name | String | |
| target_condition | String | |
| target_value_range | String | |
| target_contexts | Array | |

Table 53: Data model description: ExpectationTargetInput

| Attribute | Data type | Description |
|------------------|-----------|--|
| expectation_id | String | A unique identifier of the intentExpectation within the intent |
| expectation_verb | String | |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 132 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | | |
|----------------------|------------------------|--|
| expectation_object | ExpectationObjectInput | |
| expectation_targets | Array | |
| expectation_contexts | Array | |

Table 54: Data model description: IntentExpectationInput

| Attribute | Data type | Description |
|------------------------------|-----------|---|
| user_label | String | A user-friendly (and user assignable) name of the intent. |
| context_selectivity | String | How to select among the stated intentContexts |
| intent_preemption_capability | String | |
| observation_period | Integer | In seconds |
| intent_expectations | Array | |
| intent_contexts | Array | |

Table 55: Data model description: IntentInput

| Attribute | Data type | Description |
|-----------|-------------|-------------|
| intent | IntentInput | |

Table 56: Data model description: IntentInputAttribute

| Attribute | Data type | Description |
|-----------|-----------|----------------|
| intent | Integer | Intent ID (PK) |

Table 57: Data model description: IntentOutput

| Attribute | Data type | Description |
|--------------------|-----------|-------------|
| target_name | String | |
| target_condition | String | |
| target_value_range | String | |

Table 58: Data model description: TargetTemplate

| Attribute | Data type | Description | Comments |
|--------------------|-----------|---------------------------|-------------|
| instance_id | String | NEMO Workload instance ID | 'uuid' |
| intent_type | String | Intent userLabels | |
| service_start_time | String | Optional | 'date-time' |
| service_end_time | String | Optional | 'date-time' |
| targets | Array | Expectation targets | |
| instance_id | String | NEMO Workload instance ID | 'uuid' |

Table 59: Data model description: IntentTemplate

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 133 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

| Attribute | Data type | Description |
|-----------|-----------|-------------------|
| action | String | Action to perform |

Table 60: Data model description: IntentActionInput

| Attribute | Data type | Description |
|--------------------|-----------|-------------|
| target_id | Integer | |
| target_value_range | String | |

Table 61: Data model description: IntentTargetUpdate

| Attribute | Data type | Description | Comments |
|------------|-----------|--|----------|
| id | Integer | | |
| username | String | Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. | |
| email | String | | 'email' |
| first_name | String | | |
| last_name | String | | |

Table 62: Data model description: User

| Attribute | Data type | Description | Comments |
|-----------|-----------|-------------------------------------|----------|
| id | Integer | | |
| name | String | The maintainers name (required) | |
| email | String | The maintainers email (optional) | 'email' |
| url | String | A url for the maintainer (optional) | 'uri' |
| chart | Integer | | |

Table 63: Data model description: WorkloadDocumentChartMaintainer

| Attribute | Data type | Description |
|------------|-----------|--|
| id | Integer | |
| name | String | The name of the chart |
| version | String | A SemVer 2 version string |
| repository | String | The repository URL or alias ("repo-name") (optional) |
| condition | String | A yaml path that resolves to a boolean, used for enabling/disabling charts (e.g. subchart1.enabled) (optional) |

| | | | | | | |
|-----------------------|---|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 134 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |

| | | |
|---------------|---------|---|
| tags | Array | Tags can be used to group charts for enabling/disabling together |
| import_values | Array | ImportValues holds the mapping of source values to parent key to be imported. |
| alias | String | Alias to be used for the chart. Useful when you have to add the same chart multiple times |
| chart | Integer | |

Table 64: Data model description: WorkloadDocumentChartDependency

| Attribute | Data type | Description |
|----------------------|-----------|------------------------------------|
| id | Integer | |
| container_registries | Object | The mapped container registries |
| manifests | Array | The default generated manifests |
| values | Object | The chart default values |
| resource_mappings | Object | The container resource mappings |
| memory_requests | Integer | Total memory requests (in bytes) |
| memory_limits | Integer | Total memory limits (in bytes) |
| cpu_requests | Integer | Total CPU requests (in milli cpus) |
| cpu_limits | Integer | Total CPU limits (in milli cpus) |

Table 65: Data model description: WorkloadDocumentChartMetadata

| Attribute | Data type | Description | Comments |
|--------------|-------------------------------|---|----------|
| id | Integer | | |
| maintainers | Array | | |
| dependencies | Array | | |
| metadata | WorkloadDocumentChartMetadata | | |
| api_version | String | The chart API version (required) | |
| name | String | The name of the chart (required) | |
| version | String | A SemVer 2 version string | |
| kube_version | String | A SemVer range of compatible Kubernetes versions (optional) | |
| description | String | A single-sentence description of this project (optional) | |
| type | String | The type of the chart (optional) | |
| keywords | Array | A list of keywords about this project (optional) | |
| home | String | The URL of this projects home page (optional) | 'uri' |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 135 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | | | |
|-------------|---------|--|-------|
| sources | Array | A list of URLs to source code for this project (optional) | |
| icon | String | A URL to an SVG or PNG image to be used as an icon (optional) | 'uri' |
| app_version | String | The version of the app that this contains (optional). Needn't be SemVer. Quotes recommended. | |
| deprecated | Boolean | Whether this chart is deprecated (optional, boolean) | |
| annotations | Object | A list of annotations keyed by name (optional) | |

Table 66: Data model description: WorkloadDocumentChart

| Attribute | Data type | Description | Comments |
|------------------|-----------------------|---|-------------|
| id | Integer | | |
| user | User | The NEMO user | |
| chart | WorkloadDocumentChart | The associated helm chart | |
| created | String | | 'date-time' |
| modified | String | | 'date-time' |
| name | String | The workload document name | |
| version | String | A SemVer 2 version string | |
| schema | Object | The document schema | |
| intents | Array | List of supported intents | |
| type | String | The workload document type | |
| status | String | The workload document status | |
| ingress_support | Boolean | Whether the workload document can be exposed via NEMO | |
| enabled | Boolean | If the workload document is enabled | |
| rejection_reason | String | Rejection reason | |

Table 67: Data model description: WorkloadDocumentList

| Attribute | Data type | Description |
|-----------|-----------|------------------------------|
| id | Integer | |
| status | String | The workload document status |
| user | Integer | The NEMO user |
| name | String | The workload document name |
| version | String | A SemVer 2 version string |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 136 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | | |
|-----------------|---------|---|
| schema | Object | The document schema |
| type | String | The workload document type |
| intents | Array | List of supported intents |
| ingress_support | Boolean | Whether the workload document can be exposed via NEMO |

Table 68: Data model description: WorkloadDocumentCreate

| Attribute | Data type | Description | Comments |
|------------------------|-----------|--------------------------------------|-------------|
| id | Integer | | |
| type | String | Lifecycle event type | |
| deployment_cluster | String | The NEMO deployment cluster | |
| migration_from_cluster | String | The NEMO migration from cluster | |
| migration_to_cluster | String | The NEMO migration to cluster | |
| timestamp | String | The timestamp of the lifecycle event | 'date-time' |

Table 69: Data model description: WorkloadDocumentLifecycleEvent

| Attribute | Data type | Description | Comments |
|--------------------|-----------|---|-------------|
| id | Integer | | |
| lifecycle_events | Array | | |
| created | String | | 'date-time' |
| modified | String | | 'date-time' |
| instance_id | String | NEMO unique workload document instance identifier | 'uuid' |
| release_name | String | The workload document instance release name | |
| status | String | The workload document instance status | |
| lifecycle_metadata | Array | The lifecycle metadata associated with the instance | |
| cluster_name | String | The NEMO cluster name that the instance resides in | |
| ingress_enabled | Boolean | Whether the instance should be exposed via NEMO | |
| ingress_metadata | Object | Ingress metadata | |
| workload_document | Integer | The workload document | |

Table 70: Data model description: WorkloadDocumentInstance

| Attribute | Data type | Description |
|-----------|-----------|------------------------------|
| name | String | The workload document name |
| version | String | A SemVer 2 version string |
| schema | Object | The document schema |
| type | String | The workload document type |
| status | String | The workload document status |
| user | Integer | The NEMO user |

| | | | | | | | |
|----------------|---|----------------|----|----------|-----|---------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | | Page: | 137 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: | final |

| | | |
|-----------------|---------|---|
| intents | Array | List of supported intents |
| ingress_support | Boolean | Whether the workload document can be exposed via NEMO |

Table 71: Data model description: WorkloadDocumentUpdate

| Attribute | Data type | Description |
|-----------------|-----------|-----------------------------------|
| release_name | String | The release name |
| values_override | Object | values.yaml override |
| include_crds | Boolean | Include CRDS |
| is_upgrade | Boolean | If its upgrade |
| namespace | String | Namespace to associate with |
| no_hooks | Boolean | No hooks flag |
| ingress_enabled | Boolean | Expose workload instance via NEMO |
| cluster_name | String | Target cluster override |

Table 72: Data model description: WorkloadDocumentTemplateInput

8.3 Intent-based API endpoints

8.3.1 POST /api/v1/auth/login/

Create a new auth token for the user (POST)

| Parameters | | | | |
|------------|----------------|-------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| data | body | | True | AuthToken |

Table 73: POST authorization token parameters

8.3.2 POST /api/v1/auth/logout/

Clears the token associated with the user. (POST)

| Responses | | | |
|-----------|-------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | | Object | AuthToken |

Table 74: POST authorization logout responses

8.3.3 POST /api/v1/cluster/register/

This endpoint writes a message to the rabbitmq topic that MOCA component listens to. This is performed in asynchronous manner. (POST)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 204 | No Content | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

| | | | |
|-----------------------|---|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 138 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |

Table 75: POST cluster registration responses

| Parameters | | | | |
|------------|----------------|-------------|----------|-----------------|
| Attribute | Parameter Type | Description | Required | Data type |
| data | body | | True | ClusterRegister |

Table 76: POST cluster registration parameters

8.3.4 GET /api/v1/cluster/retrieve/

Retrieve cluster details from MOCA component (GET)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | Created | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 77: GET cluster retrieve responses

8.3.5 GET /api/v1/cluster/retrieve/{id}/

Retrieve a single cluster details from MOCA component (GET)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object list | Cluster |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 78: GET cluster retrieve (id) responses

| Parameters | | | | |
|------------|----------------|-------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | | True | |

Table 79: GET cluster retrieve (id) parameter

8.3.6 GET, POST /api/v1/intent/

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object list | Cluster |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 139 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

| | | | |
|-----|--------------------|--|--|
| 404 | Resource not found | | |
|-----|--------------------|--|--|

Table 80: GET/POST intent responses

| Parameters | | | | |
|---------------------|----------------|---------------------|----------|----------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| user_label | query | user_label | False | String |
| object_instance | query | object_instance | False | String |
| fulfilment_status | query | fulfilment_status | False | String |
| not_fulfilled_state | query | not_fulfilled_state | False | String |
| intent_id | query | intent_id | False | String |
| data | body | | True | IntentInputAttribute |

Table 81: GET/POST intent parameters

| Responses | | | |
|-----------|-------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object list | Intent |

Table 82: GET/POST intent responses

8.3.7 POST /api/v1/intent/template/

Creates an Intent with the given template (POST)

| Responses | | | |
|-----------|---------------------------------------|-------------|--------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | | Object | IntentOutput |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 83: POST create intent responses

| Parameters | | | | |
|------------|----------------|-------------|----------|----------------|
| Attribute | Parameter Type | Description | Required | Data type |
| data | body | | True | IntentTemplate |

Table 84: POST create intent parameters

8.3.8 GET /api/v1/intent/types/

Lists the valid intent types (GET)

| Responses | | | |
|-----------|-------------|-------------|--------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | | Object | IntentOutput |

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 140 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

| | | | |
|-----|---------------------------------------|--|--|
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 85: GET intent types responses

8.3.9 PUT /api/v1/intent/{id}/action/

Perform an action to a given Intent (PUT)

| Responses | | | |
|-----------|-------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | | |

Table 86: PUT intent action responses

| Parameters | | | | |
|------------|----------------|-------------|----------|-------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | | True | |
| data | body | | True | IntentActionInput |

Table 87: PUT intent action request

8.3.10 PUT /api/v1/intent/{id}/target/

Intent has to be in a valid state. It is best to use this in a rest api tool, e.g. postman and send data via ``application/yaml`` in order to derive data types better. (PUT)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | Ok | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 88: PUT intent's target (id) action responses

| Parameters | | | | |
|------------|----------------|-------------|----------|--------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | | True | |
| data | body | | True | IntentTargetUpdate |

Table 89: PUT intent's target (id) action parameters

8.3.11 GET, POST /api/v1/workload/

List or Create a new workload document(s) (GET)

| Responses | | | |
|-----------|-------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |

| | | | | | |
|-----------------------|---|-----------------------|-------|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 141 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | Status: | final | | |

| | | | |
|-----|---------------------------------------|--|--|
| 200 | Ok | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 90: GET workload documents list responses

| Parameters | | | | |
|------------|----------------|-------------|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| name | query | name | False | String |
| version | query | version | False | String |
| data | body | | True | WorkloadDocumentCreate |

Table 91: GET workload documents list parameters

8.3.12 List or Create a new workload document(s) (POST)

| Responses | | | |
|-----------|-------------|-------------|----------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object list | WorkloadDocumentList |

Table 92: POST workload document responses

| Parameters | | | | |
|------------|----------------|-------------|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| name | query | name | False | String |
| version | query | version | False | String |
| data | body | | True | WorkloadDocumentCreate |

Table 93: POST workload document parameters

8.3.13 GET /api/v1/workload/instance/

Lists all the workload documents instances (GET)

| Responses | | | |
|-----------|-------------|-------------|------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | | Object | WorkloadDocumentCreate |

Table 94: GET workload instances responses

| Parameters | | | | |
|-------------------|----------------|-------------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| release_name | query | release_name | False | String |
| cluster_name | query | cluster_name | False | String |
| workload_document | query | workload_document | False | String |
| status | query | status | False | String |

Table 95: GET workload instances parameters

| | | | | | |
|-----------------------|--|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 142 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | Status: | | | final |

8.3.14 PUT /api/v1/workload/instance/{instance_id}/delete/

Propagate a workload document instance deletion request for a deployed workload instance to the MO. (PUT)

| Responses | | | |
|-----------|-------------|-------------|--------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object list | WorkloadDocumentInstance |

Table 96: PUT workload instance delete responses

| Parameters | | | | |
|-------------|----------------|-------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| instance_id | path | | True | |

Table 97: PUT workload instance delete parameters

8.3.15 GET /api/v1/workload/instance/{instance_id}/manifests/

Fetch workload instance manifests in a single ``.yaml`` format. (GET)

| Responses | | | |
|-----------|-------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | | |

Table 98: GET workload instance manifests responses

| Parameters | | | | |
|-------------|----------------|-------------------------------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| instance_id | path | | True | |
| instance_id | path | The Workload Document instance uuid | True | String |

Table 99: GET workload instance manifests parameters

8.3.16 POST /api/v1/workload/upload/

The following must apply:

- The helm chart must be packed as ``.tgz`` (by running helm package).
- The helm chart must have a matching (name, version) pair with the associated workload document. - The helm chart must have a valid structure, files ``Chart.yaml``, ``values.yaml`` and folder ``templates`` are mandatory.
- The helm chart must be able to render (via helm template) without any errors.
- The helm chart underlying containers images must exist and be reachable by NEMO Intent API (either public or private registries with appropriate imagePullSecrets).

If everything is OK, the helm chart is uploaded to the NEMO S3 Helm Repository. After successful upload, the Workload Document is set to ``status=onboarding`` for further validation. After successful validation, the Workload Document is set to ``status=accepted`` or ``status=rejected`` if validation has failed. RabbitMQ is notified as per README.md (POST)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | Kubernetes Manifests | Object list | |
| 400 | Provided data is invalid or malformed | | |

| | | | |
|-----------------------|---|-----------------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | Page: | 143 of 146 |
| Reference: | D4.2 | Dissemination: | PU |
| | Version: | 1.0 | Status: final |

| | | | |
|-----|---------------------------------------|--|--|
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 100: POST workload upload request responses

| Parameters | | | | |
|------------|----------------|---|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |
| file | formData | Packaged helm chart in ```.tgz``` extension | True | file |
| name | formData | Workload Name to associate with | True | String |
| version | formData | Workload Version to associate with | True | String |

Table 101: POST workload upload request parameters

8.3.16.1 GET, PUT, PATCH, DELETE /api/v1/workload/{id}/

Update & Delete operations are only allowed when a workload document is in ``status=pending`` and the same user is performing the operation. (GET)

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | Created | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 102: GET workload responses

| Parameters | | | | |
|------------|----------------|--|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | A unique integer value identifying this Workload Document. | True | |
| data | body | | True | WorkloadDocumentUpdate |
| data | body | | True | WorkloadDocumentUpdate |

Table 103: GET workload parameters

Update & Delete operations are only allowed when a workload document is in ``status=pending`` and the same user is performing the operation. (PUT)

| Responses | | | |
|-----------|-------------|-------------|----------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object | WorkloadDocumentList |

Table 104: PUT workload responses

| Parameters | | | | |
|------------|----------------|-------------|----------|-----------|
| Attribute | Parameter Type | Description | Required | Data type |

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 144 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

| | | | | |
|------|------|--|------|------------------------|
| id | path | A unique integer value identifying this Workload Document. | True | |
| data | body | | True | WorkloadDocumentUpdate |
| data | body | | True | WorkloadDocumentUpdate |

Table 105: PUT workload parameters

Update & Delete methods are only allowed when a workload document is in ``status=pending`` and the same user is performing the operation. (PATCH)

| Responses | | | |
|-----------|-------------|-------------|------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object | WorkloadDocumentUpdate |

Table 106: PATCH workload responses

| Parameters | | | | |
|------------|----------------|--|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | A unique integer value identifying this Workload Document. | True | |
| data | body | | True | WorkloadDocumentUpdate |
| data | body | | True | WorkloadDocumentUpdate |

Table 107: PATCH workload parameters

Update & Delete operations are only allowed when a workload document is in ``status=pending`` and the same user is performing the operation. (DELETE)

| Responses | | | |
|-----------|-------------|-------------|------------------------|
| HTTP Code | Description | Schema Type | Data Model |
| 200 | | Object | WorkloadDocumentUpdate |

Table 108: DELETE workload responses

| Parameters | | | | |
|------------|----------------|--|----------|------------------------|
| Attribute | Parameter Type | Description | Required | Data type |
| id | path | A unique integer value identifying this Workload Document. | True | |
| data | body | | True | WorkloadDocumentUpdate |
| data | body | | True | WorkloadDocumentUpdate |

Table 109: DELETE workload parameters

8.3.16.2 POST /api/v1/workload/{id}/template/

Set header ``Accept`` to ``application/json`` or ``application/yaml`` (default). This action creates a workload document instance with a unique NEMO workload identifier (``instance_id``). RabbitMQ is notified as per README.md (POST)

| Parameters | | | | |
|------------|--|--|--|--|
|------------|--|--|--|--|

| | | | | | |
|-----------------------|---|-----------------------|----|-----------------|------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | Page: | 145 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 |
| | | | | Status: | final |

| Attribute | Parameter Type | Description | Required | Data type |
|-----------|----------------|-------------|----------|-------------------------------|
| id | path | | True | |
| data | body | | True | WorkloadDocumentTemplateInput |

Table 110: POST workload document instance parameters

| Responses | | | |
|-----------|---------------------------------------|-------------|------------|
| HTTP Code | Description | Schema Type | Data Model |
| 201 | Created | | |
| 400 | Provided data is invalid or malformed | | |
| 401 | Invalid credentials | | |
| 403 | Forbidden from performing this action | | |
| 404 | Resource not found | | |

Table 111: POST workload document instance responses

| | | | | | | |
|-----------------------|--|-----------------------|----|-----------------|--------------|----------------------|
| Document name: | D4.2 Advanced NEMO platform & laboratory testing results. Initial version | | | | Page: | 146 of 146 |
| Reference: | D4.2 | Dissemination: | PU | Version: | 1.0 | Status: final |